IBM System/3
Disk Systems
RPG II
Logic Manual

Program Numbers:

- 5702-RG1 (Model 10 Disk System)
- 5703-RG1 (Model 6)
- 5704-RG1 (Model 15)
- 5704-RG2 (Model 15)
- 5705-RG1 (Model 12)

LY21-0501-5
File No. S3-28

Sixth Edition (December 1975)

This is a major revision of, and obsoletes, LY21-0501-4 and Technical Newsletters LN21-5252 and LN21-7761. Information for the Model 12 RPG II Compiler has been added. Changes are indicated by a vertical line to the left of the change; new or extensively revised illustrations are denoted by a bullet (●) to the left of the figure title.

This edition, a part of version 04, modification 00 of the IBM System/3 Model 15 RPG II (Program Product Number 5704-RG1), also applies to the IBM System/3 Model 6 RPG II (Program Product Number 5703-RG1), IBM System/3 Model 10 Disk System RPG II (Program Product Number 5702-RG1), and IBM System/3 Model 12 RPG II (Program Product Number 5705-RG1). This edition remains in effect for all subsequent versions and modifications unless specifically altered by a new edition or a technical newsletter. Changes are continually made to the specifications herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/3 Bibliography,* Order Number GC20-8080, for the editions that are applicable and current.

Request for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901.

This publication describes the internal logic of the RPG II compilers and associated object programs for the following IBM systems:

- IBM System/3 Model 6
- IBM System/3 Model 8
- IBM System/3 Model 10 Disk System
- IBM System/3 Model 12
- IBM System/3 Model 15

The System/3 Model 8 is supported by System/3 Model 10 Disk System control programming and program products. The facilities described in this publication for the Model 10 are also applicable to the Model 8, although the Model 8 is not referenced. It should be noted that not all devices and features which are available on the Model 10 are available on the Model 8. Therefore, Model 8 users should be familiar with the contents of IBM System/3 Model 8 Introduction, GC21-5114.

The purpose of the RPG II compiler is to produce an RPG II object program. This manual enables the reader to determine the logic of specific areas of the object program and to relate these areas to the program listing and dump.

In this publication the Model 15 logic (except that designated as 5704-RG2) refers to both System/3 Program Numbers 5704-RG1 and 5704-RG2.

Related Publications

These IBM System/3 reference manuals are recommended for additional information:

Model 6

- *Components Reference Manual,* GA34-0001
- *Halt Guide,* GC21-7541
- *RPG II Reference Manual,* SC21-7517

Model 10 Disk System

- *Components Reference Manual,* GA21-9236
- *Halt Guide,* GC21-7540
- *RPG II Reference Manual,* SC21-7504

Model 12

- *Components Reference Manual,* GA21-9236
- *Halt Guide,* GC21-5145
- *RPG II Reference Manual,* SC21-7504

Model 15

- *Components Reference Manual,* GA21-9236
- *System Messages,* GC21-5076
- *RPG II Reference Manual,* SC21-7504

The following program logic manuals are reference in this manual via function/module names. The following charts tie the function/module names to the correct manual:

Model 6, Model 10 Disk System, and Model 12.

ALLOCATE	.	.	X	X
OPEN	.	X	.	X
CLOSE	.	X	.	X
EOJ	.	.	X	X
SYSLOG	.	.	X	X
D. M. (data management)	.	X	.	X
ROLLOUT	.	.	X	X
BSC-IOCS	X			

Model 12 System Control Program Logic Manual, SY21-0046

Disk Systems System Control Program Logic Manual, SY21-0502 (Models 6 and 10)

Disk Systems Data Management and Input/Output Supervisor Logic Manual, SY21-0512 (Models 6 and 10)

Disk Systems Binary Synchronous Communications Programming Support Input/Output Control System Logic Manual, SY21-0526

Model 15

ALLOCATE	.	.	X
OPEN	.	X	
CLOSE	.	X	
EOJ	.	.	X
SYSLOG	.	.	X
D. M. (data management)	.	X	
ROLLOUT	.	.	X
BSC-IOCS	X		

Supervisor and IOS Logic Manual, SY21-0033

Data Management Logic Manual, SY21-0034

Binary Synchronous Communications Programming Support Input/Output Control System Logic Manual, SY21-0526

IBM System/3 Model 15 System Data Areas and Diagnostic Aids, SY21-0032, contains all system data area formats.

How this Publication is Organized

This publication is divided into the following sections:

1. *Introduction* contains an overview of operational, environmental, and physical characteristics of the compiler.
2. *Program Logic* describes the functions of each phase and the program flow from phase to phase.
3. *Data Areas* describes the contents of all data areas used by two or more phases.
4. *Object Program* contains the structure, logic flow, and storage layout of the object program.
5. *Appendixes* describe the flowcharting techniques used in this publication and the Dump Facility.

The IBM System/3 Model 6, Model 10 Disk System, Model 12, and Model 15 each use a separate, disk resident RPG II compiler. These compilers convert RPG II source programs into machine language object programs. The compilers can also produce source program listing with diagnostic messages. The compilers punch out the object programs on cards or enter the object program into an object library.

Since the Models 6, 10, 12, and 15 compilers are similar both in function and physical characteristics, the term *compiler* will refer to all four compilers. Descriptions of compiler functions and routines apply to the four compilers unless otherwise noted.

COMPILER OPERATION

The RPG II Compiler consists of six groups of phases necessary to create an object program:

Input and Compression phases
Assign and Diagnostic phases
Assemble I phases
Pre-Assemble phases
Assemble II phases
Overlay phases

There are approximately 130 phases that make up the six groups of phases. As each phase is brought into the Compiler Phase Area, the previous phase is overlaid. Figure 1-1 shows the order the groups of phases are loaded and also lists the functions of each group of phases.

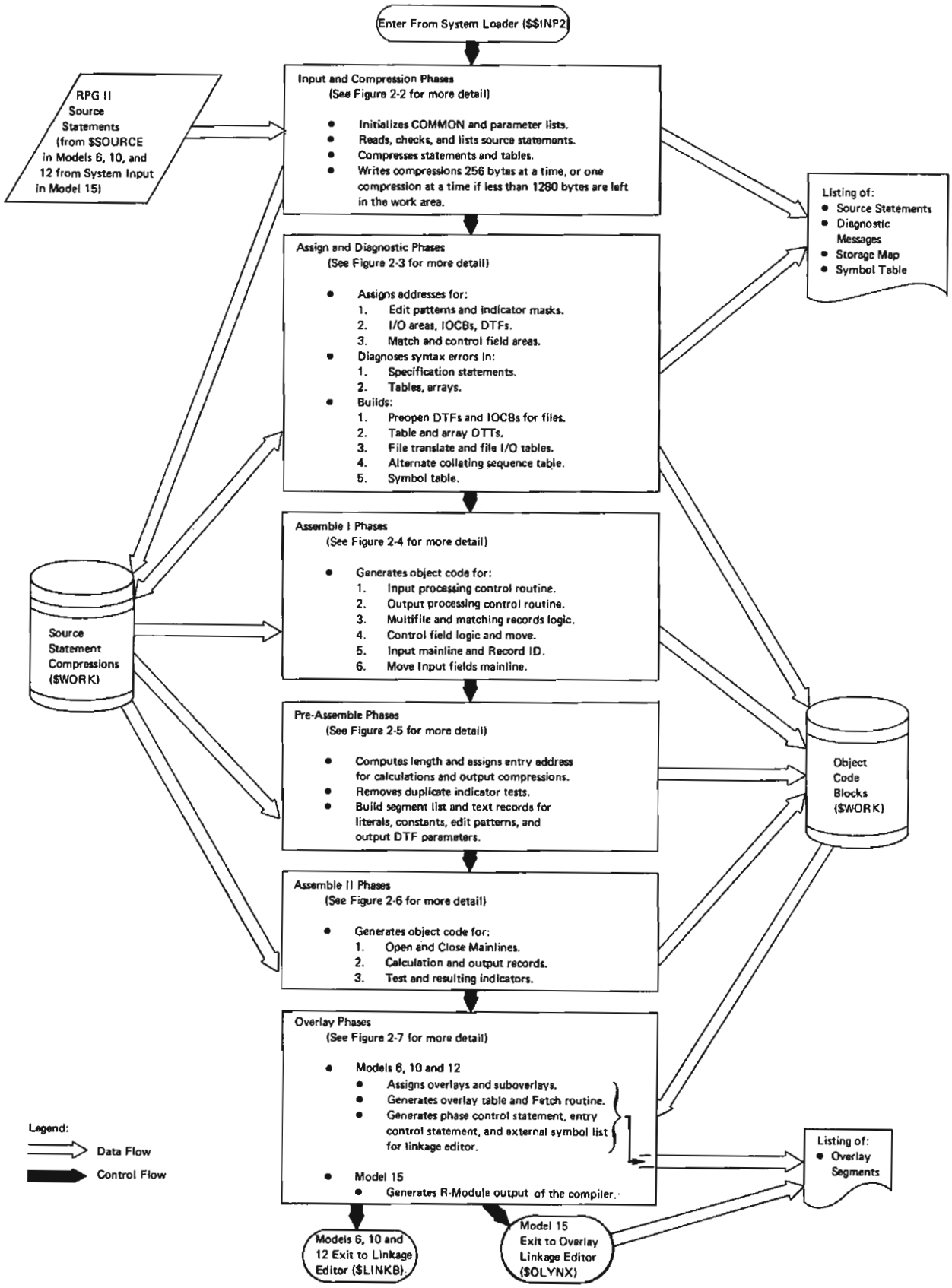
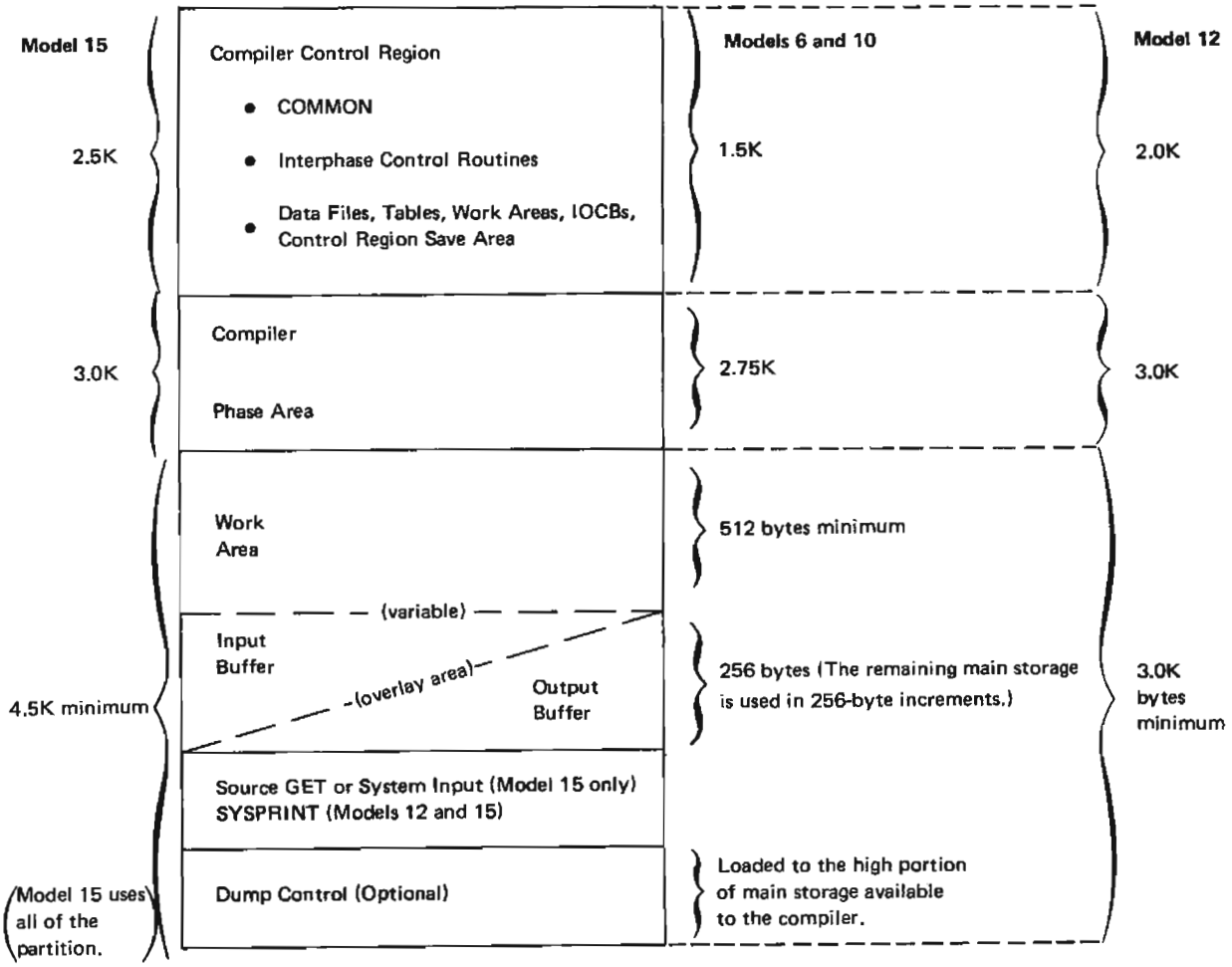


Figure 1-1. RPG II Compiler Overview

COMPILER CONTROL REGION

The Compiler Control Region, defined in phases $RPG and $RPIC, contains information needed by more than one phase. Figure 1-2 shows the contents of the Compiler Control Region. For a detailed description of this area see *Data Areas.*



- Figure 1-2. Layout of Main Storage During Operation of RPG II Compiler

LINKAGE BETWEEN PHASES

Phases are linked together with a branch to Interphase Control routine DRGCZZ and the 4-byte name of the next phase. The calling routine branches to the system loader which loads the next phase into main storage following the Interphase Control routines. Each new phase overlays the previous phase. XR1 and XR2 are not saved between phases. Some phases pass information to subsequent phases in the form of tables and constants stored temporarily in high-order main storage of the Compiler Control Region.

This section consists of a series of diagrams showing the functions, input, and output of each phase and each Interphase Control routine.

COMPILER PHASE DESCRIPTIONS

Figure 2-1 shows charting techniques used on Figures 2-2 through 2-7 to describe each of the phases. Each figure describes all the phases within a particular group of phases.

Input and Compression Phases

The Input and Compression phases (Figure 2-2) read the RPG II specifications and the user's source statements, diagnose and list the source program, and place a compressed version of the source program in $WORK.

Assign and Diagnostic Phases

The Assign and Diagnostic phases (Figure 2-3) assign addresses to all fields and data areas in the object program. These assigned areas are part of the overlay root segment and may not be overlaid. All noncode areas, buffers, control blocks, hold areas, work areas, input record fields, and tables are in the root segment. The Assign and Diagnostic phases provide comprehensive error detection by locating errors not found by the Input and Compression phases. The error detection routines analyze errors both within and across compressions. Errors found are indicated by error numbers stored on disk and later printed out.

Assemble I Phases

The Assemble I phases (Figure 2-4) generate all object code blocks except the calculations and output specified in the source program. Each phase checks for particular information in the compressions and generates code if that information is found. The generated object code blocks are stored in $WORK.

Pre-Assemble Phases

The Pre-Assemble phases (Figure 2-5) eliminate repetitive indicator testing throughout the object program and mark duplicate code segments, literals, and edit words for later deletion. They also calculate the length of code to be generated for calculations and output, assign relative addresses to code segments, select library subroutines, and generate a segment list for the Overlay phases.

Assemble II Phases

The Assemble II phases (Figure 2-6) generate the object code which performs calculations, output, initialization, and end-of-job operations. The generated object code blocks are stored in $SOURCE.

Overlay Phases

The Overlay phases (Figure 2-7) determines the overlay structure of the object program, sort the blocks of generated object code into the required overlay segments, and place the object program in the object library.

For Models 6, 10, and 12, the Overlay phases determine the overlay structure of the object program, sort the blocks of generated object code into the required overlay segments, and place the object program in the object library.

For the Model 15, the Overlay phases generate the R-module output of the compiler for input to the Overlay Linkage Editor.

The Overlay phases are shown in Figure 2-7.

Dump Control Phase

This phase is loaded to the high portion of the main storage available to the compiler when dump functions are requested. It intercepts all next phase calls and all text output calls. (See Appendix B.)

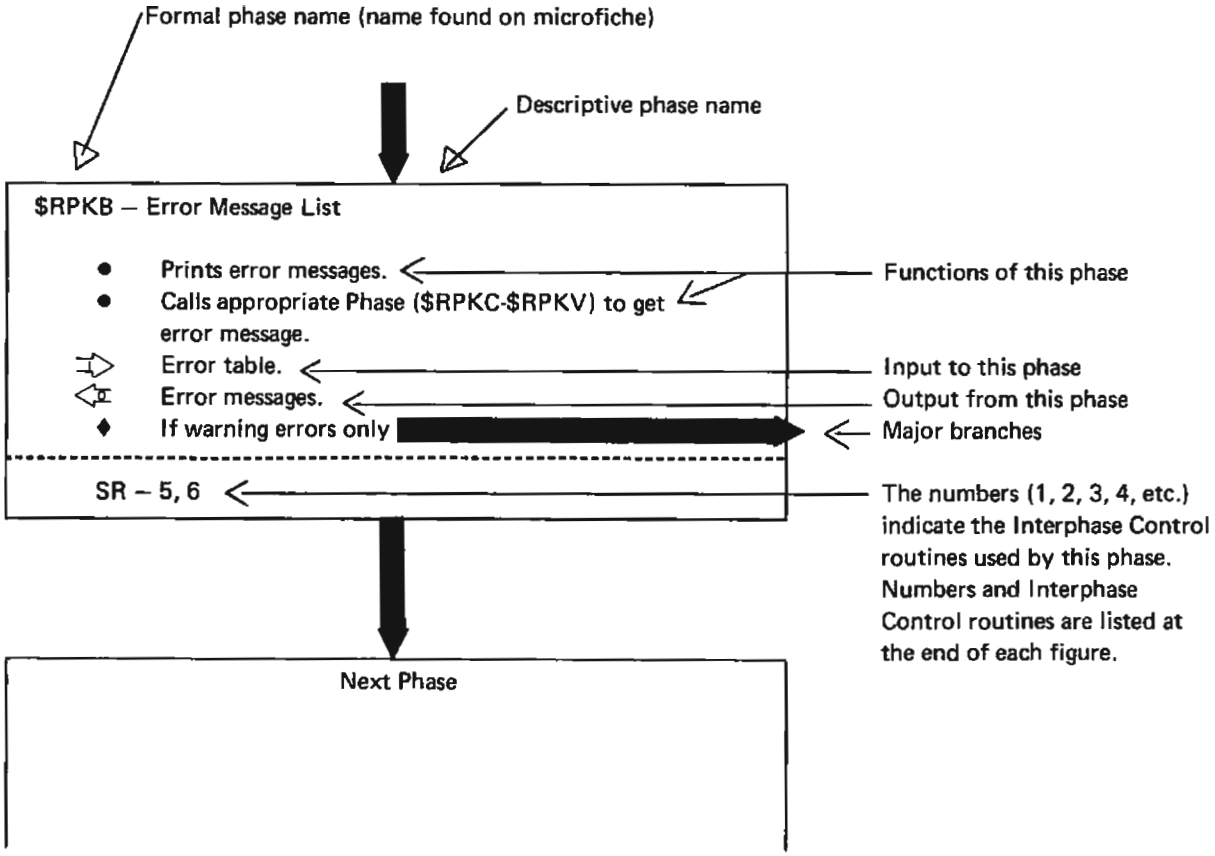
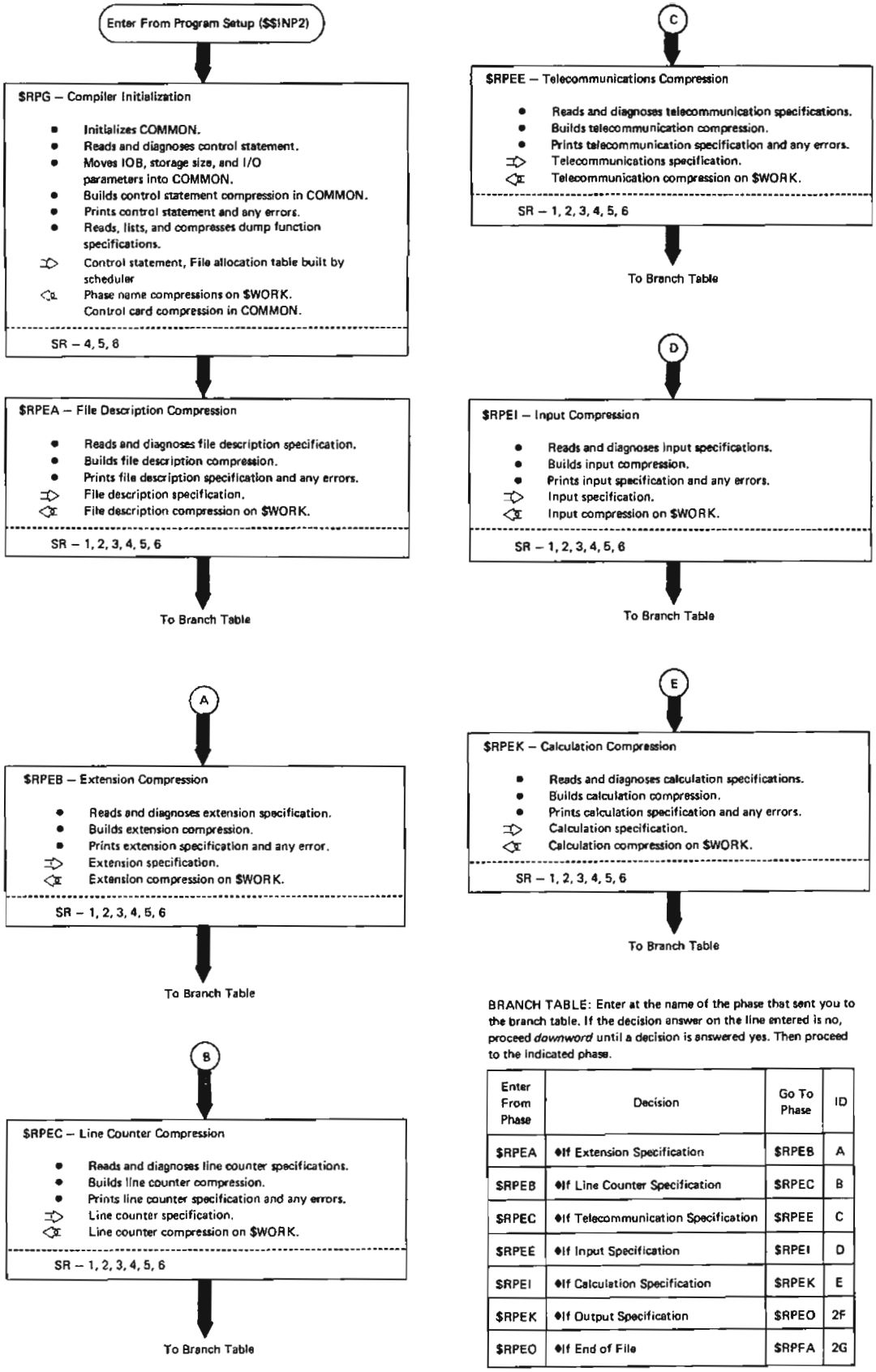


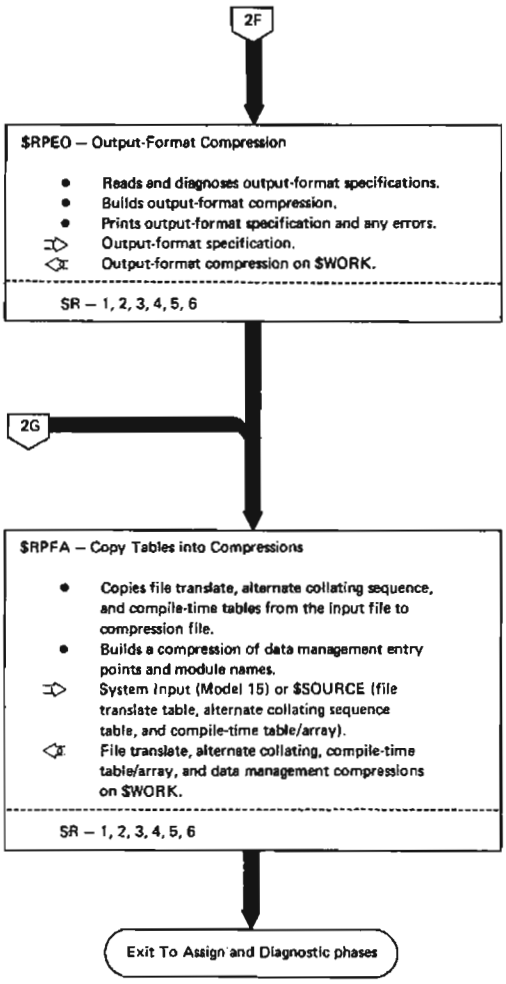
Figure 2-1. Explanation of Program Organization Charts



BRANCH TABLE: Enter at the name of the phase that sent you to the branch table. If the decision answer on the line entered is no, proceed *downword* until a decision is answered yes. Then proceed to the indicated phase.

Enter From Phase	Decision	Go To Phase	ID
$RPEA	•If Extension Specification	$RPEB	A
$RPEB	•If Line Counter Specification	$RPEC	B
$RPEC	•If Telecommunication Specification	$RPEE	C
$RPEE	•If Input Specification	$RPEI	D
$RPEI	•If Calculation Specification	$RPEK	E
$RPEK	•If Output Specification	$RPEO	2F
$RPEO	•If End of File	$RPFA	2G

Figure 2-2 (Part 1 of 2). Input and Compression Phases



Note: Interphase Control routines that can be used by the Input and Compression Phases are:

SR1 – DRGCZA – Open a Compression Area
SR2 – DRGCZC – Write a Compression
SR3 – DRGCZE – Close a Compression
SR4 – DRGCZN – Get Next Source Record
SR5 – DRGCZP – Printer Control
SR6 – DRGCZZ – Call Next Compiler Phase
SR7 – PIOCS – Disk Control

See Figure 2-9 for a description of the Interphase Control routines.

Note: Since fields in COMMON are input and/or output for all phases, COMMON is not listed as input or output.

Figure 2-2 (Part 2 of 2). Input and Compressions Phases

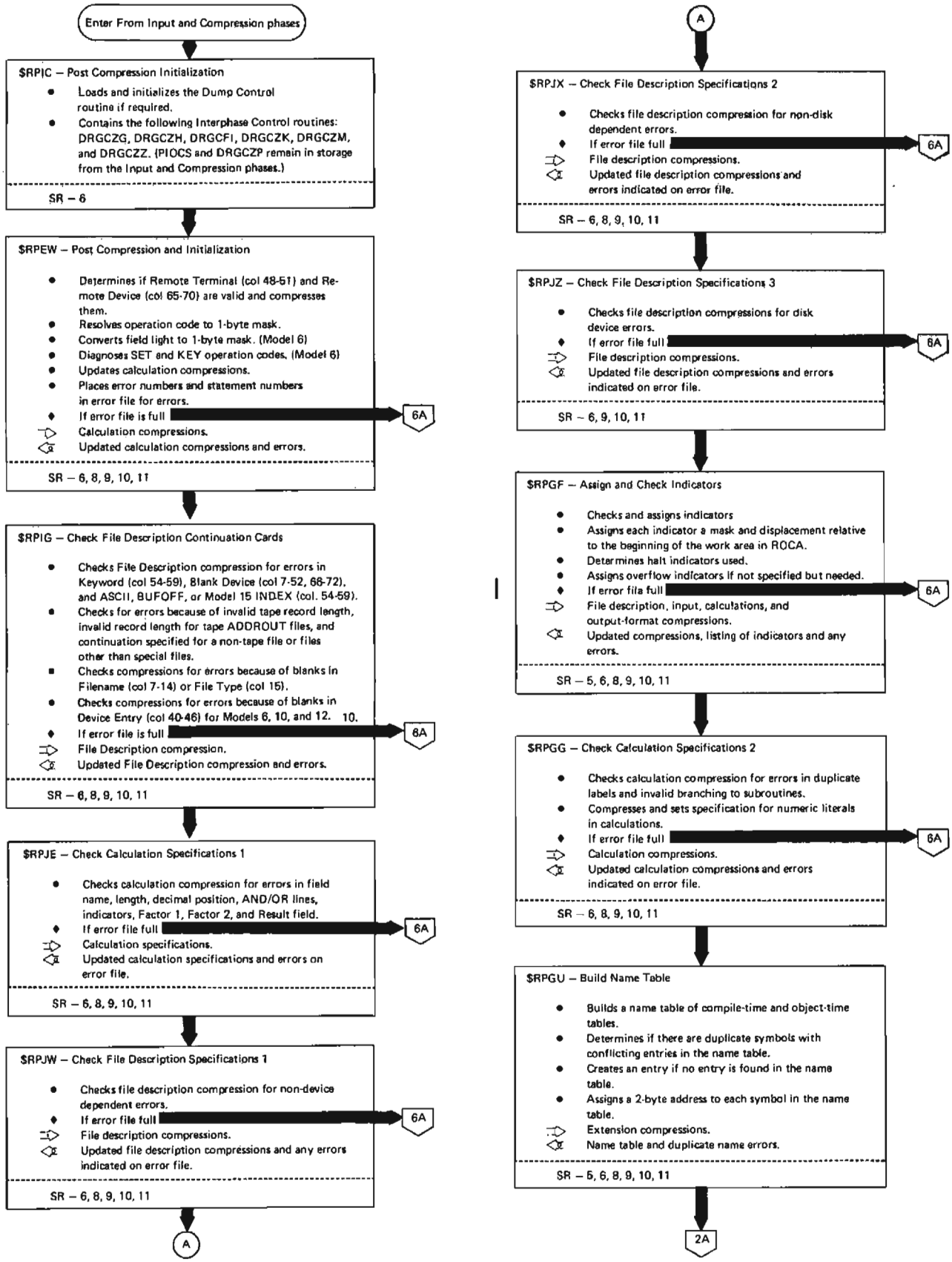


Figure 2-3 (Part 1 of 6). Assign and Diagnostic Phases

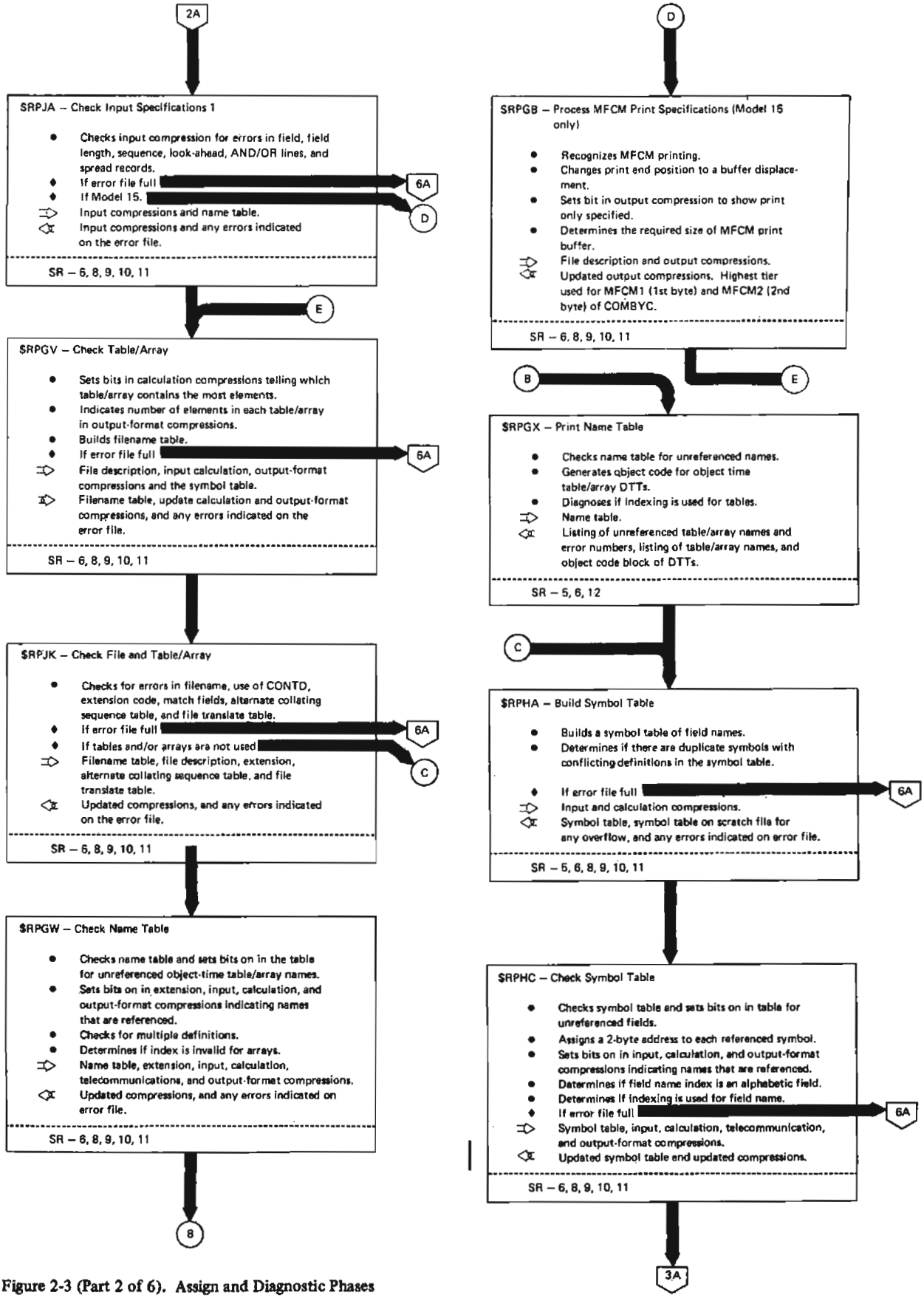


Figure 2-3 (Part 2 of 6). Assign and Diagnostic Phases

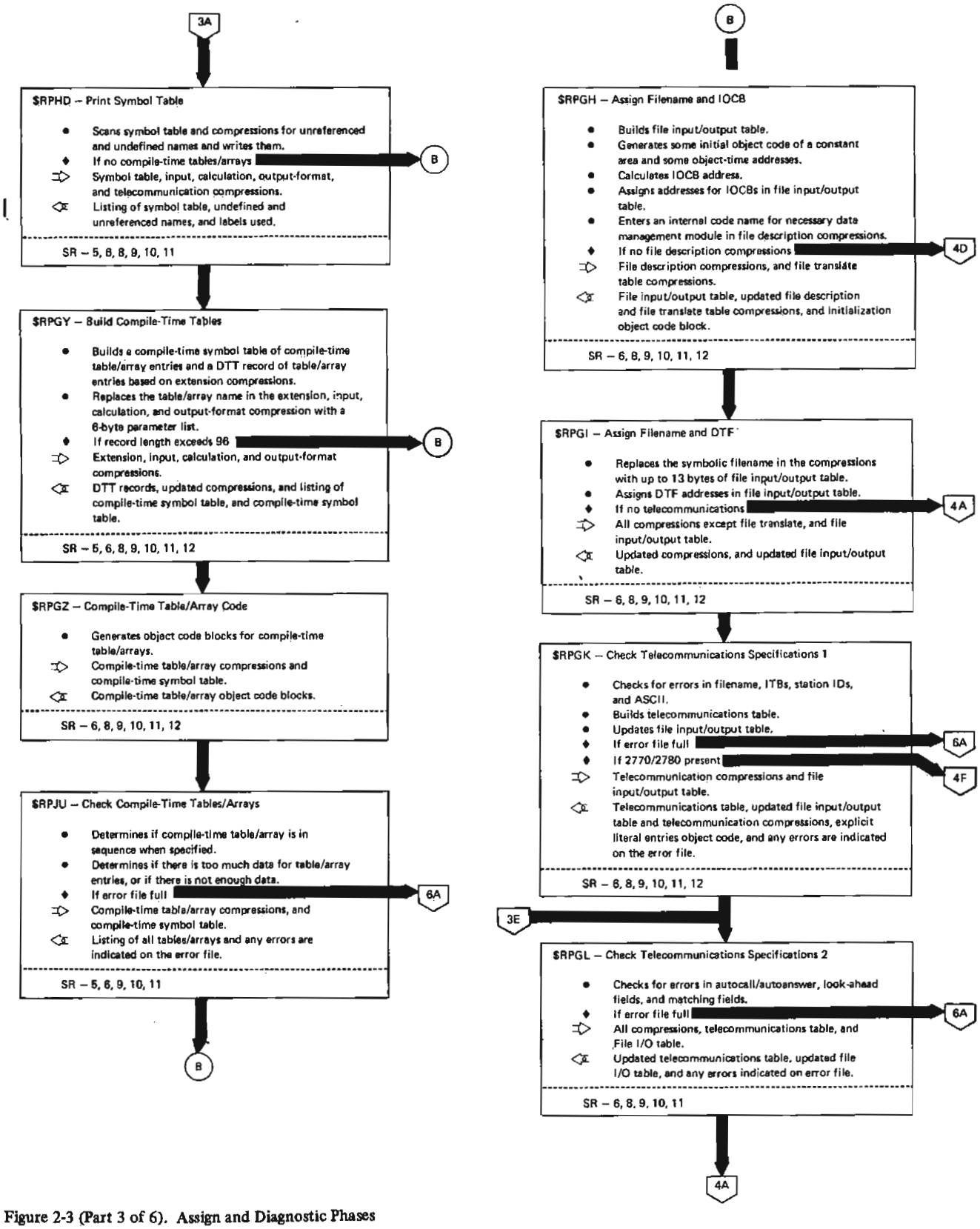


Figure 2-3 (Part 3 of 6). Assign and Diagnostic Phases

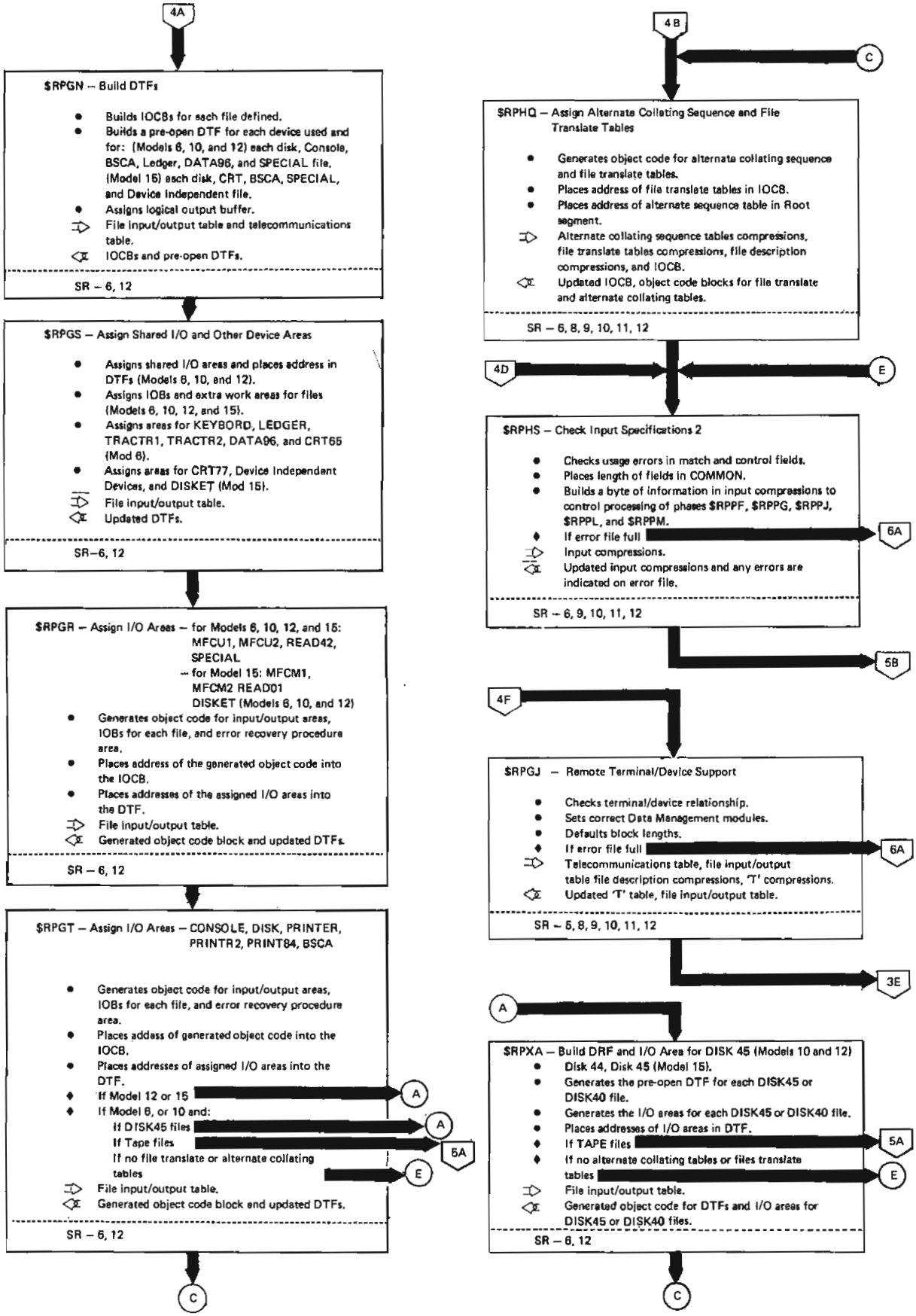


Figure 2-3 (Part 4 of 6). Assign and Diagnostic Phases

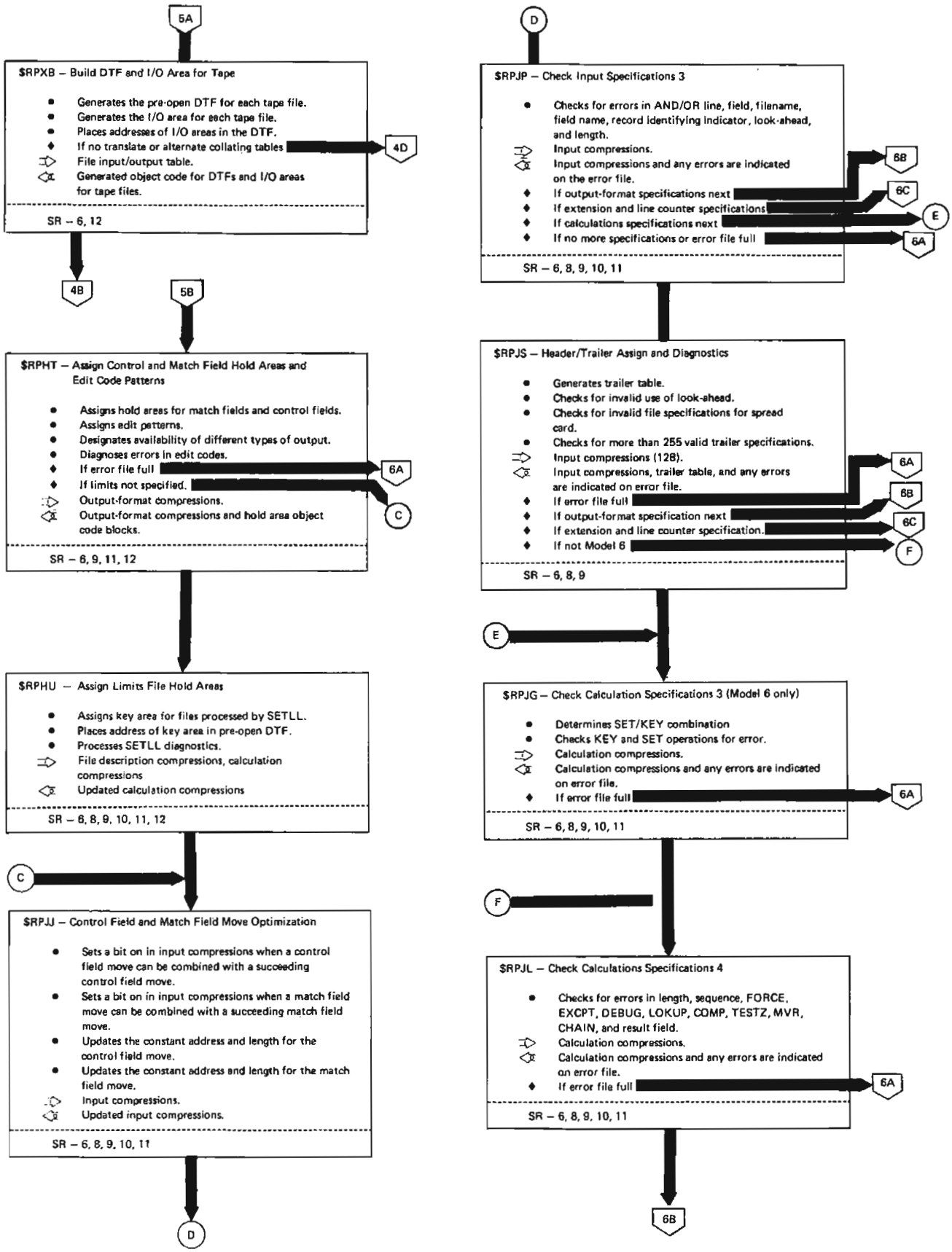
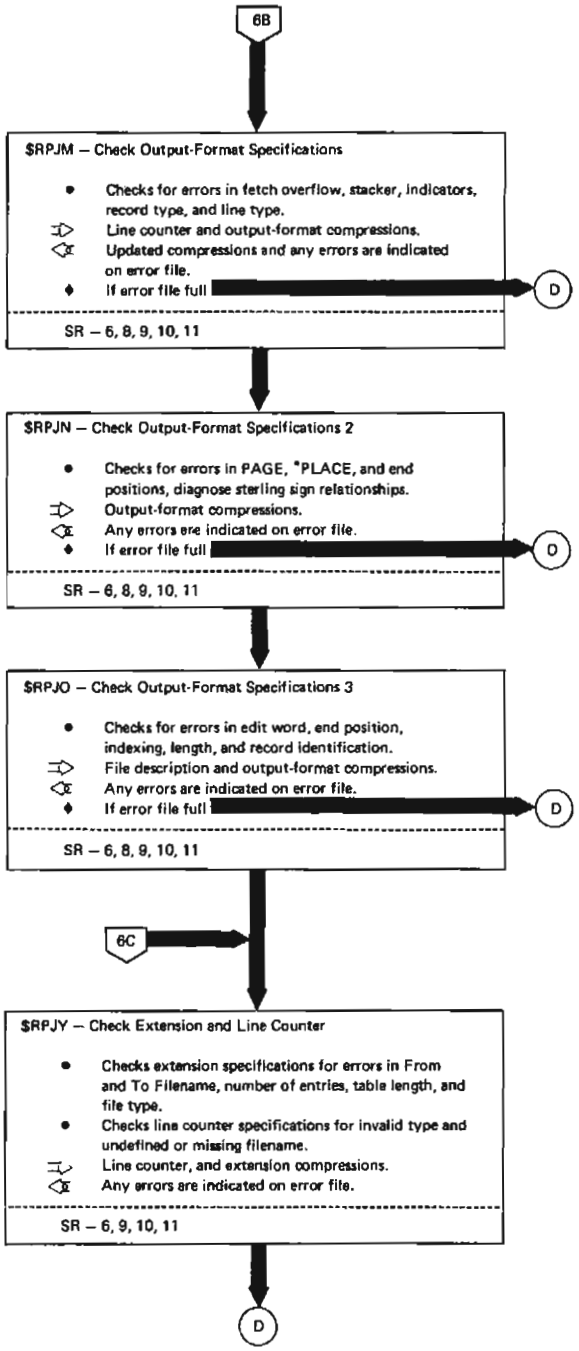
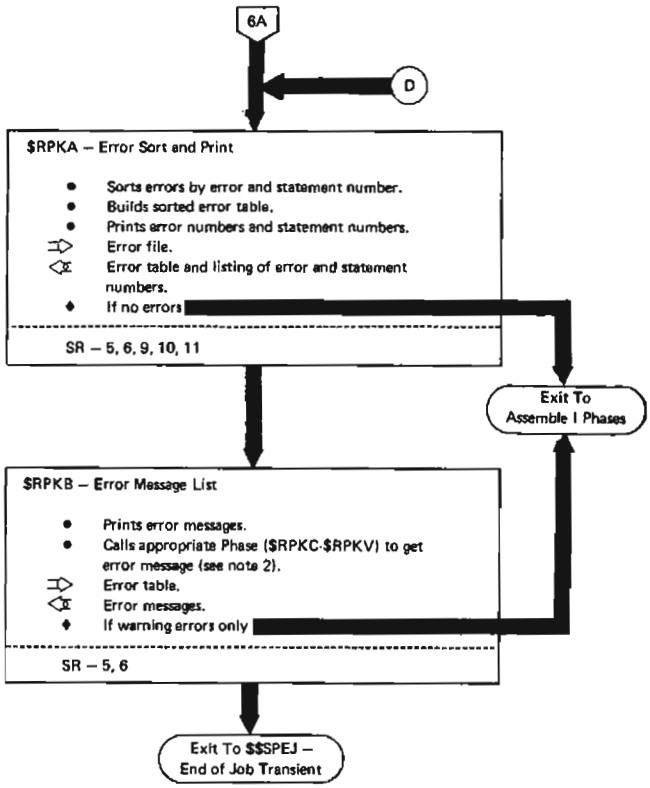


Figure 2-3 (Part 5 of 6). Assign and Diagnostic Phases





Note 1: Interphase Control routines that can be used during the Assign and Diagnostic Phases are:

SR5 – DRGCZP – Printer Control
SR6 – DRGCZZ – Call Next Compiler Phase
SR7 – PIOCS – Disk Control
SR8 – DRGCFI – Find Item in Compression
SR9 – DRGCZG – Open a Compression Block
SR10 – DRGCZH – Get Next Compression
SR11 – DRGCZK – Close a Compression Block
SR12 – DRGCZM – Write an Object Code Block

See Figure 2-9 for a description of the Interphase Control routines.

Note 2: Phases $RPKC-$RPKV contain the error messages used by phase $RPKB. These phases are optional. The phases and message numbers of the messages each phase contains are:

Phase	*Entry Point*	*Message Numbers*
RPKC	DRGKCI	32-63
RPKD	DRGKDI	64-90
RPKE	DRGKEI	91-116
RPKF	DRGKFI	117-142
RPKG	DRGKGI	143-169
RPKH	DRGKHI	170-195
RPKI	DRGKII	196-228
RPKJ	DRGKJI	229-250
RPKK	DRGKKI	251-276
RPKL	DRGKLI	277-310
RPKM	DRGKMI	311-340
RPKN	DRGKNI	341-360
RPKO	DRGKOI	361-379
RPKP	DRGKPI	380-453
RPKQ	DRGKQI	454-541
RPKR	DRGKRI	542-574
RPKS	DRGKSI	575-597
RPKT	DRGKTI	598-630
RPKU	DRGKUI	631-800

Note: Since fields in COMMON are input and/or output for all phases, COMMON is not listed as input or output.

Figure 2-3 (Part 6 of 6). Assign and Diagnostic Phases

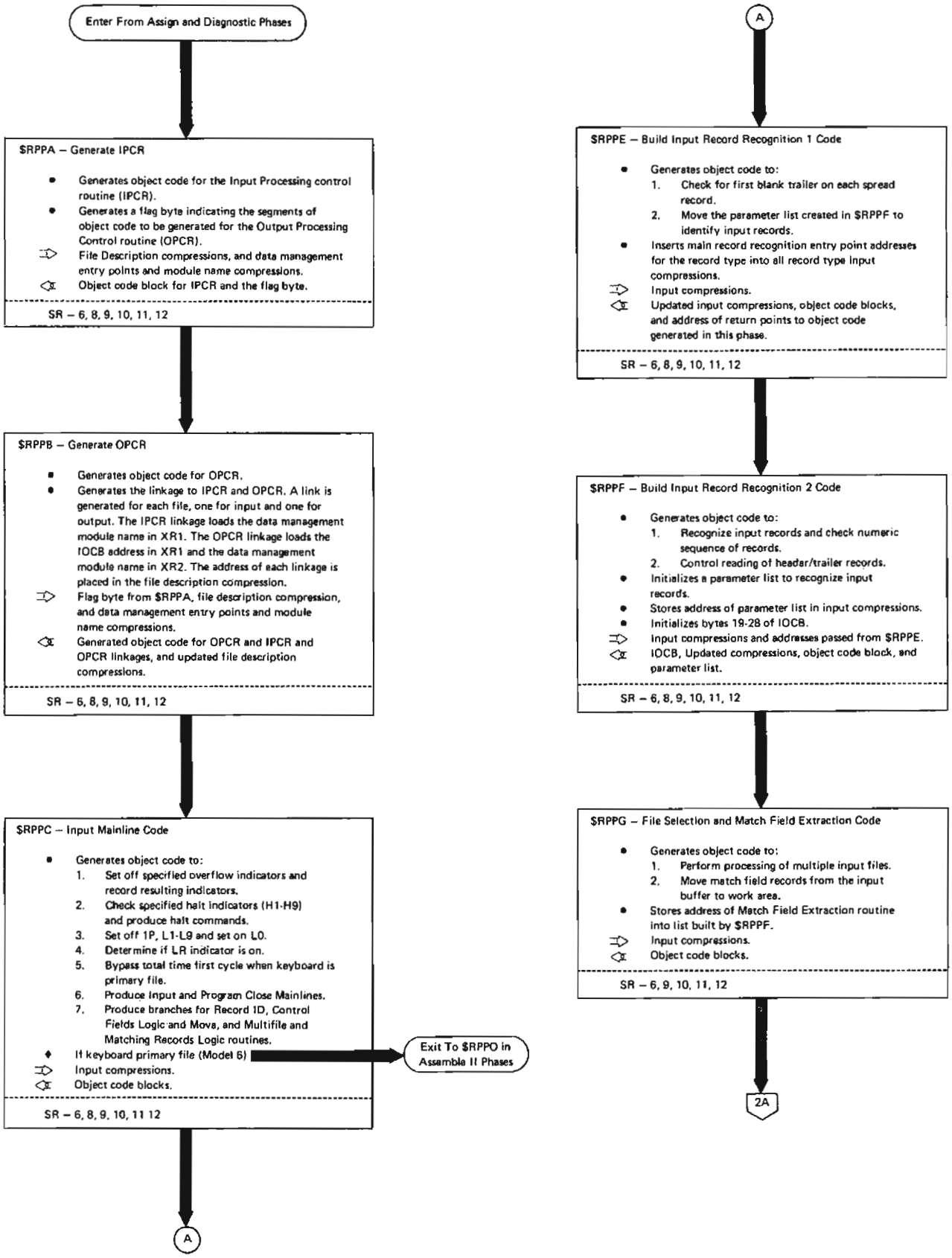
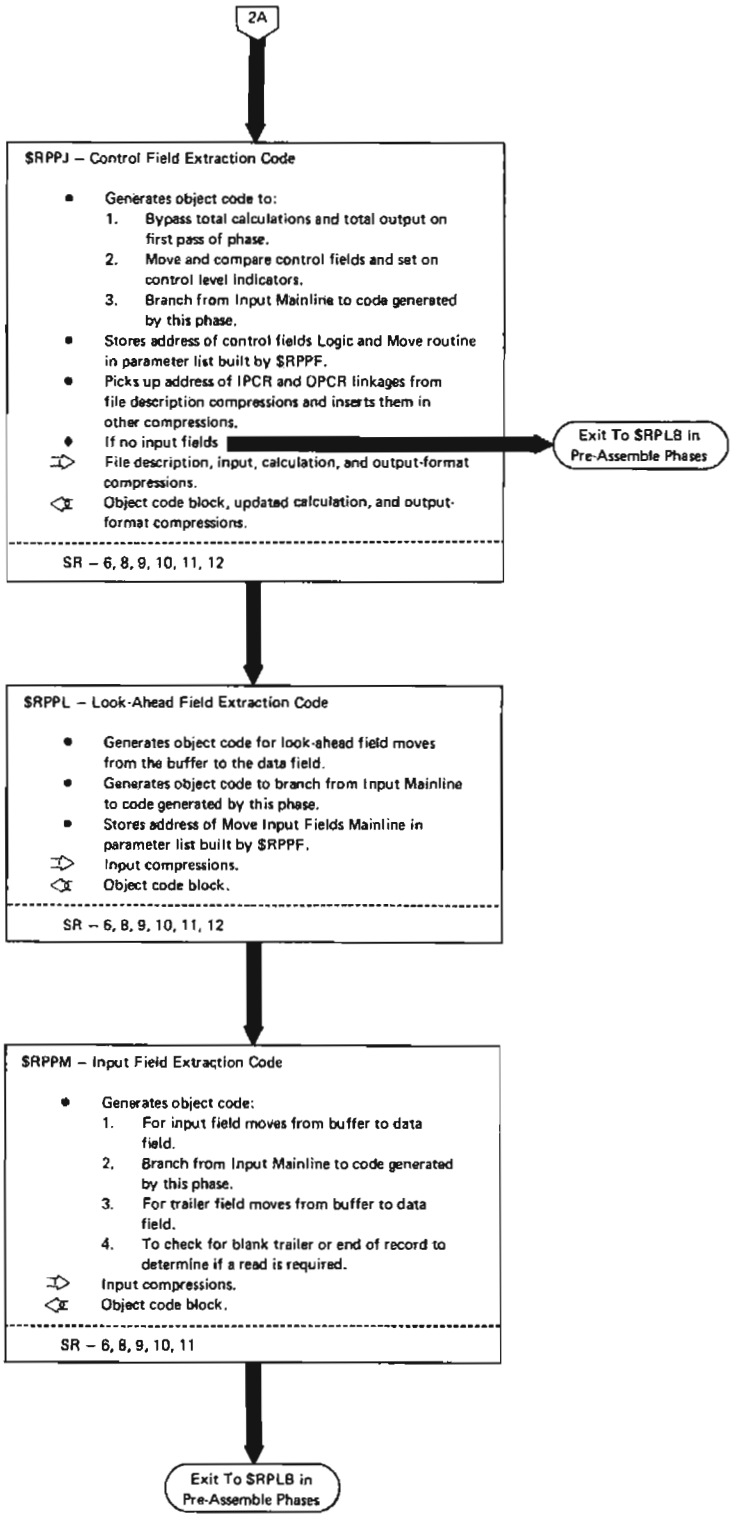


Figure 2-4. Assemble I Phases (Part 1 of 2)



Note: Interphase Control routines that can be used during the Assemble I Phases are:

SR5 – DRGCZP – Printer Control
SR6 – DRGCZZ – Call Next Compiler Phase
SR7 – PIOCS – Disk Control
SR8 – DRGCFI – Find Item in Compression
SR9 – DRGCZG – Open a Compression Block
SR10 – DRGCZH – Get Next Compression
SR11 – DRGCZK – Close a Compression Block
SR12 – DRGCZM – Write an Object Code Block

See Figure 2-9 for a description of the Interphase Control routines.

Note: Since fields in COMMON are input and/or output for all phases, COMMON is not listed as input or output.

Figure 2-4. Assemble I Phases (Part 2 of 2)

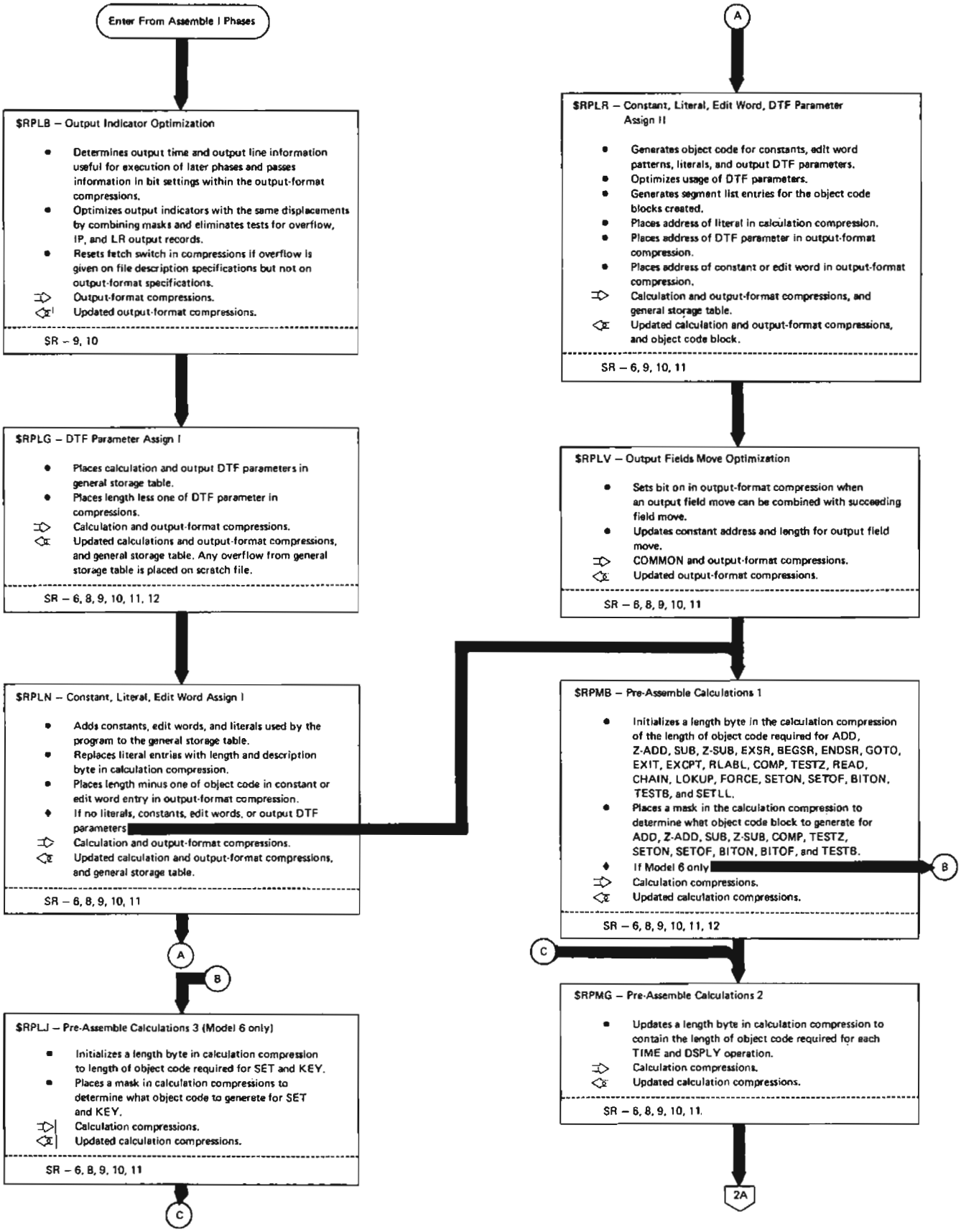
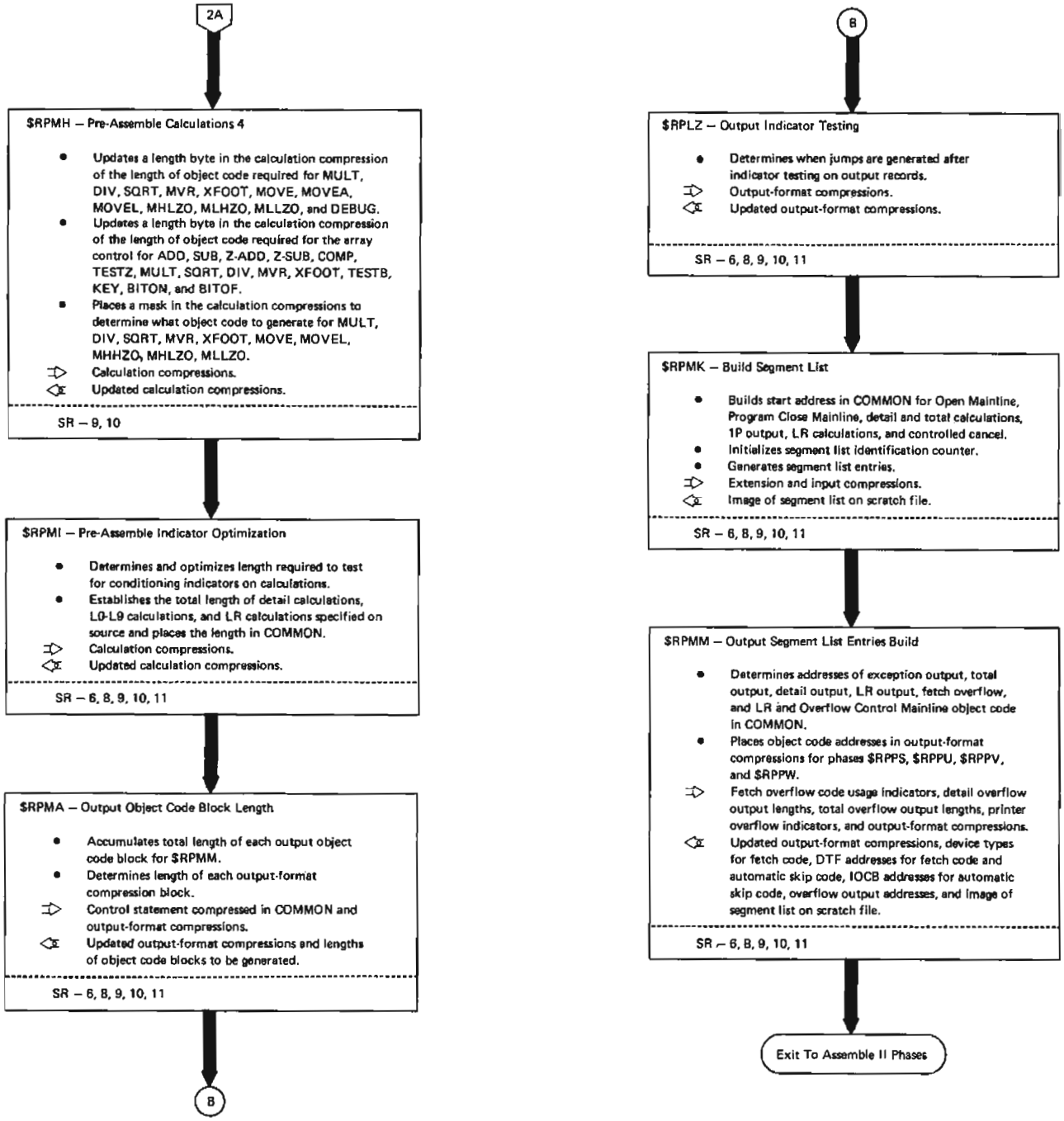


Figure 2-5 (Part 1 of 2). Pre-Assemble Phases



Note: Interphase Control routines that can be used during Pre-Assemble Phases are:

SR5 – DRGCZP – Printer Control
SR6 – DRGCZZ – Call Next Compiler Phase
SR7 – PIOCS – Disk Control
SR8 – DRGCFI – Find Item in Compression
SR9 – DRGCZG – Open a Compression Block
SR10 – DRGCZH – Get Next Compression
SR11 – DRGCZK – Close a Compression Block
SR12 – DRGCZM – Write an Object Code Block

Note: Since fields in COMMON are input and/or output for all phases, COMMON is not listed as input or output.

Figure 2-5. Pre-Assemble Phases (Part 2 of 2)

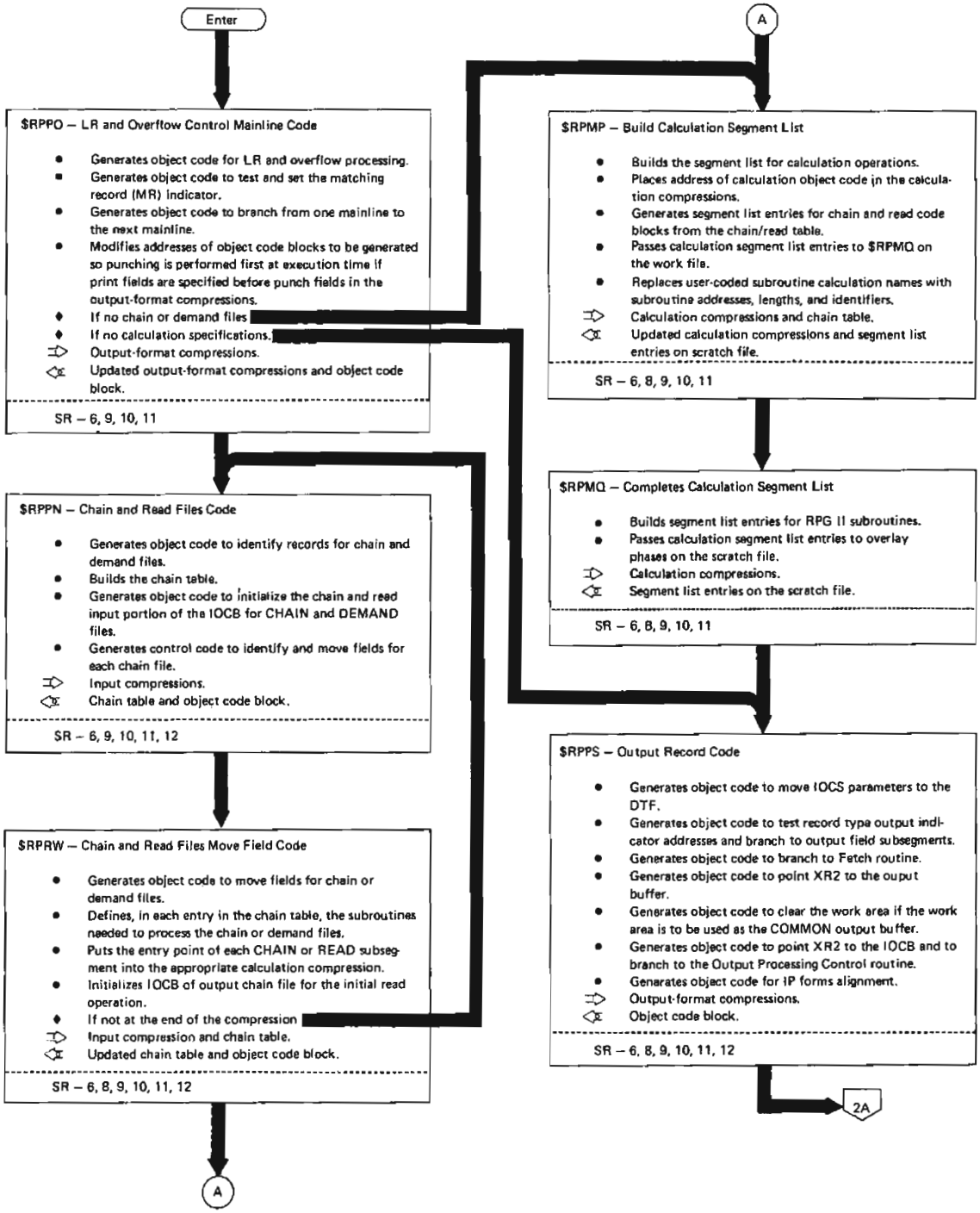


Figure 2-6. Assemble II Phases (Part 1 of 4)

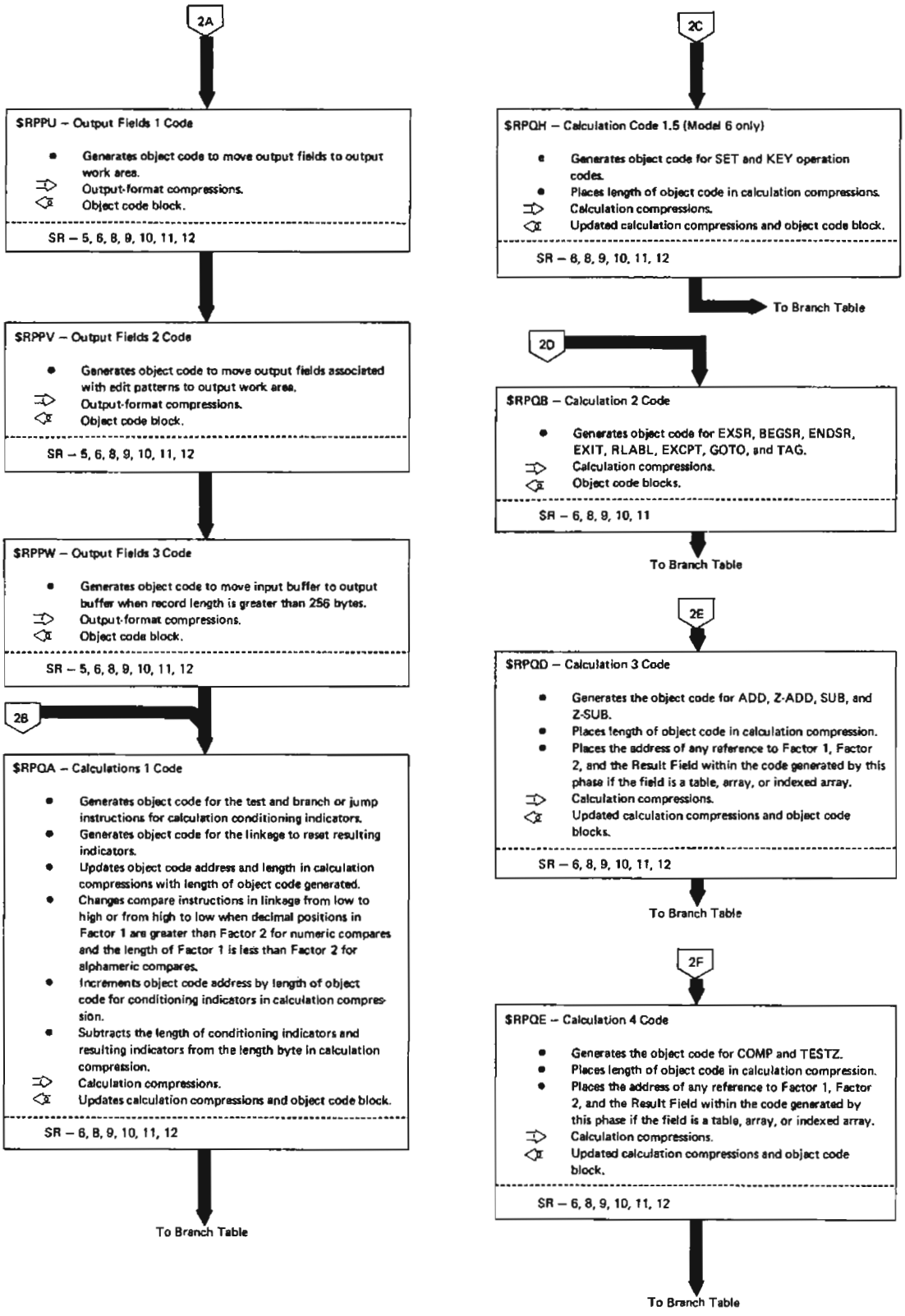


Figure 2-6. Assemble II Phases (Part 2 of 4)

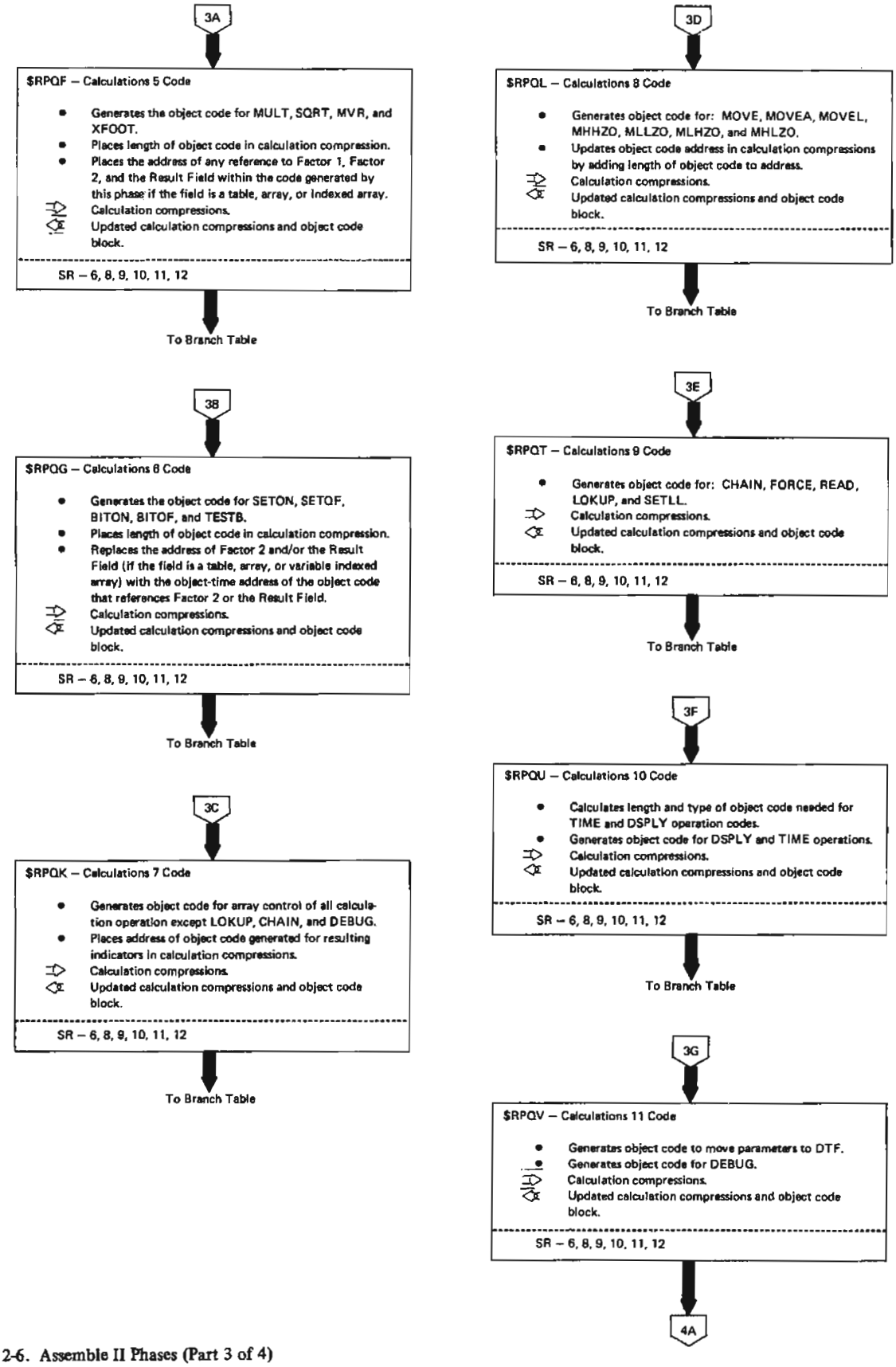
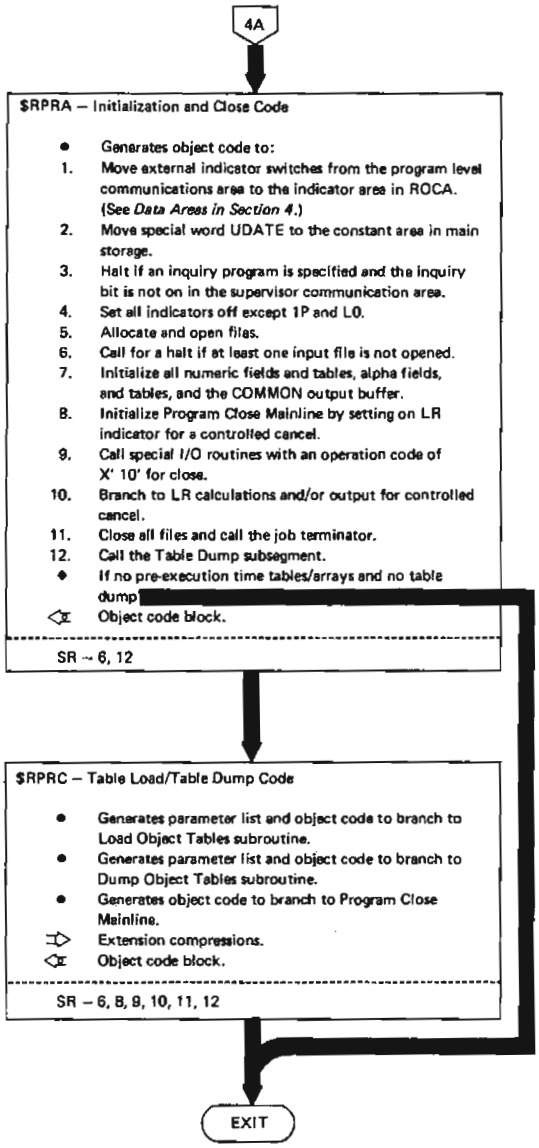


Figure 2-6. Assemble II Phases (Part 3 of 4)



Branch Table: Enter at the name of the phase that sent you to the branch table. If the decision answer on the line entered is no, proceed *downward* until a decision is answered yes. Then proceed to the indicated phase.

From Phase:	Decision	Go To Phase:	ID
$RPQA	♦ If using Model 6	$RPQH	2C
$RPQH	♦ If EXSR, BEGSR, ENDSR, EXIT, RLABL, EXCPT, GOTO, or TAG operation code	$RPQB	2D
$RPQB	♦ If ADD, Z-ADD, SUB, Z-SUB operation code	$RPQD	2E
$RPQD	♦ If COMP or TESTZ operation code	$RPQE	2F
$RPQE	♦ If MULT, SQRT, MVR, or XFOOT operation code	$RPQF	3A
$RPQF	♦ If SETON, SETOF, BITON, BITOF, or TEST8 operation code	$RPQG	3B
$RPQG	♦ If LOKUP, CHAIN, or DEBUG operation code	$RPQK	3C
$RPQK	♦ If MOVE, MOVEA, MOVEL, MHHZO, MLLZO, MLHZO, or MHLZO operation code.	$RPQL	3D
$RPQL	♦ If CHAIN, FORCE, READ, LOKUP, or SETLL operation code	$RPQT	3E
$RPQT	♦ If DSPLY operation code	$RPQU	3F
$RPQU	♦ If DEBUG operation code	$RPQV	3G

Note: Interphase Control routines that can be used during Assemble II phases are:

SR5. DRGCZP – Printer Control
SR6. DRGCZZ – Call Next Compiler Phase
SR7. PIOCS – Disk Control
SR8. DRGCFI – Find Item in Compression
SR9. DRGCZG – Open a Compression Block
SR10. DRGCZH – Get Next Compression
SR11. DRGCZK – Close a Compression Block
SR12. DRGCZM – Write an Object Code Block

See Figure 2-9 for a description of the Interphase Control routines.

Note: Since fields in COMMON are input and/or output for all phases, COMMON is not listed as input or output.

Figure 2-6. Assemble II Phases (Part 4 of 4)

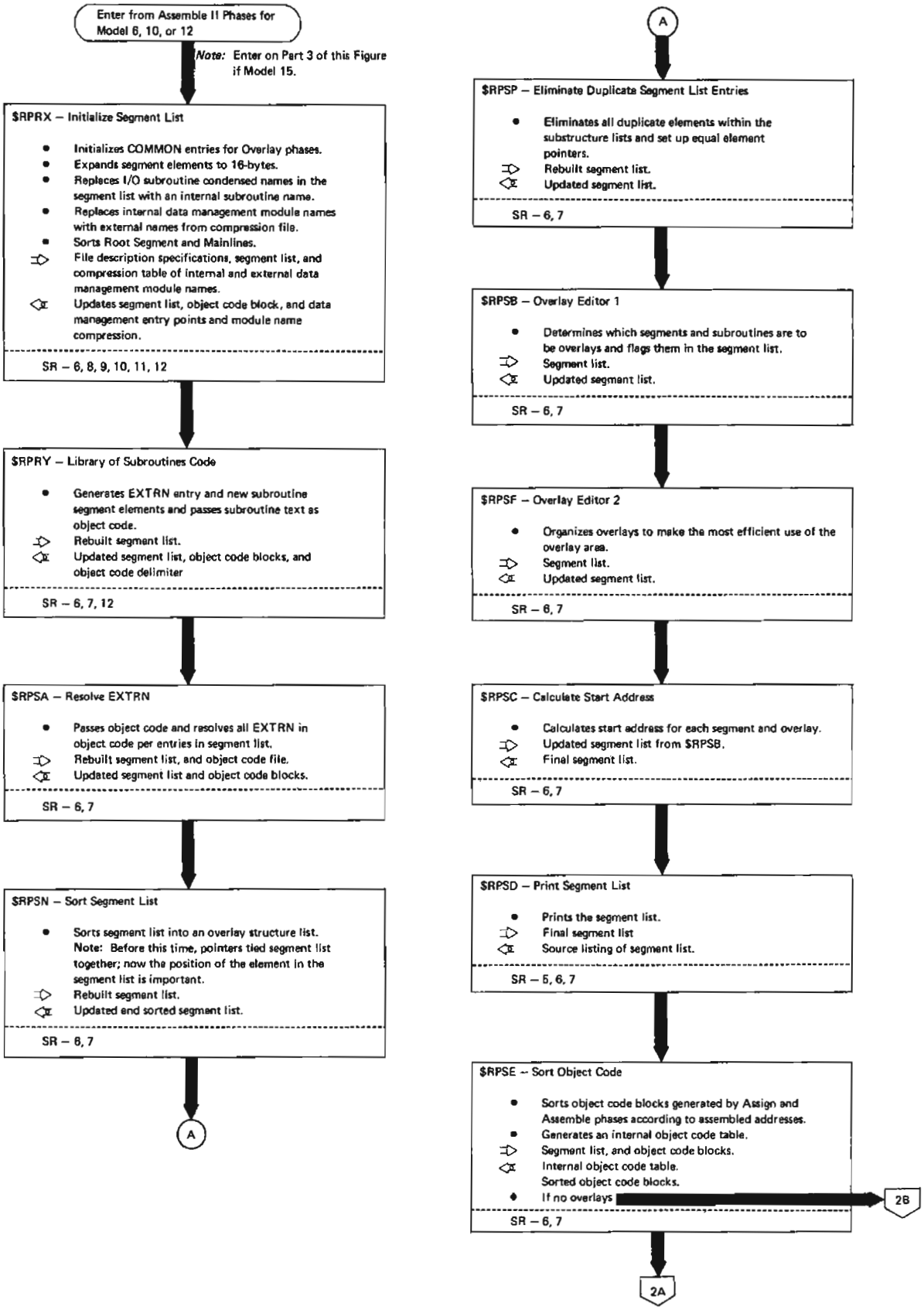


Figure 2-7 (Part 1 of 3). Overlay Phases

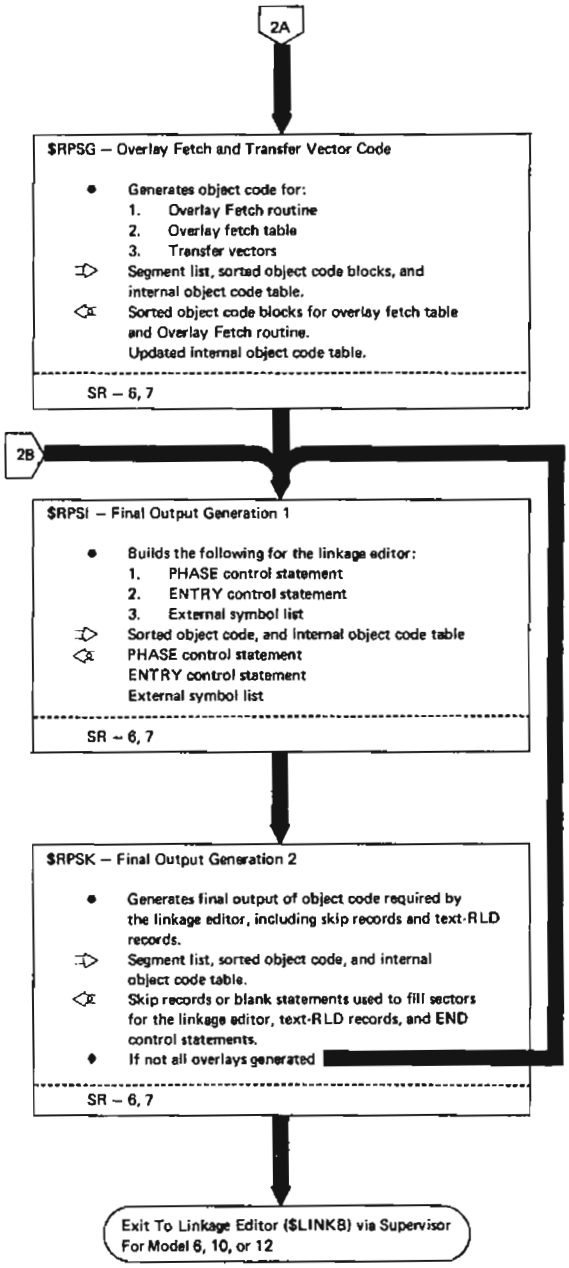


Figure 2-7 (Part 2 of 3). Overlay Phases

Note: Interphase Control routines that can be used by the Overlay phases are:

SR5 – DRGCZP – Printer control
SR6 – DRGCZZ – Call next compiler phase
SR7 – PIOCS – Disk Control
SR8 – DRGCFI – Find Item in Compression
SR9 – DRGCZG – Open a Compression Block
SR10 – DRGCZH – Get Next Compression
SR11 – DRGCZK – Close a Compression Block
SR12 – DRGCZM – Write an Object Code Block

See Figure 2-9 for a description of the Interphase Control routines.

Note: Since fields in COMMON are input and/or output for all phases, COMMON is not listed as input or output.

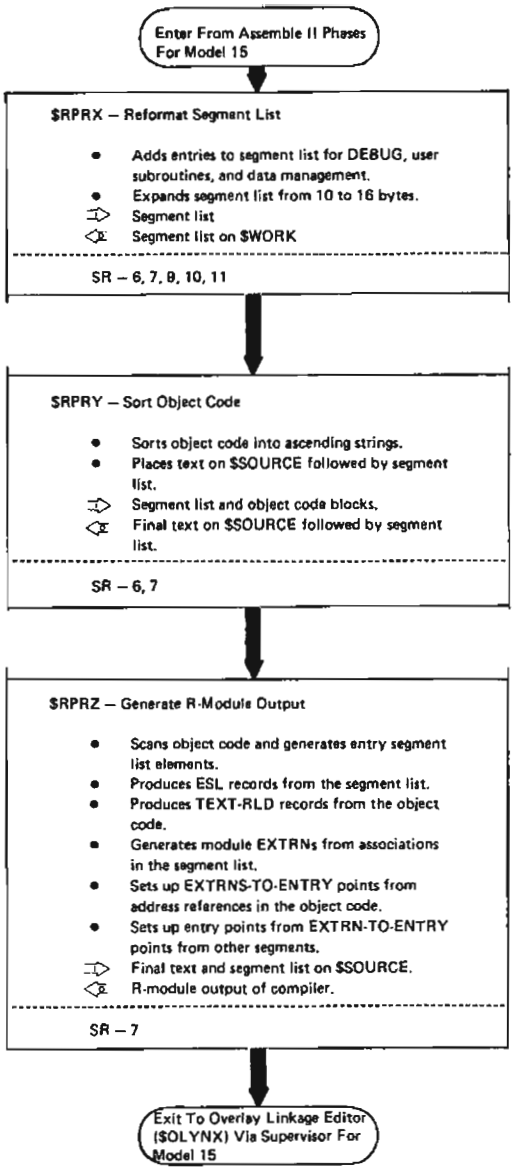


Figure 2-7 (Part 3 of 3). Overlay Phases

INTERPHASE CONTROL ROUTINES DESCRIPTION

Figure 2-8 shows the charting techniques used on Figure 2-9 to describe the Interphase Control Routines.

The Interphase Control routines reside in the compiler control region defined in phase $RPG and phase $RPIC. These routines control all disk input and output.

During the Input and Compression phases the following Interphase Control routines are present:

Open a Compression Area (DRGCZA)
Write a Compression (DRGCZC)
Close a Compression Area (DRGCZE)
Get Next Source Record (DRGCZN)
Printer Control (DRGCZP)
Call Next Compiler Phase (DRGCZZ)
Disk Control (PIOCS)

When phase $RPIC overlays the last Input and Compression phase, the following Interphase Control routines are loaded into the compiler control region, overlaying all previous Interphase Control routines except PIOCS and DRGCZP:

Find Item in Compression (DRGCFI)
Open a Compression Block (DRGCZG)
Get Next Compression (DRGCZH)
Close a Compression Block (DRGCZK)
Write an Object Code Block (DRGCZM)
Call Next Compiler Phase (DRGCZZ)

Some Interphase Control routines function in a special way if less than the required work area is available for the normal Interphase Control routines.

If only 1K or less is available for the work area and the source input, the following Interphase Control routines use other than normal routines.

DRGCZC – Basic function is same as normal, but allows only one compression to be built and then puts that compression out to disk. Shares the 512 bytes of work area with DRGCZN.

DRGCZE – Same function as normal, but works with special DRGCZC.

DRGCZN – Same function as normal, but source shares work area with compressions. One source record is retrieved from disk and located in main storage as required.

If only 512 or 718 bytes are available for the work area and object code, the following Interphase Control routines use other than the normal routines.

DRGCZG – Basic function is same as normal, but only allows one compression type in storage at one time. Requires 512 bytes of available work area.

DRGCZH – Basic function same as normal, but only allows one compression type in core at one time. Requires two sectors of available work area.

DRGCZK – Same function as normal, but works with special DRGCZH. Always writes the updated compressions.

DRGCZM – Basic function is same as normal, but uses 256-byte buffer area in the control region of storage instead of normal object code area.

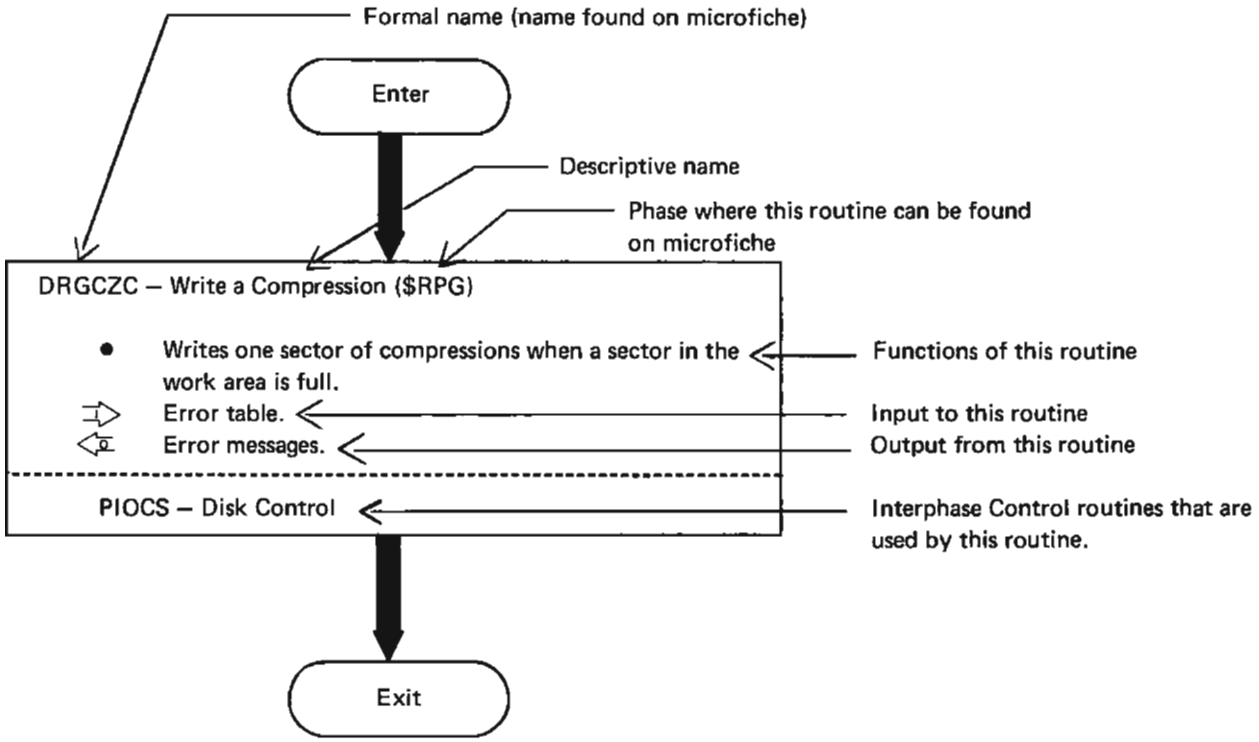


Figure 2-8. Description of the Format Used in Describing the Interphase Control Routines

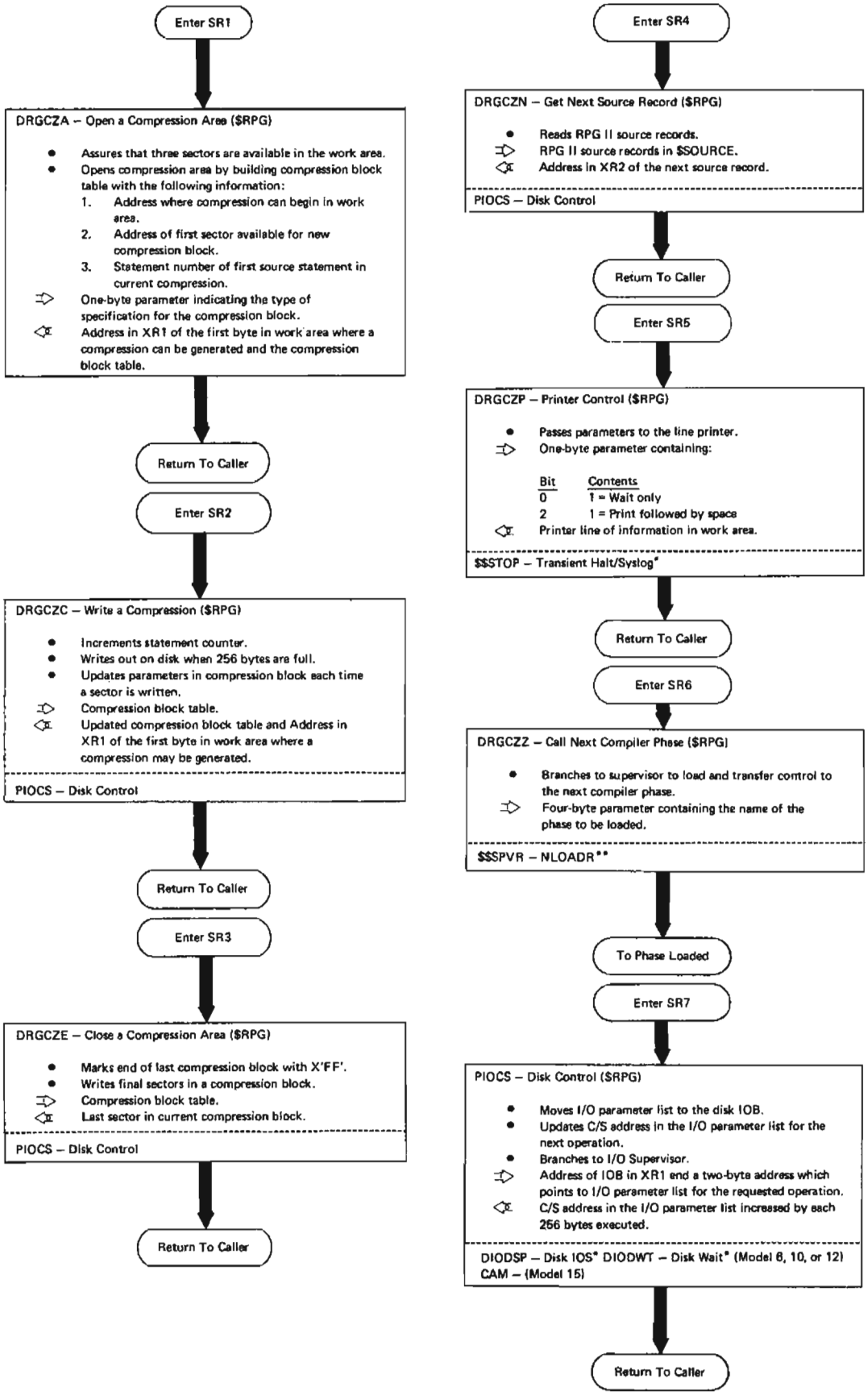


Figure 2-9 (Part 1 of 2). Interphase Control Routines

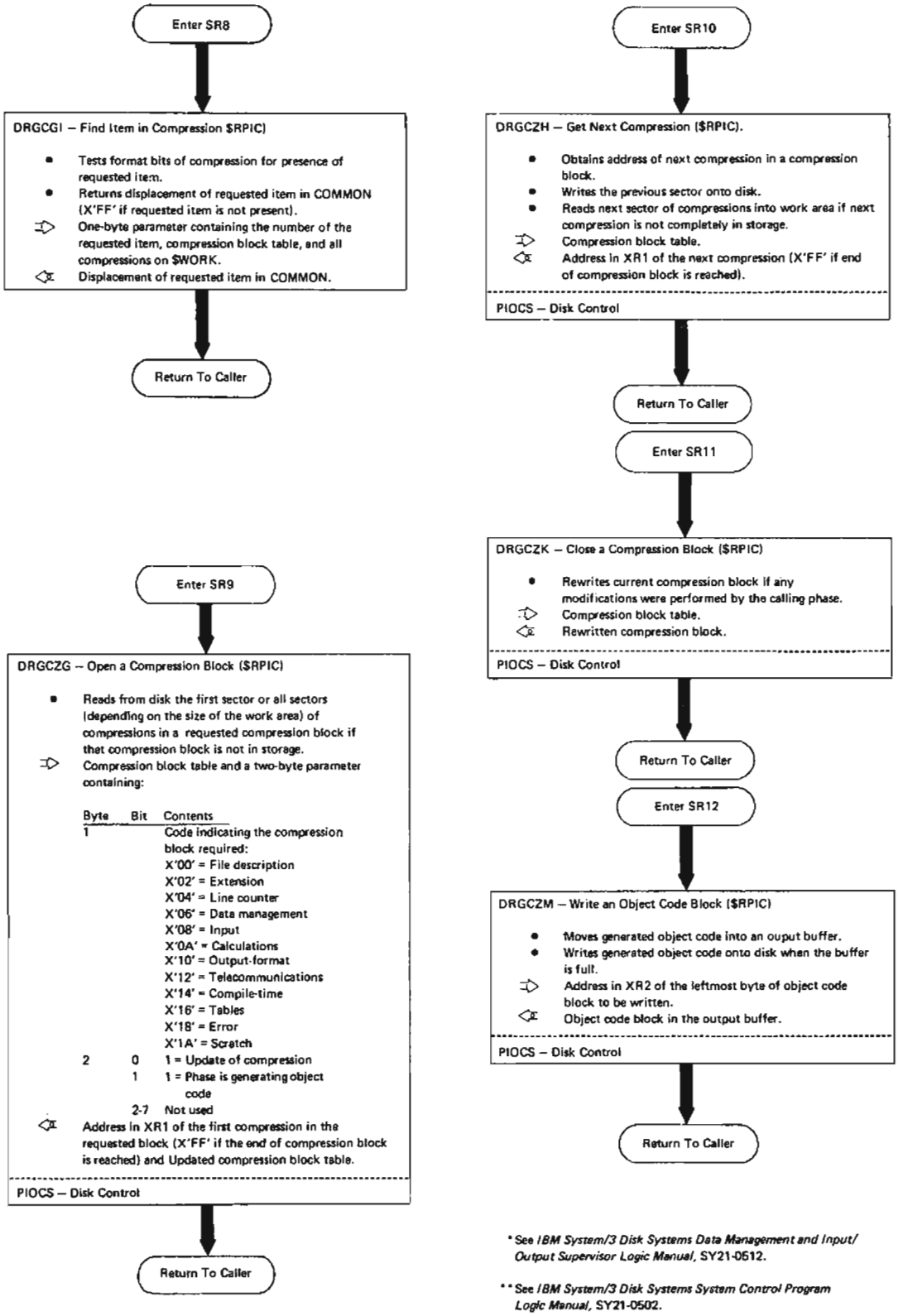


Figure 2-9. Interphase Control Routines (Part 2 of 2)

This section describes the format and contents of the control blocks and data areas created by the RPG II Compiler and used by more than one phase of the compiler.

COMPILER CONTROL REGION

For Models 6 and 10, the compiler control region is a 1.5K byte area in main storage following the end of the Supervisor. For Model 12, the control region is a 2.0K byte area in main storage at the starting address of the partition in which the compiler is loaded. For Model 15, it is a 2.5K byte area in main storage at the beginning of the partition in which the compiler is loaded. The compiler control region contains:

1. COMMON
2. Interphase Control routines
3. Data files, tables, buffers, IOCBs, control routine save area

For a description of the Interphase Control routines, see *Interphase Control Routines Description* in *Section 2. Program Logic.*

COMMON

COMMON is a 128-byte interphase area used to save information used by more than one compiler phase. Each phase can access information placed there by preceding phases and can transfer information to following phases. COMMON is located X'0A' from the end of the Supervisor. Figure 3-1 shows the information found in COMMON, where this information is located, and the phases where this information is defined and modified.

COMMON

Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanation
COMAAC	00	0	$RPG	$RPGF $RPKA $RPKE	Control card specification information: Bit 0 1 = Object program permanently in library (C in col 10) Bit 1 1 = Object program punched on cards (P in col 10) Bit 2 1 = No object program produced (col 11) Bit 3 1 = No list Bit 4 1 = Shared I/O (Models 6, 10, and 12 only) Bit 5 1 = DEBUG operation (col 15) Bit 6-7 00 = Domestic format (col 21) 01 = United Kingdom format (col 21) 10 = World Trade date format (I) col 21 11 = World Trade date format (J) col 21
COMABC	01	1	$RPG	None	Control statement information; object program size (col 12-14)
COMACC	02	2	$RPG	$RPGH $BPGI $RPGJ $RPGN $RPGS $RPGR $RPGT $RPXA $RPXB	Bit 7 1 = Program exceeds 64K bytes of storage or disk overflow error
COMADC	03	3	$RPG	$RPHC $RPJM	Control statement information: Bit 0 1 = No halt for unprintable characters (col 45) Bit 1 1 = Program allows inquiry interrupts (col 37) Bit 2 1 = Inquiry program (col 37) Bit 3 1 = Alternate collating sequence (col 26) Bit 4 1 = Repeat 1P lines (col 41) Bit 5 1 = File translation (col 43) Bit 6 1 = MFCU or DATA96 zero suppression (col 44) Bit 7 1 = UDATE field

Figure 3-1 (Part 1 of 14). COMMON

COMMON					
Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanation
COMAEC	09	4-9	$RPG	None	Program identification specified on the control statement (col 75-80)
COMAFC	0B	10-11	$RPG	$RPIC $RPJW $RPJZ $RPJA $RPHS $RPJG	Statement number of the compression in storage
COMBIC	10	12-16	$RPG	None	The compression sequence number
COMAGC	0C	12	$RPGF	$RPGU $RPGW $RPGX $RPHA $RPHD $RPHT $RPGU	Error information for assign and diagnostic phases: Bit 0 1 = Unreferenced indicator Bit 1 1 = Undefined indicator Bit 2 1 = Multi-defined field Bit 3 1 = Unreferenced field Bit 4 1 = Undefined field Bit 5 1 = Multi-defined tables/arrays Bit 6 1 = Too many tables/arrays
COMBCC	0D	12-13	$RPPC	None	Branch address of the operand portion in Input Mainline to detail calculations
COMAVC	0F	14-15	$RPPC	$RPPE	Address of the return point in Input Mainline
COMAVC	0F	14-15	$RPPE	None	Address of the Record ID routine
COMCKC	0F	14-15	$RPRA	None	Address of Load Object Tables subroutine
COMAWC	11	16-17	$RPPC	$RPPG	Branch address of the operand portion in Input Mainline to Multifile and Matching Records Logic routine
COMAWC	11	16-17	$RPPG	None	Address of Multifile and Matching Records Logic routine
COMFCC	11	16-17	$RPMM	None	Binary length of exception output
COMBJC	50	17-80	$RPG	$RPEA $RPEB $RPEC $RPEE $RPEI $RPEK $RPEO	Errors found in the Input and Compression phases
COMAXC	13	18-19	$RPPC	$RPPJ	Branch address of the operand portion of the branch in Input Mainline to Control Fields Logic and Move routine
COMAXC	13	18-19	$RPPJ	$RPMK	Address of Control Fields Logic and Move routine

Figure 3-1 (Part 2 of 14). COMMON

COMMON					
Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanation
COMFDC	13	18-19	$RPMK	$RPMM	Compression sequence number of ISAM file that requires record address file processing
COMDUC	14	19-20	$RPSE	$RPSG	Spindle address for file one
COMEDC	15	20-21	$RPPC	$RPPE	Address in Input Mainline to call RPG II Halt Processor
COMAYC	15	20-21	$RPPL	None	Address of Input Fields routine
COMDVC	16	21-22	$RPSE	$RPSG	File one disk start address
COMAUC	17	22-23	$RPPC	None	Address of Input Mainline
COMDWC	18	23-24	$RPSE	None	File one disk end address
COMAZC	19	24-25	$RPPC	None	Branch address of the operand portion in Input Mainline to LR and Overflow Control Mainline
COMCDC	19	24-25	$RPGS	$RPGR $RPGT $RPXA $RPXB	Address of limits I/O save area
COMDXC	1A	25-26	$RPSE	$RPSG	Address of the next available sector
COMCEC	1B	26-27	$RPGS	$RPGT $RPXA	Limits shared I/O DTF address
COMEOC	1B	26-27	$RPPA	$RPPB $RPPJ	BSCA IOCB@ transmit with reply
COMCWC	1B	26-27	$RPMK	$RPRA	Address of controlled cancel in Program Close Mainline
COMDYC	1C	27-28	$RPSB	$RPSF	Byte 0, Bit 0 1 = Object program will not fit in main storage
COMBNC	1C	28	$RPJW	$RPJX $RPJZ	File description information: Bit 0 1 = Ascending sequence Bit 1 1 = Descending sequence Bit 4 1 = Ascending sequence Bit 5 1 = Descending sequence
COMBAC	1C	28	$RPHQ	None	Alternate sequence character to initialize match field save areas

Figure 3-1 (Part 3 of 14). COMMON

COMMON					
Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanation
COMBQC	1D	28-29	$RPHT	None	Address of decimal point for edit pattern 1
COMBOC	1E	29-30	$RPJW	$RPJX $RPJZ	File and indicator information: Byte 0, Bit 0 1 = Overflow indicator used Bit 1 1 = Primary file Bit 2 1 = Record address file Bit 3 1 = Input files conditioned by external indicators Bit 4 1 = Input files Bit 5 1 = Default second primary to secondary Bit 6 1 = Non-demand RA associated file Bit 7 1 = RA associated file Byte 1, Bit 0 1 = External indicator used to condition opening of a file
COMDZC	1E	29-30	$RPSG	None	Address of overlay fetch table
COMBRC	1F	30-31	$RPHT	None	Address of decimal point for edit pattern 2
COMGAC	20	31-32	$RPSC	None	Length of object program
COMBSC	21	32-33	$RPHT	None	Address of left slash of date edit pattern
COMGBC	22	33-34	$RPSC	None	Length of overlay area
COMBTC	23	34-35	$RPHS	None	The control field type. If the bit is set to 0, the control field is alphameric; if the bit is set to 1, the field is numeric. Byte 0, Bit 7 = L9 Byte 1, Bit 0 = L8 Bit 1 = L7 Bit 2 = L6 Bit 3 = L5 Bit 4 = L4 Bit 5 = L3 Bit 6 = L2 Bit 7 = L1
COMFBC	23	34-35	$RPMK	$RPLN $RPLR $RPMM $RPMP $RPRX	The next available segment list number
COMGCC	24	35-36	$RPSC	None	Length of suboverlay area

Figure 3-1 (Part 4 of 14). COMMON

COMMON					
Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanation
COMBUC	25	36-37	$RPHS	None	Two bytes indicating the match field type. If the bit is set to 0, the match field is alphameric; if the bit is set to 1, the field is numeric. Byte 0, Bit 7 = M9 Byte 1, Bit 0 = M8 Bit 1 = M7 Bit 2 = M6 Bit 3 = M5 Bit 4 = M4 Bit 5 = M3 Bit 6 = M2 Bit 7 = M1
COMCRC	25	36-37	$RPMI	$RPMK	Length of detail calculations
COMCRC	25	36-37	$RPMK	None	Address of detail calculations
COMGDC	26	37-38	$RPSC	None	Length of Fetch routine and table
COMBEC	26	38	$RPJA	$RPGH $RPGI $RPHS $RPPC $RPPJ	Field information: Bit 0 1 = Halt indicator H1 used Bit 1 1 = Direct file without ADDROUT Bit 2 1 = Limits file specification Bit 3 1 = BSCA conversational transmit file Bit 4 1 = Alphabetic and numeric fields Bit 5 1 = FORCE operation code Bit 6 1 = Numeric record ID sequence Bit 7 1 = Look ahead field
COMBWC	26	38	$RPPN	$RPPM	Subroutines used by Input Fields: Bit 0 1 = Unpack subroutine Bit 1 1 = Not used Bit 2 1 = Convert to Decimal routine Bit 3 1 = Array Index subroutine Bit 4 1 = Set Resulting Indicators subroutine Bit 5 1 = Not used Bit 6 1 = Chain or demand file in program Bit 7 1 = First time entry switch
COMBFC	27	39	$RPJX	$RPGF	Overflow indicators used: Bit 0 1 = OV Bit 1 1 = OG Bit 2 1 = OF Bit 3 1 = OE Bit 4 1 = OD Bit 5 1 = OC Bit 6 1 = OB Bit 7 1 = OA

Figure 3-1 (Part 5 of 14). COMMON

COMMON					
Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanation
COMBFC	27	39	$RPGF	None	Halt indicators used: Bit 0 1 = H9 Bit 1 1 = H8 Bit 2 1 = H7 Bit 3 1 = H6 Bit 4 1 = H5 Bit 5 1 = H4 Bit 6 1 = H3 Bit 7 1 = H2
COMGEC	28	39-40	$RPSC	None	Length of available object storage
COMBHC	29	40-41	$RPGH	None	Address of the file I/O table
COMASC	2A	40-42	$RPHS	$RPHT	Bytes 0-1 Address of the high-order byte of the first match field save area. Byte 2 Length of the save area.
COMCTC	29	40-41	$RPMI	$RPMK	Length of total calculations
COMCTC	29	40-41	$RPMK	None	Address of total calculations
COMGFC	2A	41-42	$RPSC	None	Address of the overlay area
COMBKC	2B	42-43	$RPGI	None	Address of the FROM file DTF
COMFAC	2B	42-43	$RPPN	None	Address in chain control code where input fields return after moving fields
COMGGC	2C	43-44	$RPSC	$RPSG	Address of suboverlay area
COMATC	33	43-51	$RPPG	None	Length of each match field: Byte 0 = Length of M1 field Byte 1 = Length of M2 field Byte 2 = Length of M3 field Byte 3 = Length of M4 field Byte 4 = Length of M5 field Byte 5 = Length of M6 field Byte 6 = Length of M7 field Byte 7 = Length of M8 field Byte 8 = Length of M9 field
COMBLC	2D	44-45	$RPGI	None	Address of the TO file DTF
COMEUC	2D	44-45	$RPPN	None	Statement number for $RPRW to begin processing
COMGHC	2E	45-46	$RPSB	None	Length of Open-Close overlay

Figure 3-1 (Part 6 of 14). COMMON

COMMON					
Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanation
COMBMC	2F	46-47	$RPJX	$RPGH $RPGI	Byte 0 Not used Byte 1 Bit 0 1 = Card print required Bit 1 1 = BSCA output file Bit 2 1 = BSCA input file Bit 3 1 = BSCA file given Bit 4 1 = MFCU1 print buffer needed (Models 12 and 15) Bit 5 1 = MFCU2 print buffer needed (Models 12 and 15)
COMGIC	30	47-48	$RPSB	$RPSC	Length of Open-Close suboverlay
COMCFC	30	48	$RPPL	None	Temporary save area for COMBWC
COMCQC	31	48-49	$RPMM	None	Address of first fetch code block
COMCGC	32	49-50	$RPPL	None	Address of return to main code for move fields
COMGJC	32	49-50	$RPSB	None	Address of $WORK
COMCIC	33	50-51	$RPPO	None	Address of last record in LR Output routine
COMAQC	36	52-54	$RPHS	$RPHT	Bytes 0-1 Address of the high-order byte of the control field save area. Byte 2 Length of the save area.
COMEYC	37	54-55	$RPLR	$RPQU	Address of DSPLY constant
COMARC	3F	55-63	$RPHS	$RPPJ	Length of each control field: Byte 0 = Length of L1 field Byte 1 = Length of L2 field Byte 2 = Length of L3 field Byte 3 = Length of L4 field Byte 4 = Length of L5 field Byte 5 = Length of L6 field Byte 6 = Length of L7 field Byte 7 = Length of L8 field Byte 8 = Length of L9 field
COMCMC	3B	56-57	$RPMA	$RPMK	Length of 1P Output routine
COMCMC	39	56-57	$RPMK	None	Address of 1P Output routine
COMBDC	3B	58-59	$RPPL	None	Branch address of the operand portion in the Move Input Fields to detail calculations
COMCPC	3D	60-61	$RPMA	$RPMM	Length of Exception Output routine
COMCPC	3D	60-61	$RPMM	None	Address of Exception Output routine

Figure 3-1 (Part 7 of 14). COMMON

COMMON					
Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanation
COMGPC	3E	61-62	$RPSA	$RPRY	Largest item number in segment list
COMGPC	3E	61-62	$RPRY	$RPSA	Byte 1, Bit 0 1 = Entry point not found in segment list
COMBPC	3F	63	$RPGU	$RPGW $RPHA	Table and array information: Bit 0 1 = Symbol table overflowed to disk Bit 1 1 = No field names defined Bits 2-4 Not used Bit 5 1 = Compile-time tables Bit 6 1 = Multi-defined field name heading printed. Bit 7 1 = Multi-defined table/array name heading printed
COMGOC	40	63-64	$RPRY	$RPSN	Address of the delimiter in the segment list
COMAPC	40	64	$RPJX	$RPMM	Overflow indicators specified: Bit 0 1 = OV Bit 1 1 = OG Bit 2 1 = OF Bit 3 1 = OE Bit 4 1 = OD Bit 5 1 = OC Bit 6 1 = OB Bit 7 1 = OA
COMAPC	40	64	$RPMM	None	Overflow segment needed
COMAOC	41	65	$RPGU	$RPGI $RPPC $RPMA	File information: Bit 0 1 = ADDROUT file program specified Bit 1 1 = CHAIN file program specified Bit 2 1 = Dual IOCBs required Bit 3 1 = Table Load subroutine at object time in program Bit 4 1 = Table Dump subroutine required in program Bit 5 0 = Single input file 1 = Multiple input files Bit 6 1 = Match fields are ascending Bit 7 1 = Match fields are descending

Figure 3-1 (Part 8 of 14). COMMON

COMMON					
Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanation
COMANC	43	66-67	$RPGN	$RPGU $RPGY $RPHA $RPLR $RPGH $RPMK $RPGI $RPMM $RPPO $RPGK $RPPN $RPRW $RPGS $RPMP $RPHQ $RPHT $RPPA $RPPC $RPPE $RPPF $RPPG $RPPJ $RPPL $RPSC $RPRZ $RPHP $RPRY $RPJJ $RPPM $RPGR $RPPB $RPHU	Address of next available byte in storage
COMAMC	45	68-69	$RPGY	$RPGU $RPHA $RPHD $RPGH $RPGI $RPGK $RPGS $RPHQ $RPHT $RPJS $RPPA $RPGR $RPHU	Address of end of Root Segment
COMCXC	45	68-69	$RPMK	$RPHA	Address of normal close entry in Program Close Mainline

Figure 3-1 (Part 9 of 14). COMMON

COMMON					
Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanation
COMALC	47	70-71	$RPIC	$RPQA $RPHS $RPJA	Displacement of an item in a compression
COMAHC	49	72-73	$RPGU	None	Address minus one byte of numeric tables
COMGRC	4B	73-74	$RPSN	None	The address of the first root element that is not a mainline
COMAIC	4B	74-75	$RPGU	$RPHA	Address minus one byte of the end of numeric tables (start address minus one byte of alphabetic tables)
COMGSC	4C	75-76	$RPSN	None	The address of the first mainline element that is not in the root element
COMAJC	4D	76-77	$RPGU	$RPHA	Address minus one byte of the end of either alphabetic fields and tables or just alphabetic fields (start address of numeric fields)
COMAKC	4F	78-79	$RPHA	$RPHD	Address of end of numeric fields
COMBVC	51	80-81	$RPGU	None	Address of beginning of the object program
COMEEC	52	82	$RPGW	$RPGN $RPGR	Where compile-time tables/arrays are specified: Bit 0 1 = KEYBORD blind mode only Bit 1 Print used on MFCU Bit 2 1 = Dual I/O on printer Bit 3 1 = Extension specifications Bit 4 1 = Output-format specifications Bit 5 1 = Calculation specifications Bit 6 1 = Input specifications Bit 7 1 = BSCA specifications
COMBGC	52	82	$RPPA	$RPPG $RPPJ $RPMK	RAF and unpack information: Bit 0 1 = Unpack subroutine used by Multifile Logic Bits 1-2 Keyboard Input/Output Control routine hooks required Bit 3 1 = Unpack subroutine used by Control Fields Bit 4 Reserved Bit 5 1 = Pack needed by Random Access File Bit 6 1 = Unpack needed by Random Access File Bit 7 1 = Get RAF used by Input Mainline
COMCLC	53	82-83	$RPPO	None	Address of total output
COMCYC	55	84-85	$RPPA	$RPPB	Address of Output Processing Control routine
COMIDC	55	85	$RPEA	$RPEB $RPEC $RPEE $RPEI $RPEK $RPEO	Save area for specification type being processed

Figure 3-1 (Part 10 of 14). COMMON

COMMON					
Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanation
COMCZC	57	86-87	$RPMK	None	Address of Open Mainline
COMBBC	57	87	$RPMM	None	Number of SETLL files times 10
COMCHC	58	87-88	$RPG	$RPEE	Address of current compression for Input and Compression phases
COMBYC	59	88-89	$RPGB	$RPGN	Common output buffer address Byte 0 = High tier used for MFCM1 } Model 15 Byte 1 = High tier used for MFCM2 }
COMDAC	5A	90	$RPG	None	Address of spindle file one
COMDBC	5C	91-92	$RPG	$RPSC $RPSK	Address of beginning of disk file one
COMDCC	5E	93-94	$RPG	$RPSG	Address of end of disk file one
COMDEC	5F	95	$RPG	$RPSG	Address of spindle file two
COMDFC	61	96-97	$RPG	$RPSG	Address of beginning of disk file two
COMDGC	63	98-99	$RPG	$RPSG	Address of end of disk file two
COMBXC	65	100-101	$RPGN	None	Output buffer length
COMDKC	64	100	$RPRX	$RPSC	Number of 256-byte segments in the segment list
COMDMC	66	101-102	$RPSC	$RPSG	Address of beginning of the sort strings of object code
COMHMC	67	102-103	$RPG	None	Time compile started (Model 15 only)
COMETC	67	102-103	$RPGH	$RPGI	Address of first table load DTF
COMDNC	68	103-104	$RPRY	$RPRZ	Number of 256-byte segments of object text
COMEFC	69	104-105	$RPEW	$RPGV $RPGK $RPGL $RPJA $RPJK $RPJJ $RPHS $RPHT $RPJM	The number of errors found so far
COMCNC	69	104-105	$RPPA	None	Address of Input Processing Control routine
COMEAC	6A	106	$RPPA	None	Length of Input Processing Control routine
COMDSC	6A	105-106	$RPRY	$RPRZ	Address of the work area

Figure 3-1 (Part 11 of 14). COMMON

COMMON

Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanation
COMEGC	6C	108	$RPFA	$RPGU	File information: Bit 0 1 = File description given Bit 1 1 = Input file Bit 2 1 = Object-time table/array Bit 3 Not used Bit 4 1 = Line counter specification Bit 5 1 = Extension specifications Bit 6 Not used Bit 7 1 = Compile-time table/array data
COMENC	6C	108	$RPPA	$RPMK	Number of special devices
COMEHC	6D	109	$RPG $RPPB $RPGH $RPGI $RPHS $RPJM	$RPGN	Output information: Bit 0 1 = Match and/or control field present Bit 1 1 = Tractor 2 used for carbon (Model 6 only) Bit 2 1 = MR used with ledger card Bit 3 1 = DSPLY used with numeric Bit 4 1 = Overflow indicator with ledger card Bit 5 1 = Matrix printer in field mode (Model 6 only) Bit 6 1 = Numeric printing on keyboard (Model 6 or special devices used with Model 10 or 12) Bit 7 1 = Work area used as output buffer
COMEIC	6E	110	$RPPE	$RPJS	The number of trailer specifications in the program release level
COMCUC	6F	110-111	$RPMI	$RPMK	The length of LR Calculations Mainline
COMCUC	6F	110-111	$RPMK	None	Address of first byte of LR Calculations Mainline
COMEPC	6F	110-111	$RPG	$RPGH	Release level
COMEKC	70	112	$RPIC $RPGL $RPHS $RPJW $RPJZ $RPEW $RPHT	$RPGK	Error and Keyboard information: Bit 0 1 = Error file is full Bit 1 1 = Error caught in Input and Compression phase Bit 2 1 = Keyboard primary (Model 6) Bit 3 Reserved Bit 4 1 = Model 6 Bit 5 1 = Header card column 11 = P (Partial list) Bits 6-7 01 = Model 12 10 = Model 15 11 = External buffers specified (Program Number 5704-RG2)

Figure 3-1 (Part 12 of 14). COMMON

COMMON

Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanation
COMELC	71	113	$RPJE	None	Operation code information specified on calculation specifications: Bit 0 1 = ADD, SUB, Z-SUB, Z-ADD Bit 1 1 = MULT, DIV, MVR, XFOOT, SQRT Bit 2 1 = MOVE, MOVEL, MHLZO, MLHZO, MHHZO, MLLZO, MOVEA Bit 3 1 = SETON, SETOF, BITON, BITOF, TESTB Bit 4 1 = COMP, TESTZ Bit 5 1 = LOKUP, READ, CHAIN, FORCE, SETLL Bit 6 1 = DEBUG Bit 7 1 = GOTO, TAG, EXIT, EXCPT, BEGSR, ENDSR, EXSR, RLABL
COMEMC	72	114	$RPG	$RPMH $RPJX	Operation code information specified on calculation specifications: Bit 0 1 = DSPLY or TIME Bit 1 1 = Array control Bit 2 1 = SET or KEY (Model 6 only) Bit 3 Not used Bit 4 1 = CRT in program Bit 5 1 = IBM application program Bit 6 1 = Halt for warning diagnostics Bit 7 1 = IPCR is larger than 256 bytes; when this bit is on, COMEAC contains number of bytes over 256
COMEBC	74	115-116	$RPPA	None	Length of Output Processing Control routine
COMEVC	74	115-116	$RPG	None	Gold Key information: Byte 0 Not used Byte 1 Bit 5 1 = Magnetic Tape Bit 6 1 = 5445 Disk Bit 7 1 = BSCA
COMEWC	75	117-118	$RPG	None	Application program ID in binary: Byte 0 Not used Byte 1 Bit 0 1 = R module requested (Model 15) Bit 1 1 = D card processed Bits 2-7 Not used
COMEJC	77	119	$RPJE	$RPGH $RPGI $RPHT	File information: Bit 0 1 = Exception output Bit 1 1 = Total output Bit 2 1 = Demand files Bit 3 1 = EXCPT op code in calculation specifications Bit 4 1 = Calculation specifications Bit 5 1 = Output-format specifications Bit 6 1 = Header/trailer in program Bit 7 Not used

Figure 3-1 (Part 13 of 14). COMMON

COMMON					
Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanation
COMECC	79	120-121	$RPHA	None	Address of linkage to table dump routine
COMEZC	79	120-121	$RPGK	$RPPC	Address of the last IOCB
COMCAC	7A	122	$RPJU	None	Error information: Bit 0 1 = No compile time table data found Bit 1 1 = Compile time table data not in sequence Bit 2 1 = Table/array full Bit 3 1 = Table/array too small Bit 4 1 = Last entry in table/array blank Bits 5-7 Date Bits
COMCVC	7A	122	$RPMM	None	Block of fetch code needed in the program
COMERC	7B	123	$RPGL	None	BSCA file number
COMEXC	7D	124-125	$RPG	$RPGX	Address of first byte of print buffer
COMDTC	7F	126-127	$RPGX	$RPGY $RPGZ	Address of end of work area
COMCCC	7F	126-127	$RPGL	None	BSCA Record Available indicator

Figure 3-1 (Part 14 of 14). COMMON

Parameters to RPG II Halt Processor

This area in main storage is used to store parameters needed for the RPG II Halt Processor and can be found at COMMON+X'11C'.

IOB

Detailed descriptions of the input/output blocks (IOBs) for all devices are given in *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual*, SY21-0512.

I/O Parameters

This 18-byte area contains parameters which are moved into the disk IOB by Interphase Control routine PIOCS. PIOCS moves the parameters when this 18-byte area is called by one of the other Interphase Control routines for disk input or output. The I/O parameters are located at COMMON + X'BC'. These parameters are:

Byte	*Contents*
0-5	Parameters for a read operation
6-11	Parameters for rewriting a modified compression
12-17	Parameters for writing an object code block

Control Routine Save Area

The 26-byte control routine save area, located at COMMON + X'15C', contains addresses and pointers vital to the functioning of the compiler. The contents of this area are:

Byte	*Contents*
0-1	Address of current entry in compression block table
2-3	Address of next sector available for compressions
4-5	Current compression name
6-7	Address of first byte of current compression block
8-9	Address of the work area
10-11	Address of the last byte in the work area
12-13	End of the current compression block in the work area
14-19	Three two-byte work areas
20-21	Pointer to compression in main storage
22-23	Start address of the output buffer
24-25	End address plus one of the work area

Compression Block Table (CZATAB)

This 126-byte table, located at COMMON + X'2F6', contains control information meaningful to the Interphase Control routines. The table contains a 9-byte entry for each compression type. Phases requesting a compression block pass a flag to the Interphase Control routine indicating the type of compression. This flag is multiplied by 4.5 to find the displacement into the table of the entry for that compression type. Flags and displacements for the different compression types are as follows:

Flag	*Displacement*	*Type of compression block*
X'00'	X'00'	File description specifications
X'02'	X'09'	Extension specifications
X'04'	X'12'	Line counter specifications
X'06'	X'1B'	Data management entry points and module names
X'08'	X'24'	Input specifications
X'0A'	X'2D'	Calculations specifications
X'0C'	X'36'	Dump control compressions
X'10'	X'48'	Output-format specifications
X'12'	X'51'	Telecommunications specifications
X'14'	X'5A'	Compile time table compressions
X'16'	X'63'	Alternate collate and file translate compressions
X'18'	X'6C'	Error file
X'1A'	X'75'	Scratch file

Each 9-byte entry in the compression block table has the following format:

Byte	*Contents*
0	X'08' = Opened with update X'10' = Opened with update (used if only 512 or 768 bytes are available for work area and object code) X'20' = Last sector of compressions started X'40' = Beginning of this part is not in core X'80' = Compression is completely in core
1-2	Address of first sector of compression on disk
3-4	Number of disk sectors in current compression block
5-6	Storage address of first compression in current compression block. If the address points to X'FF', the block has not been built.
7-8	Number of first statement in this compression block.

COMPRESSION WORK AREA

The compression work area is built by the Compiler Initialization phase ($RPG). This area is normally 1024 bytes long, allowing 768 bytes for compressions and 256 bytes for output. Some Interphase Control routines are used in a special way if less than 1024 bytes of main storage are available for the work area. See *Interphase Control Routines* for a description of these routines.

COMPRESSION FORMATS

Compressions of source records are of variable length depending on the specifications. Each logical section of information in a compression is assigned an item number so that a compiler phase may call the Find Item in Compression routine (DRGCFI) to obtain the displacement from the beginning of the compression. Bits in the format bytes (items 129-131) are set on by the Input and Compression phases to indicate the presence of items in the compression. The compression block table (CZATAB) tells the type of the compression and its address.

Control Statement Compression

This compression contains information compressed from the control statement specification and is placed in the beginning of COMMON. See Figure 3-1 for the contents of the compression.

File Description Compressions

These compressions contain information compressed from each file description specification. The start address of the first file description compression is found in bytes 5-6 of the compression block table. Figure 3-2 shows the compression format. See the Control Card and File Description Specifications sheet for the columns referred to by this figure. Phases are listed in order of use by the compiler.

Extension Compressions

These compressions contain information compressed from each extension specification. The start address of the first compression is found in bytes 14-15 of the compression block table. Figure 3-3 shows the compression format. See the Extension and Line Counter Specifications sheet for the columns referred to by this figure. Phases are listed in order of use by the compiler.

Line Counter Compressions

These compressions contain information compressed from each line counter specification. The address of the first compression is found in bytes 23-24 of the compression block table. Figure 3-4 shows the compression format. See the Extension and Line Counter Specifications sheet for the columns referred to by this figure. Phases are listed in order of use by the compiler.

Input Compressions

These compressions contain information compressed from each input specification. The address of the first compression is found in bytes 41-42 of the compression block table. See Figure 3-5 for the compression format and the Input Specifications sheet for the columns referred to by this figure. Phases are listed in order of use by the compiler.

Dump Control Compressions

See Appendix B.

Calculation Compressions

These compressions contain information compressed from each calculation specification. The start address of the first compression is in bytes 50-51 of the compression block table. Figure 3-6 shows the compression format. See the Calculation Specifications sheet for the columns referred to by this figure. Phases are listed in order of use by the compiler.

Output-Format Compressions

These compressions contain information compressed from each output-format specification. The start address of the first compression is found in bytes 78-79 of the compression block table. Figure 3-7 gives the compression format. See the Output-Format Specification sheet for the columns referred to by this format. Phases are listed in the order of use by the compiler.

Telecommunications Compressions

These compressions contain information compressed from each telecommunication specification. The start address of the first compression is found at bytes 86-87 of the compression block table. Figure 3-8 gives the compression format. See the File Description Specification sheet for the columns referred to by this format. Phases are listed in order of use by the compiler.

Item Number	Byte Length	Defined/ Modified	Description
128	1	$RPEA	Length of compression in binary
129	1	$RPEA	Indicates presence of item in compression
130	1		Item 129 Bit 0 1 = Item number 132 Bit 1 1 = Item number 133 Bit 2 1 = Item number 134 Bit 3 1 = Item number 135 Bit 4 1 = Item number 136 Bit 5 1 = Item number 137 Bit 6 1 = Item number 138 Bit 7 1 = Item number 139 Item 130 Bit 0 1 = Item number 140 Bit 1 1 = Item number 141 Bit 2 1 = Item number 142 Bit 3 1 = Item number 143 Bit 4 1 = Item number 144 Bit 5 1 = Item number 145 Bit 6 1 = Item number 146 Bit 7 1 = Item number 147
131	1		Item 131 Not used
132	1	$RPEA	Bit 0 1 = Invalid compression Bit 1 1 = Invalid filename (col 7-14) and compressed as X'0' Bit 2 1 = Special IOS routine name invalid (col 54-59) and not compressed Bit 3 1 = Filename (col 7-14) missing Bit 4 1 = Input or output table file Bit 5 1 = BSCA printer type device Bit 6 1 = ASCII continuation specified for this file Bit 7 0 = Fixed length records 1 = Variable length records
		$RPGV	Bit 1 Set to 0
		$RPJK	Bit 1 1 = Debug only output to this file
133	1	$RPEA	Bits 0-3 0000 = Blank or invalid file type (col 15) 1000 = Display file 1001 = Input file 1010 = Output file 1011 = Combined file 1100 = Update file 1111 = Invalid Bits 4-7 0000 = Entry not specified (col 16) 0001 = Primary file 0010 = Secondary file 1000 = Chained file 1001 = Demand file 1010 = Record address file 1100 = Table file 1111 = Invalid file
134	1	$RPEA	Bits 0-1 00 = Entry not specified (col 18) 01 = Descending sequence 10 = Ascending sequence 11 = Invalid sequence Bits 2-3 00 = Entry not specified (col 17) 10 = End of file 11 = Invalid entry Bits 4-5 00 = Entry not specified (col 66) 01 = Unordered sequence 10 = Invalid entry (assume A) Bits 6-7 00 = Sequential file processed consecutively (col 28) (blank) 01 = Random 10 = Indexed file processed sequentially within limits 11 = Invalid entry
		$RPJK	If no entry is specified, bit 0 and 1 are set to 00 (not specified) or 10 (ascending sequence) by default
135	1	$RPEA	Bits 0-1 00 = Entry not specified (col 39) 01 = Line counter specifications 10 = Extension specifications 11 = Invalid entry Bits 2-3 00 = Sequential or direct file (col 31) (blank) 01 = Packed keys 10 = Invalid entry/A/K 11 = ADDROUT file or file processed by ADDROUT file Bit 4 Reserved Bits 5-7 000 = Sequential or direct file — I/O area (col 32) (blank) 011 = Sequential or direct file — two I/O areas 100 = ADDROUT file (T entry in column 32) 110 = Indexed file 111 = Invalid entry

Figure 3-2 (Part 1 of 4). File Description Compressions

Item Number	Byte Length	Defined/ Modified	Description
136	12	$RPEA	Filename (col 7-14) Byte 11 (Models 10 and 15) = V or D (file format for variable length tape files)
		$RPIG	Byte 11 (Models 10 and 15) = X'00'
		$RPGH	Bytes 0-7 Filename Bytes 8-9 Address of IOCB Byte 10 Bits 0-3 1000 = Consecutive processing 1001 = Indexed file processed randomly by key 1010 = Direct (disk address) file 1011 = Direct (record number) file 1100 = Indexed file 1101 = Indexed file processed sequentially within limits Bits 4-7 0001 = Input file 0010 = Output file 0011 = Update/combined file 0100 = Add/print file 0101 = Input + add file 0110 = Output + add file 0111 = Update/combined + add file 1000 = Output unordered file 1010 = Output chain file Byte 11 **Model 6** X'10' = CONSOLE (printer-keyboard) X'12' = KEYBORD (keyboard) X'40' = DISKET X'5F' = Special IOS routine X'80' = BSCA X'84' = BSCA with first-time logic X'90' = CRT (display station) X'A0' = DISK (disk unit) X'A1' = Multivolume Disk X'E1' = TRACTR1 (tractor 1) X'E2' = TRACTR2 (tractor 2) X'E9' = LEDGER (ledger card device) X'F1' = DATA96 (data recorder) X'FF' = Invalid entry **Model 12** X'10' = Console X'40' = DISKET X'50' = READ42 (1442 Card Read-Punch) X'5F' = Special X'60' = Tape X'80' = BSCA X'84' = BSCA (first time logic) X'A0' = DISK (5444 simulation area) X'C0' = DISK40 or DISK45 (3340 main data area) X'E0' = PRINTER — carriage 1 X'E8' = PRINTR2 — carriage 2 X'F0' = MFCU1 — primary hopper X'F8' = MFCU2 — secondary hopper X'FF' = Invalid entry **Model 10** X'10' = CONSOLE (Printer-keyboard) X'40' = DISKET X'50' = READ42 (1442 Card Read-Punch) X'5F' = Special IOS routine X'60' = TAPE X'80' = BSCA X'84' = BSCA with first-time logic X'A0' = DISK (disk unit) A1 Multivolume disk X'A1' = Multivolume disk X'C0' = DISK45 X'E0' = PRINTER (line printer — carriage 1) X'E8' = PRINTR2 (line printer — carriage 2) X'F0' = MFCU1 (MFCU primary hopper) X'F8' = MFCU2 (MFCU secondary hopper) **Model 15** X'01' = Device independent input X'02' = Device independent output X'18' = CRT77 X'40' = DISKET X'50' = READ42 X'58' = READ01 X'5F' = SPECIAL IOS routine X'60' = TAPE X'80' = BSCA

Figure 3-2 (Part 2 of 4). File Description Compressions

Item Number	Byte Length	Defined/ Modified	Description
			Model 15 (continued) X'A0' = DISK44 X'C0' = DISK45, DISK40 X'E0' = PRINTER X'E4' = PRINT84 X'F0' = MFCU1 X'F2' = MFCM1 X'F4' = MFCM2 X'F8' = MFCU2 X'FF' = Invalid entry
		$RPGI	*Bytes 2-3. Address of DTF for this file
		$RPGH	*Bytes 10-11. Data management entry point
		$RPGI	*Bytes 10-11. Data management entry point
		$RPPA	*Bytes 4-5. Address of linkage to IPCR routine for this file
		$RPPB	*Bytes 6-7. Address of linkage to OPCR routine for this file
137	2	$RPEA	Block length in binary (col 20-23)
138	2	$RPEA	Record length in binary (col 24-27)
139	1	$RPEA	**Model 6** X'10' = CONSOLE (printer-keyboard) X'12' = KEYBORD (keyboard) X'40' = DISKET X'5F' = Special IOS routine X'80' = BSCA X'84' = BSCA with first-time logic X'90' = CRT (display station) X'A0' = DISK (disk unit) X'E1' = TRACTR1 (tractor 1) X'E2' = TRACTR2 (tractor 2) X'E9' = LEDGER (ledger card device) X'F1' = DATA96 (data recorder) or DATA29 (80 col card device) X'FF' = Invalid entry **Model 10** X'00' = Blank device X'10' = CONSOLE (printer-keyboard) X'40' = DISKET X'50' = READ42 (1442 Card Read-Punch) X'5F' = Special IOS routine X'60' = TAPE X'80' = BSCA X'84' = BSCA with first time logic X'A0' = DISK (disk unit) X'C0' = DISK45 X'E0' = PRINTER (line printer — carriage 1) X'E8' = PRINTR2 (line printer — carriage 2) X'F0' = MFCU1 (MFCU primary hopper) X'F8' = MFCU2 (MFCU secondary hopper) X'FF' = Invalid entry **Model 12** X'10' = Console X'40' = DISKET X'50' = READ42 (1442 Card Read-Punch) X'5F' = Special X'60' = Tape X'80' = BSCA X'84' = BSCA (first time logic) X'A0' = DISK (5444 simulation area) X'C0' = DISK40 or DISK45 (3340 main data area) X'E0' = PRINTER — carriage 1 X'E8' = PRINTR2 — carriage 2 X'F0' = MFCU1 — primary hopper **Model 15** X'01' = Device independent input X'02' = Device independent output X'18' = CRT77 X'40' = DISKET X'50' = READ42 X'58' = READ01 X'5F' = SPECIAL IOS routine X'60' = TAPE X'80' = BSCA X'A0' = DISK44 X'C0' = DISK45, DISK40

* Indicates an addition or change to a previously defined area.

Figure 3-2 (Part 3 of 4). File Description Compressions

Item Number	Byte Length	Defined/ Modified	Description
			Model 12 (continued) X'F8' = MFCU2 — secondary hopper X'FF' = Invalid entry **Model 15 (continued)** X'E0' = PRINTER X'E4' = PRINT84 X'F0' = MFCU1 X'F2' = MFCM1 X'F4' = MFCM2 X'F8' = MFCU2 X'FF' = Invalid entry
140	1	$RPEA	Keyfield or record address length in binary (col 29-30)
141	2	$RPEA	Keyfield start location in binary (col 35-38)
142	2	$RPEA	Cylinder index in binary (col 60-65)
143	1	$RPEA	Number of extents (col 68-69)
		$RPGI	Maximum skip value
144	2	$RPEA	Overflow indicator (col 33-34)
		$RPJX	Overflow default
		$RPJZ	Overflow default
		$RPGF	Mask and displacement
145	2	$RPEA	File conditioners U1 - U8 (col 71-72)
		$RPGF	Mask and displacement
146	6	$RPEA	Name of special IOS routine (col 54-59), or array name if continuation
		$RPJW	Bytes 0-1. Sequence number of extension specifications defining the array used, if SPECIAL is specified
147	1	$RPEA	Continuation Support Bits 0-2 000 = Blank in col 53 100 = SPECIAL used 010 = K in col 53 110 = Invalid Bits 3-4 00 = Blank or invalid in col 70 01 = U in col 70 10 = R in col 70 11 = N in col 70 Bits 5-7 Reserved

Figure 3-2 (Part 4 of 4). File Description Compressions

Item Number	Byte Length	Defined/ Modified	Description
128	1	$RPEB	Compression length in binary
129	1	$RPEB	Indicates presence of items in compression
130	1		Item 129 Bit 0 = 1 = Item number 132 Bit 1 = 1 = Item number 133 Bit 2 = 1 = Item number 134 Bit 3 = 1 = Item number 135 Bit 4 = 1 = Item number 136 Bit 5 = 1 = Item number 137 Bit 6 = 1 = Item number 138 Bit 7 = 1 = Item number 139 Item 130 Bit 0 = 1 = Item number 140 Bit 1 = 1 = Item number 141 Bit 2 = 1 = Item number 142 Bits 3-7 = Not used
131	1		Item 131 = Not used
132	2	$RPEB	Byte 0, Bit 0 = 1 = Invalid compression Bit 1 = 1 = Invalid From Filename (col 11-18) Bit 2 = 1 = Invalid To Filename (col 19-26) Bit 3 = 1 = Invalid Table or Array Name (col 27-32) Bit 4 = 1 = Invalid Alternate Table or Array Name (col 46-51) Bit 5 = Not used
		$RPGW/ $RPGY**	*Byte 0, Bit 6 = 0 = Object-time table or array name 1 = Compile-time table or array name *Bit 7 = 0 = Object-time alternate table or array name 1 = Compile-time alternate table or array name Byte 1 = Not used
133	1	$RPEB	Bit 0 = 1 = Table in ascending sequence (col 45) Bit 1 = 1 = Table in descending sequence Bits 2-3 = 00 = Unpacked or alphameric (col 43) or invalid 01 = Packed 10 = Binary Bits 4-7 = 0000-1001 = Decimal positions entry (col 44) 1010 = Alphameric table or array
134	1	$RPEB	Bit 0 = 1 = Alternate table in ascending sequence (col 57) Bit 1 = 1 = Alternate table in descending sequence Bits 2-3 = 00 = Unpacked or alphameric (col 55) 01 = Packed 10 = Binary Bits 4-7 = 0000-1001 = Decimal positions entry (col 56) 1010 = Alphameric table or array
135	14	$RPEB	FROM filename (col 11-18)
		$RPGI	Defines bytes 0-11. See note 1
136	14	$RPEB	TO filename (col 19-26)
		$RPGI	Defines bytes 0-11. See note 1
		$RPPJ	*Byte 0, Bit 0 = 1 = MFCU print (Models 10 and 12) *Byte 4, Bit 6 = 1 = End of file *Bit 7 = 1 = File translate *Bytes 12-13 = Address of OPCR routine

* Indicates an addition or change to a previously defined area.

Figure 3-3. Extension Compressions (Part 1 of 3)

Item Number	Byte Length	Defined/ Modified	Description
137	6	$RPEB	Table or array name (col 27-32)
		$RPGW/ $RPGY**	Byte 0. Alphameric length = Length minus one Numeric length = Length minus one in the numeric position; number of decimal positions in the zone portion Byte 1 Bit 0 1 = Look-ahead field Bit 1 Not used Bits 2-3 01 = Array name 10 = Table name Bits 4-5 00 = Table sequence not specified 01 = Table is descending 10 = Table is ascending Bit 6 0 = Length is alphameric (see byte 0) 1 = Length is numeric Bit 7 Not used Bytes 2-3. Address of rightmost byte of first element Bytes 4-5. DTT address
138	2	$RPEB	Number of entries per table or array in binary (col 36-39)
139	2	$RPEB	Length of entry in binary (col 40-42)
		$RPGW/ $RPGY**	If the length specified is invalid, the length is set to 15 (X'0F').
140	2	$RPEB	Number of entries per record (col 33-35)
141	6	$RPEB	Alternate table or array name (col 46-51)
		$RPGW/ $RPGY**	Byte 0 1. Alphameric length = Length minus one 2. Numeric length = Length minus one in the numeric position; number of decimal positions in the zone portion Byte 1 Bit 0 1 = Look-ahead field Bit 1 Not used Bits 2-3 01 = Array name 10 = Table name Bits 4-5 00 = Table sequence not specified 01 = Table is descending 10 = Table is ascending Bit 6 0 = Length is alphameric (see byte 0) Bit 7 Not used Bytes 2-3. Address of rightmost byte of first element Bytes 4-5. DTT address
142	2	$RPEB	Length of entry in binary (col 52-54)
		$RPGW/ $RPGY**	If the length specified is invalid, the length is set to 15 (X'0F').

Figure 3-3. Extension Compressions (Part 2 of 3)

Note 1:

Byte 0, Bits 3-7	Sequence number
Bytes 1-2	DTF address
Byte 3 (Model 6)	X'10' = CONSOLE (printer-keyboard) X'12' = KEYBORD (keyboard) X'40' = DISKET X'5F' = Special IOS routine X'80' = BSCA X'84' = BSCA with first-time logic X'90' = CRT (display station) X'A0' = DISK (disk unit) X'E1' = TRACTR1 (tractor 1) X'E2' = TRACTR2 (tractor 2) X'E9' = LEDGER (ledger card device) X'F1' = DATA96 (data recorder) X'FF' = Invalid entry
Byte 3 (Model 10)	X'10' = CONSOLE (printer-keyboard) X'40' = DISKET X'50' = READ42 (1442 Card Read-Punch) X'5F' = Special IOS routine X'60' = TAPE X'80' = BSCA X'84' = BSCA with first time logic X'A0' = DISK (disk unit) X'C0' = DISK45 X'E0' = PRINTER (line printer — carriage 1) X'E8' = PRINTR2 (line printer — carriage 2) X'F0' = MFCU1 (MFCU primary hopper) X'F8' = MFCU2 (MFCU secondary hopper) X'FF' = Invalid entry
Byte 3 (Model 12)	X'10' = Console X'40' = DISKET X'50' = READ42 (1442 Card Read-Punch) X'5F' = Special X'60' = Tape X'80' = BSCA X'84' = BSCA (first time logic) X'A0' = DISK (5444 simulation area) X'C0' = DISK40 or DISK45 (3340 main data area) X'E0' = PRINTER — carriage 1 X'E8' = PRINTR2 — carriage 2 X'F0' = MFCU1 — primary hopper X'F8' = MFCU2 — secondary hopper X'FF' = Invalid entry
Byte 3 (Model 15)	X'01' = Device independent input X'02' = Device independent output X'18' = CRT77 X'40' = DISKET X'50' = READ42 X'58' = READ01 X'5F' = SPECIAL IOS routine X'60' = TAPE X'80' = BSCA X'A0' = DISK44 X'C0' = DISK45, DISK40 X'E0' = PRINTER X'E4' = PRINT84 X'F0' = MFCU1 X'F0' = MFCU1 X'F2' = MFCM1 X'F4' = MFCM2 X'F8' = MFCU2 X'FF' = Invalid entry
Byte 4, Bits 0-3	0001 = Primary file 0010 = Secondary file 1000 = Chained file 1001 = Demand file 1010 = Record address file 1100 = Table file
Bits 4-5	00 = No specified sequence 01 = Descending sequence 10 = Ascending sequence
Byte 5, Bits 0-1	00 = Sequential file 01 = Indexed file 10 = Direct file 11 = ADDROUT file
Bits 2-3	00 = Consecutive processing 01 = Random processing 10 = Indexed file processed sequentially by key 11 = Indexed file processed sequentially within limits
Bits 4-6	001 = Display file 011 = Update file 100 = Combined file 101 = Regular output file 110 = Input file 111 = Unordered sequence
Bit 7	1 = Addition specified
Byte 6, Bit 0	0 = Variable format 1 = Fixed format
Bit 1	0 = Unblocked file format 1 = Blocked file format
Bits 2-3	01 = Extension specifications 10 = External indicators
Bits 4-6	000 = Indexed file processed consecutively 010 = Indexed key (alphameric) 011 = Indexed key (packed) 100 = Record identification (disk address) 110 = Record number
Bit 7	1 = Dual I/O
Byte 7	Overflow mask
Bytes 8-9	Record length
Bytes 10-11	IOCB address

* Indicates an addition or change to a previously defined area.

** Either phase $RPGW or $RPGY is loaded depending on whether an object-time table or a compile time table has been loaded.

Figure 3-3. Extension Compressions (Part 3 of 3)

Item Number	Byte Length	Defined/ Modified	Description
128	1	$RPEC	Compression length in binary
129	1	$RPEC	Indicates presence of item in compression when bits are on (Bit=1) Item 129 Bit 0 1 = Item number 132 Bit 1 1 = Item number 133 Bit 2 1 = Item number 134 Bit 3 1 = Item number 135 Bits 4-7 Not used
130	1		Item 130 Not used
131	1		Item 131 Not used
132	1	$RPEC	Bit 0 1 = Invalid compression Bit 1 1 = Invalid or missing filename Bits 2-7 Not used
133	13	$RPEC	Filename (col 7-14)
		$RPGI	Byte 0 Bits 3-7 Sequence Number Bytes 1-2 DTF address Byte 3 **Model 6** X'10' = CONSOLE (printer-keyboard) X'12' = KEYBORD (keyboard) X'40' = DISKET X'5F' = Special IOS routine X'80' = BSCA X'84' = BSCA with first-time logic X'90' = CRT (display station) X'A0' = DISK (disk unit) X'E1' = TRACTR1 (tractor 1) X'E2' = TRACTR2 (tractor 2) E'E9' = LEDGER (ledger card device) X'F1' = DATA96 (data recorder) X'FF' = Invalid entry **Model 12** X'10' = Console X'40' = DISKET X'50' = READ42 (1442 Card Read-Punch) X'5F' = Special X'60' = Tape X'80' = BSCA X'84' = BSCA (first time logic) X'A0' = DISK (5444 simulation area) X'C0' = DISK40 or DISK45 (3340 main data area) X'E0' = PRINTER — carriage 1 X'E8' = PRINTR2 — carriage 2 X'F0' = MFCU1 — primary hopper X'F8' = MFCU2 — secondary hopper X'FF' = Invalid entry **Model 10** X'10' = CONSOLE (printer-keyboard) X'40' = DISKET X'50' = READ42 (1442 Card Read-Punch) X'5F' = Special IOS routine X'60' = TAPE X'80' = BSCA X'84' = BSCA with first-time logic X'A0' = DISK (disk unit) X'C0' = DISK45 X'E0' = PRINTER (line printer — carriage 1) X'E8' = PRINTR2 (line printer — carriage 2) X'F0' = MFCU1 (MFCU primary hopper) X'F8' = MFCU2 (MFCU secondary hopper) X'FF' = Invalid entry **Model 15** X'01' = Device Independent input X'02' = Device Independent output X'18' = CRT77 X'40' = DISKET X'50' = READ42 X'58' = READ01 X'5F' = SPECIAL IOS routine X'60' = TAPE X'80' = BSCA X'A0' = DISK44 X'C0' = DISK45, DISK40 X'E0' = PRINTER X'E4' = PRINT84 X'F0' = MFCU1

Figure 3-4 (Part 1 of 2). Line Counter Compressions

Item Number	Byte Length	Defined/ Modified	Description		
					Model 15 (continued) X'F0' = MFCU1 X'F2' = MFCM1 X'F4' = MFCM2 X'F8' = MFCU2 X'FF' = Invalid entry
			Byte 4	Bits 0-3 0001 = Primary file 0010 = Secondary file 1000 = Chained file 1001 = Demand file 1010 = Record address file	Bits 4-5 00 = No specified sequence 01 = Descending sequence 10 = Ascending sequence Bit 6 1 = End of file
			Byte 5	Bits 0-1 00 = Sequential file 01 = Indexed file 10 = Direct file 11 = ADDROUT file Bits 2-3 00 = Consecutive processing 01 = Random processing 10 = Indexed file processed sequentially by key 11 = Indexed file processed sequentially within limits	Bits 4-6 001 = Display file 011 = Update file 100 = Combined file 101 = Regular output file 110 = Input file 111 = Unordered sequence Bit 7 1 = Addition specified
			Byte 6	Bit 0 0 = Variable format 1 = Fixed format Bit 1 0 = Unblocked file format 1 = Blocked file format Bits 2-3 01 = Line Counter specifications 10 = External indicators	Bits 4-6 000 = Indexed file processed consecutively 010 = Indexed key (alphameric) 011 = Indexed key (packed) 100 = Record identification (disk address) 110 = Record number Bit 7 1 = Dual I/O
			Byte 7	Overflow mask	
			Bytes 8-9	Record Length	
			Bytes 10-11	IOCB address	
			Byte 12	Bits 0-3 Operation Code Bits 4-7 Number of parameters	
134	2	$RPEC	Form length in binary		
135	2	$RPEC	Overflow number in binary		

Figure 3-4 (Part 2 of 2). Line Counter Compressions

Item Number	Byte Length	Defined/ Modified	Description
128	1	$RPEI	Compression length in binary
129	1	$RPEI	Indicates presence of item in compression when bits are on (Bit=1)
130	1		Item 129 Bit 0 = Item number 132; Bit 1 = Item number 133; Bit 2 = Item number 134; Bit 3 = Item number 135; Bit 4 = Item number 136; Bit 5 = Item number 137; Bit 6 = Item number 138; Bit 7 = Item number 139
131	1		Item 131 Bit 0 = Item number 148; Bit 1 = Item number 149; Bit 2 = Item number 150; Bit 3 = Item number 151; Bit 4 = Item number 152; Bit 5 = Item number 153; Bit 6 = Item number 154; Bit 7 = Reserved
			Item 130 Bit 0 = Item number 140; Bit 1 = Item number 141; Bit 2 = Item number 142; Bit 3 = Item number 143; Bit 4 = Item number 144; Bits 5-7: 100 = Item number 145; 110 = Item numbers 145 and 146 (2-byte extension); 101 = Item numbers 145 and 147 (4-byte extension)
132	3	$RPEI	Byte 0, Bit 0 = 1 = Invalid compression; Bit 1 = 1 = Invalid filename; Bit 2 = 1 = Invalid field name; Bit 3 = 1 = TR (col 19-20); Bytes 1-2 = Sequence (col 15-16)
		$RPJA	Bytes 1-2 = X'0000'
		$RPGW/ $RPGY**	*Byte 1, Bit 4 = 1 = Field name definition resolved; *Bit 5 = 1 = Index resolved; *Bit 7 = 1 = Symbol in table (set by phase $RPHA)
		$RPHC	*Byte 1, Bit 4 = 1 = Field name definition resolved; *Bit 5 = 1 = Index resolved; *Bit 7 = 1 = Symbol in table (set by phase $RPHA)
		$RPJJ	*Byte 0, Bit 5 = 1 = Control fields move optimized; *Bit 6 = 1 = Match fields move optimized; *Bit 7 = 1 = Move field optimized
		$RPJS	Bytes 0-1 = Address of the trailer table for the current trailer specification is placed in the trailer record type specification; Byte 0, Bit 0 = 1 = For all record type compressions (AND/OR lines included) of header portion of spread card to flag as header record
		$RPPE	*Bytes 1-2 = Address for the record ID routine to return to when the record is identified
		$RPPF	*Bytes 1-2 = Address of the record identification parameter
		$RPPN	*Bytes 1-2 = Address of parameter list for chain and demand files

Figure 3-5. Input Compressions (Part 1 of 6)

Item Number	Byte Length	Defined/ Modified	Description	
133	1	$RPEI	Bits 0-1	00 = Number is blank (col 17) 01 = Number contains a 1 10 = Number contains an N 11 = Invalid entry
			Bit 2	1 = Option contains an 0 (col 18)
			Bit 3	1 = AND
			Bit 4	1 = OR
			Bits 5-7	Stacker Select (col 42) 000 = blank or invalid 100 = stacker 4 101 = stacker 1 110 = stacker 2 111 = stacker 3
		$RPJA	*Bit 0	0 = Alphabetic sequence entry (col 15-16) 1 = Numeric sequence entry
			*Bit 1	0 = Number contains an N (col 17) 1 = Number contains a 1
			*Bit 2	1 = Optional
		$RPJS	The length minus one of one trailer of current trailer specification is placed in the record type specification.	
134	1	$RPEI	Bit 0	1 = Look-ahead field (col 19-20)
			Bits 2-3	00 = Unpacked or alphameric (col 43) 01 = Packed 10 = Binary
			Bits 4-7	0000-1001 = Decimal positions entry (col 52) 1010 = Alphameric field
		$RPJA	If bit is on, it is propagated for all look-ahead fields specified. A decimal position defaults to zero.	
		$RPGW/ $RPGY**	*Bit 0	0 = Alphabetic sequence entry (col 15-16) 1 = Numeric sequence entry
			*Bit 1	0 = Object-time table or array name 1 = Compile-time table or array name
		$RPHS	This byte is not defined for demand or chain files.	
			*Bit 1	1 = Numeric sequence used in this file
			*Bit 2	1 = Console file — record length needed in IOS parameter
			*Bit 3	1 = Numeric sequence
			*Bit 4	1 = Match fields
			*Bit 5	1 = Control fields
			*Bit 6	1 = Input fields
			*Bit 7	1 = Stacker select needed
		$RPJJ	*Bits 4-7. Numeric portion set to packed, binary, or sterling length	
135	15	$RPEI	Filename (col 7-14)	
		$RPGI	Byte 0, Bits 3-7	Sequence number
			Bytes 1-2	Pre-open DTF address

Figure 3-5. Input Compressions (Part 2 of 6)

Item Number	Byte Length	Defined/ Modified	Description

Byte 3

Model 6

X'10' = CONSOLE (printer keyboard)
X'12' = KEYBORD (keyboard)
X'40' = DISKET
X'5F' = SPECIAL (special IOS routine)
X'80' = BSCA
X'84' = BSCA (BSCA with first-time logic)
X'90' = CRT (display station)
X'A0' = DISK (disk unit)
X'E1' = TRACTR1 (tractor 1)
X'E2' = TRACTR2 (tractor 2)
X'E9' = LEDGER (ledger card device)
X'F1' = DATA96 (data recorder)
X'FF' = Invalid entry

Model 10

X'10' = CONSOLE (printer keyboard)
X'40' = DISKET
X'50' = READ42 (1442 Card Read-Punch)
X'5F' = Special IOS routine
X'60' = TAPE
X'80' = BSCA
X'84' = BSCA with first time logic
X'A0' = DISK (disk unit)
X'C0' = DISK45
X'E0' = PRINTER (line printer — carriage 1)
X'E8' = PRINTR2 (line printer — carriage 2)
X'F0' = MFCU1 (MFCU primary hopper)
X'F8' = MFCU2 (MFCU secondary hopper)
X'FF' = Invalid entry

Model 12

X'10' = Console
X'40' = DISKET
X'50' = READ42 (1442 Card Read-Punch)
X'5F' = Special
X'60' = Tape
X'80' = BSCA
X'84' = BSCA (first time logic)
X'A0' = DISK (5444 simulation area)
X'C0' = DISK40 or DISK45 (3340 main data area)
X'E0' = PRINTER — carriage 1
X'E8' = PRINTR2 — carriage 2
X'F0' = MFCU1 — primary hopper
X'F8' = MFCU2 — secondary hopper
X'FF' = Invalid entry

Model 15

X'01' = Device independent Input
X'02' = Device independent output
X'18' = CRT77
X'40' = DISKET
X'50' = READ42
X'58' = READ01
X'5F' = SPECIAL IOS routine
X'60' = TAPE
X'80' = BSCA
X'A0' = DISK44
X'C0' = DISK45, DISK40
X'E0' = PRINTER
X'E4' = PRINT84
X'F0' = MFCU1
X'F2' = MFCM1
X'F4' = MFCM2
X'F8' = MFCU2
X'FF' = Invalid entry

Byte 4

Bits 0-3
0001 = Primary file
0010 = Secondary file
1000 = Chained file
1001 = Demand file
1010 = Record address file

Bits 4-5
00 = No specified sequence
01 = Descending sequence
10 = Ascending sequence

Bit 6
1 = End of file

Bit 7
1 = File translate

Byte 5

Bits 0-1
00 = Sequential file
01 = Indexed file
10 = Direct file
11 = ADDROUT file

Bits 4-6
001 = Display file
011 = Update file
100 = Combined file
101 = Regular output file

Figure 3-5. Input Compressions (Part 3 of 6)

Item Number	Byte Length	Defined/ Modified	Description
			110 = Input file Bits 2-3 111 = Unordered sequence 00 = Consecutive processing 01 = Random processing Bit 7 10 = Indexed file processed sequentially by key 1 = Addition specified 11 = Indexed file processed sequentially within limits
		Byte 6	Bit 0 Bits 4-6 Byte 7 Overflow mask 0 = Variable format 000 = Indexed file processed consecutively 1 = Fixed format Bytes 8-9 Record length 010 = Indexed key (alphameric) Bytes 10-11 IOCB address Bit 1 0 = Unblocked file format 011 = Indexed key (packed) Byte 12 Bits 0-3 Operation code 1 = Blocked file format 100 = Record identification (disk address) Bits 2-3 110 = Record number Bits 4-7 Number of parameters 01 = Extension or line counter specifications Bit 7 10 = External indicators 1 = Dual I/O
	$RPPJ		*Bytes 13-14 Address of linkage to IPCR for this file
136	2	$RPEI	Record identification indicator (col 19-20)
		$RPGF	Mask and displacement
137	2	$RPEI	Position in binary (col 21-24)
		$RPJJ	Position minus one
138	2	$RPEI	See note 1
139	2	$RPEI	Position in binary (col 28-31)
		$RPJJ	Position minus one
140	2	$RPEI	See note 1
141	2	$RPEI	Position in binary (col 35-38)
		$RPJJ	Position minus one
142	2	$RPEI	See note 1
143	2	$RPEI	FROM field location in binary (col 44-47)
		$RPJA	See note 2
144	2	$RPEI	TO field location in binary (col 48-51)
		$RPJA	See note 2
		$RPJJ	TO field location minus one
145	6	$RPEI	Field or array name (col 53-58)
		$RPGF	See note 4

Figure 3-5. Input Compressions (Part 4 of 6)

Item Number	Byte Length	Defined/ Modified	Description
		$RPGW/ $RPGY**	If an array name, the following is set: Byte 0. See note 3 Byte 1. See note 4 Bytes 2-3. Address of rightmost byte of first element for an array or an array with a variable index, or, the address of the rightmost byte of indexed element for an array with a constant index. Bytes 4-5. DTT address
		$RPHC	If a field name, the following is set: Byte 0. See note 3 Byte 1. See note 4 Bytes 2-3. Address of the rightmost byte of the field Bytes 4-5. X'0000'
146	2	$RPGW/ $RPGY**	For an array with a constant index, numeric value of the array index is set.
147	4	$RPHC	For an array with a variable index, the following is set: Byte 0. See note 3 Byte 1. See note 4 Bytes 2-3. Address of rightmost byte of field
148	2	$RPEI	Control level (col 59-60)
		$RPHS	Zone portion of byte zero is set off
149	1	$RPEI	Match fields (col 61-62)
		$RPHS	Zone portion of byte is set off
150	2	$RPEI	Field record relation indicator (col 63-63)
		$RPGF	Mask and displacement
151	2	$RPEI	Plus indicator (col 65-66)
		$RPGF	Mask and displacement
152	2	$RPEI	Minus indicator (col 67-68)
		$RPGF	Mask and displacement
153	2	$RPEI	Zero or blank indicator (col 69-70)
		$RPGF	Mask and displacement
154	2	Reserved	

Note 1:

Byte 0, Bit 0 — 1 = N entry (col 25, 32, 39)
Bits 4-7 — 0011 = C entry (col 26, 33, 40)
0100 = D entry
1001 = Z entry
0000 = Entry not specified

Byte 1 — Character (col 27, 34, 41)

Note 2: If an invalid entry is specified, then default values are set:

FROM Field Location entry defaults to 1.
TO Field Location entry defaults to 15 if numeric and to 256 if alphameric.

Note 3:

Alphameric length = Length minus one
Numeric length = Length minus one in the numeric portion; number of decimal positions in the zone portion

Figure 3-5. Input Compressions (Part 5 of 6)

Note 4:

Bit 0	1 = Look-ahead field
Bits 2-3	00 = Field name
	01 = Array name
	10 = Table name
Bits 4-5	00 = Table sequence not specified
	01 = Table is descending or field name is UDATE related field
	10 = Table is ascending or, if RLABL is specified, field name or mask and displacement of the indicator
Bit 6	0 = Length is alphameric
	1 = Length is numeric

* Indicates an addition or change to a previously defined area.

** Either phase $RPGW or $RPGY is loaded depending on whether an object-time table or a compile-time table has been loaded.

Figure 3-5. Input Compressions (Part 6 of 6)

Item Number	Byte Length	Defined/ Modified	Description
128	1	$RPEK	Compression length in binary
129	1	$RPEK	Indicates the presence of an item in a compression Item 129 Bit 0 1 = Item number 132 Bit 1 1 = Item number 133 Bit 2 1 = Item number 134 Bit 3 1 = Item number 135 Bit 4 1 = Item number 136 Bit 5 1 = Item number 137 Bit 6 1 = Item number 138 Bit 7 1 = Item number 139
130	1	$RPEK	Second format byte which indicates the presence of an item in a compression Factor 1 definition — contents of Factor 1 Bits 0-3 1000 = Item number 140 — field or array name 1100 = Item numbers 140 and 141 (4-byte extension) — alphameric or numeric literal 1010 = Item numbers 140 and 142 (2-byte extension) — array with constant index 1001 = Item numbers 140 and 143 (6-byte extension) — array with variable index Factor 2 definition — contents of Factor 2 Bits 4-7 1000 = Item number 144 — array or field name 1100 = Item numbers 144 and 145 (4-byte extension) — alphameric or numeric literal 1010 = Item numbers 144 and 146 (3-byte extension) — array with constant index 1001 = Item numbers 144 and 147 (6-byte extension) — array with variable index 1011 = Item numbers 144, 146, and 147 (9-byte extension) — filename
131	1	$RPGK	Third format byte which indicates the presence of an item in a compression Result field definition — contents of result field Bits 0-2 100 = Item number 148 — field or array name 110 = Item numbers 148 and 149 (2-byte extension) — array with constant index 101 = Item numbers 148 and 150 (4-byte extension) — array with variable index

Figure 3-6. Calculation Compressions (Part 1 of 10)

Item Number	Byte Length	Defined/ Modified	Description
			Remainder of Calculation Specification defined
			Bit 3 1 = Item number 151 Bit 4 1 = Item number 152 Bit 5 1 = Item number 153 Bit 6 1 = Item number 154 Bit 7 Not used
132	4	$RPEK	Byte 0, Bit 0 1 = Invalid compression Bit 1 1 = Factor 1 (col 18-27) invalid Bit 2 1 = Factor 2 (col 33-42) invalid Bit 3 1 = Result Field (col 43-48) invalid Bit 4 1 = Factor 2 contains a filename Bit 5 1 = Factor 2: *BOTH
		$RPGV	*Byte 0, *Bit 5 1 = Factor 1 table/array shortest *Bit 6 1 = Factor 2 table/array shortest *Bit 7 1 = Result Field table/array shortest
		$RPGW/ $RPGY**	*Byte 1, Bit 0 1 = Factor 1 resolved *Bit 1 1 = Factor 1 index resolved *Bit 2 1 = Factor 2 resolved *Bit 3 1 = Factor 2 index resolved *Bit 4 1 = Result Field resolved *Bit 5 1 = Result Field index resolved *Bit 7 1 = Symbol in table (set by phase $RPHA)
		$RPHC	*Byte 1, Bit 0 1 = Factor 1 resolved *Bit 1 1 = Factor 1 index resolved *Bit 2 1 = Factor 2 resolved *Bit 3 1 = Factor 2 index resolved *Bit 4 1 = Result Field resolved *Bit 5 1 = Result Field index resolved *Bit 7 1 = Symbol in table (set by phase $RPHA)
		$RPMB	*Byte 2. Length of object code block for the operation code
		$RPMI	*Byte 2. Length of object code block for the compression
		$RPLJ	*Byte 2. Length of object code block for the operation code
		$RPMG	Byte 2. Length of object code block for the operation code.
		$RPQA	*Bytes 0-1. Address of generated object code
			*Byte 2. Length of object code block for the operation

Figure 3-6. Calculation Compressions (Part 2 of 10)

Item Number	Byte Length	Defined/ Modified	Description	
		$RPMH	*Byte 2. Length of object code block for the operation.	
			*Byte 3. Length of leading array control, or for move type operation code, the first byte of a 2-byte mask (second byte is in item 134)	
		$RPMP	*Bytes 0-1. Address of generated object code	
133	1	$RPEK	Bit 0	1 = SR (col 7-8)
			Bit 2	1 = Half adjust (col 53)
			Bit 3	1 = AN (col 7-8)
			Bit 4	1 = OR (col 7-8)
			Bit 5	1 = Not (col 15)
			Bit 6	1 = Not (col 12)
			Bit 7	1 = Not (col 9)
		$RPGG	*Bit 1	1 = DEBUG not specified so treat as comments
			*Bit 2	0 = No half adjust (this bit is set off if a DIV operation is followed by an MVR operation)
				1 = Half adjust
		$RPJG	*Bit 1	1 = Compressions relative to SET operation on SET/KEY combinations are flagged to be ignored when generating object code
		$RPQK		Object code mask for determining what (if any) code to generate for array control.
134	1	$RPEK	Bits 0-3	Not used
			Bits 4-7	X'0'-X'9' = Decimal position entry (col 52)
				X'A' = Alphameric field
		$RPGW/ $RPGY**	*Bit 0	0 = Object-time table (Factor 1)
				1 = Compile-time table (Factor 1)
			*Bit 1	0 = Object-time table (Result Field)
				1 = Compile-time table (Result Field)
			*Bit 2	0 = Object-time table (Factor 2)
				1 = Compile-time table (Factor 2)
			Bit 3	Not used
		$RPGX	*Bits 4-7. Not used	
		$RPLG	Permanent length (bit 4 or 5 for SET or KEY)	
		$RPMB	Mask of object code to be generated	
		$RPLJ	*Bits 0-2	000 = Tabset
				100 = Space/skip and/or position print element
				010 = Ledger card eject
				001 = Key operation code resulting indicator
			*Bits 4-7	0001 = Command keys (no manual mode or field name)
				0010 = Manual mode (no command keys)
				0011 = Manual mode (command keys)
				0100 = Field name (no command keys)
				0101 = Field name (command keys)
		$RPMH	Mask of remainder of object code to be generated or second byte of a 2-byte mask for a move operation code. The first byte is in item 132.	
135	5	$RPEK	Operation codes (contents of col 28-32)	

Figure 3-6. Calculation Compressions (Part 3 of 10)

Item Number	Byte Length	Defined/ Modified	Description
		$RPEW	Byte 0. Operation code mask. See note 2 Byte 1. Field light mask. See note 2
		$RPJG	*Byte 1. X'FF' = SET/KEY combination found
		$RPHU	If SETLL operation code, the following are set: Byte 1. Key length minus 1 Bytes 2-3. Address of low key area
		$RPLR	Bytes 3-4. Address of parameter list if required for operation code.
136	2	$RPEK	Control level indicator (col 7-8)
		$RPGF	Mask and displacement
		$RPMI	See note 1
137	2	$RPEK	Indicator (col 10-11)
		$RPGF	Mask and displacement
		$RPGG	*Byte 1. Bit 1 = zero in displacement if Not (N) is specified for indicator
		$RPMI	See note 1
138	2	$RPEK	Indicator (col 13-14)
		$RPGF	Mask and displacement
		$RPGG	*Byte 1. Bit 1 = zero in displacement if Not (N) is specified for indicator
		$RPMI	See note 1
139	2	$RPEK	Indicator (col 16-17)
		$RPGF	Mask and displacement
		$RPGG	*Byte 1. Bit 1 = zero in displacement if Not (N) is specified for indicator
		$RPMI	See note 1
140	6	$RPEK	Factor 1 (contents of col 18-27)
		$RPGW/ $RPGY**	If an array name, the following are set: Byte 0. See note 3 Byte 1. See note 4 Bytes 2-3. Address of rightmost byte of first element for an array or an array with a variable index, or the address of rightmost byte of indexed element for an array with a constant index Bytes 4-5. DTT address
		$RPHC	If a field name, the following are set: Byte 0. See note 3 Byte 1. See note 4 Bytes 2-3. Address of rightmost byte of field Bytes 4-5. X'0000'
		$RPLN	For a literal, the following are set: Byte 0. See note 3 Byte 1. See note 4
		$RPLR	For a literal, the following are set: Bytes 2-3. Address of rightmost byte of literal Bytes 4-5. These bytes are not used

Figure 3-6. Calculation Compressions (Part 4 of 10)

Item Number	Byte Length	Defined/ Modified	Description
		$RPMP	If BEGSR and EXSR are specified: Bytes 0-1. Start address of subroutine Bytes 2-3. Length of subroutine Bytes 4-5. Subroutine identifier
		$RPQD/ $RPQG/ $RPQH	Bytes 2-3. Object code address where element address is to be placed (for array table or array with variable index)
141	4	$RPEK	Item acts as 4-byte extension to item 140 when alphemeric or numeric literal is defined in Factor 1.
142	2	$RPEK	Item acts as a 2-byte extension to item 140 when an array with constant index is specified in Factor 1.
143	6	$RPEK	Item acts as a 6-byte extension to item 140 when an array with variable index is specified in Factor 1.
		$RPHC	For an array with a variable index, the following are set: Byte 0. See note 3 Byte 1. See note 4 Bytes 2-3. Address of rightmost byte of Tag field Bytes 4-5. X'0000'
144	6	$RPEK	Factor 2 (contents of col 33-42)
		$RPGW/ $RPGY**	If an array name, the following are set: Byte 0. See note 3 Byte 1. See note 4 Bytes 2-3. Address of rightmost byte of first element for an array or an array with a variable index, or, address of rightmost byte of indexed element for an array with a constant index. Bytes 4-5. DTT address
		$RPHC	If a field name, the following are set: Byte 0. See note 3 Byte 1. See note 4 Bytes 2-3. Address of rightmost byte of field Bytes 4-5. X'0000'

Figure 3-6. Calculation Compressions (Part 5 of 10)

Item Number	Byte Length	Defined/ Modified	Description
		$RPGI	If a filename, the following are set: Byte 0, Bit 0 = 1 = MFCU print (Model 10) Bits 3-7 = Sequence number Bytes 1-2 = DTF address Byte 3 **Model 6** X'10' = CONSOLE (printer-keyboard) X'12' = KEYBORD (keyboard) X'40' = DISKET X'5F' = Special IOS routine X'80' = BSCA X'84' = BSCA with first-time logic X'90' = CRT (display station) X'A0' = DISK (disk unit) X'E1' = TRACTR1 (tractor 1) X'E2' = TRACTR2 (tractor 2) X'E9' = LEDGER (ledger card device) X'F1' = DATA96 (data recorder) X'FF' = Invalid entry **Model 10** X'10' = CONSOLE (printer-keyboard) X'40' = DISKET X'50' = READ42 (1442 Card Read-Punch) X'5F' = Special IOS routine X'60' = TAPE X'80' = BSCA X'84' = BSCA with first-time logic X'A0' = DISK (disk unit) X'C0' = DISK45 X'E0' = PRINTER — carriage 1 X'E8' = PRINTER — carriage 2 X'F0' = MFCU1 (MFCU primary hopper) X'F8' = MFCU2 (MFCU secondary hopper) X'FF' = Invalid entry **Model 12** X'10' = Console X'40' = DISKET X'50' = READ42 (1442 Card Read-Punch) X'5F' = Special X'60' = Tape X'80' = BSCA X'84' = BSCA (first time logic) X'A0' = DISK (5444 simulation area) X'C0' = DISK40 or DISK45 (3340 main data area) X'E0' = PRINTER — carriage 1 X'E8' = PRINTR2 — carriage 2 X'F0' = MFCU1 — primary hopper X'F8' = MFCU2 — secondary hopper X'FF' = Invalid entry **Model 15** X'01' = Device independent input X'02' = Device independent output X'18' = CRT77 X'40' = DISKET X'50' = READ42 X'58' = READ01 X'5F' = SPECIAL IOS routine X'60' = TAPE X'80' = BSCA X'A0' = DISK44 X'C0' = DISK45, DISK40 X'E0' = PRINTER X'E4' = PRINT84 X'F0' = MFCU1 X'F0' = MFCU1 X'F2' = MFCM1 X'F4' = MFCM2 X'F8' = MFCU2 X'FF' = Invalid entry Byte 4 Bits 0-3 0001 = Primary file 0010 = Secondary file 1000 = Chained file 1001 = Demand file 1010 = Record address file Bits 4-5 00 = No specified sequence 01 = Descending sequence 10 = Ascending sequence Bit 6 1 = End of file Bit 7 1 = File translate

Figure 3-6. Calculation Compressions (Part 6 of 10)

Item Number	Byte Length	Defined/ Modified	Description
			Byte 5 Bits 0-1 00 = Sequential file 01 = Indexed file 10 = Direct file 11 = ADDROUT file Bits 2-3 00 = Consecutive processing 01 = Random processing 10 = Indexed file processed sequentially by key 11 = Indexed file processed sequentially within limits Bits 4-6 001 = Display file 011 = Update file 100 = Combined file 101 = Regular output file 110 = Input file 111 = Unordered sequence Bit 7 1 = Addition specified
		$RPLN	For a literal, the following are set: Byte 0. See note 3 Byte 1. See note 4
		$RPMH	Bytes 2-3 are modified if specified conditions exist: 1. Address of the leftmost byte of the element or field if MHHZO or MHLZO operation code specified. 2. Address of the rightmost byte of the part of the element or field to be moved if MOVEL operation code specified or if the length of Factor 2 is greater than the length of the Result Field.
		$RPLR	For a literal, the following are set: Bytes 2-3. Address of rightmost byte of literal Bytes 4-5. These bytes are not used
		$RPMP	If BEGSR and EXSR are specified: Bytes 0-1 Start address of subroutine Bytes 2-3 Length of subroutine Bytes 4-5 Subroutine identifier
		$RPQD/ $RPQG/ $RPQH $RPQE	Bytes 2-3. Object code address of where element address is to be placed (for array table or array with variable index)
145	4	$RPEK	Item acts as a 4-byte extension to item 144 when an alphameric or numeric literal is defined in Factor 2.
146	3	$RPEK	Item acts as a 3-byte extension to item 144 when an array with constant index is specified in Factor 2 or when filename is given.
		$RPGW/ $RPGY**	For an array with a constant index, the following are set: Bytes 0-1. Numeric value of the array index Byte 2. X'00'
		$RPGI	If filename is specified in Factor 2, the following are set: Byte 0, Bit 0: 0 = Variable format; 1 = Fixed format Bit 1: 0 = Unblocked file format; 1 = Blocked file format Bits 2-3: 01 = Extension or line counter specifications; 10 = External indicators Byte 1: Overflow mask Byte 2: First byte of record length, second byte in item number 147

Figure 3-6. Calculation Compressions (Part 7 of 10)

Item Number	Byte Length	Defined/ Modified	Description
			Bits 4-6 000 = Indexed file processed consecutively 010 = Indexed key (alphameric) 011 = Indexed key (packed) 100 = Record identification (disk address) 110 = Record number Bit 7 1 = Dual I/O
		$RPRW	If filename is specified, the following is set: Byte 2. First byte of the address of the entry point to subsegment, if CHAIN or READ operation code is specified. Second byte is contiguous in item number 147 for filename.
147	6	$RPEK	This item acts as a 6-byte extension to item 144 when factor 2 contains an array with variable index or a filename (which combines with item 146 to form a 9-byte extension).
		$RPHC	For an array with a variable index, the following are set: Byte 0. See note 3 Byte 1. See note 4 Bytes 2-3. Address of rightmost byte of the Tag field Bytes 4-5. X'0000'
		$RPGI	When filename is specified in Factor 2, this item is modified as follows: Byte 0. Second byte of record length (first byte given contiguously in item 146). Bytes 1-2. IOCB address Byte 3. Not used Byte 4. Key length
		$RPPJ	Bytes 4-5. Address of OPCR routine if DSPLY, DEBUG, SET, or KEY operation code is specified; address of IPCR routine if READ or CHAIN operation code is specified.
		$RPRW	If filename is specified, the following is set: Byte 0. Second byte of the address of the entry point to the subsegment of object code, if CHAIN or READ operation code is specified. First byte was specified in item 146.

Figure 3-6. Calculation Compressions (Part 8 of 10)

Item Number	Byte Length	Defined/ Modified	Description
148	6	$RPEK	Result Field (contents of col 43-48)
		$RPGW/ $RPGY**	If an array, the following are set: Byte 0. See note 3 Byte 1. See note 4 Bytes 2-3. Address of rightmost byte of first element for an array or an array with a variable index, or, address of rightmost byte of indexed element for an array with constant index Bytes 4-5. DTT address
		$RPHC	If a field name, the following are set: Byte 0. See note 3 Byte 1. See note 4 Bytes 2-3. Address of rightmost byte of field Bytes 4-5. X'0000'
		$RPMH	Bytes 2-3. These bytes are modified as follows, under the specified conditions: 1. Address of the leftmost byte of the element or field if MHHZO or MLHZO operation code specified. 2. Address of the rightmost byte of the part of the element or field to be moved if MOVEL operation code specified or if the length of Factor 2 is less than the length of the Result Field.
		$RPQD/ $RPQG/ $RPQH	Bytes 2-3. Object code address where element address is to be placed (for array table or array with variable index)
149	2	$RPEK	This item acts as a 2-byte extension to item 148, when an array with constant index is defined in the Result Field.
		$RPGW/ $RPGY**	For an array with a constant index, the following are set: Bytes 0-1. Numeric value of the array index
150	4	$RPEK	This item acts as a 4-byte extension to item 148, when an array with variable index is defined in the Result Field.
		$RPHC	For an array with a variable index, the following are set: Byte 0. See note 3 Byte 1. See note 4 Bytes 2-3. Address of rightmost byte of the field Bytes 4-5. X'0000'
151	2	$RPEK	Field length in binary (col 49-51)
152	2	$RPEK	Plus indicator (col 54-55)
		$RPGF	Mask and displacement
153	2	$RPEK	Minus indicator (col 56-57)
		$RPGF	Mask and displacement
154	2	$RPEK	Zero or blank indicator (col 58-59)
		$RPGF	Mask and displacement

Figure 3-6. Calculation Compressions (Part 9 of 10)

Note 1:

The indicators are reordered so that indicator strings that began in this compression are placed back into the compression with the longest first, in decreasing order by length. The start of an indicator string is flagged by setting Bit 0 equal to zero in the displacement for the indicator. In the next compression, the length of the string minus the length of the group in which it started is placed in the same relative position as in this compression. In all subsequent compressions through which the string continues, the indicator is replaced by X'0000', a no-test indicator.

Note 2:

Byte 0		X'00' = ENDSR
		X'01' = EXCPT
		X'0E' = SETON
		X'0F' = SETOF
		X'20' = RLABL
		X'23' = MVR
		X'2E' = TESTZ
		X'40' = EXIT
		X'41' = GOTO
		X'42' = EXSR
		X'50' = DEBUG
		X'51' = DSPLY
		X'52' = FORCE
		X'53' = READ
		X'60' = Z-ADD
		X'61' = Z-SUB
		X'62' = SORT
		X'63' = XFOOT
		X'64' = MHHZO
		X'65' = MHLZO
		X'66' = MLHZO
		X'67' = MLLZO
		X'68' = MOVE
		X'69' = MOVEL
		X'6A' = MOVEA
		X'6C' = BITON
		X'6D' = BITOF
		X'6E' = TESTB
		X'71' = KEY } Model 6 only
		X'72' = SET } Model 6 only
		X'80' = BEGSR
		X'81' = TAG
		X'84' = TIME (Model 15 only)
		X'CE' = COMP
		X'CF' = LOKUP
		X'D0' = CHAIN
		X'D3' = SETLL
		X'E0' = ADD
		X'E1' = SUB
		X'E2' = MULT
		X'E3' = DIV
		X'FF' = Invalid entry
Byte 1, (for KEY or SET operation codes)	Bit 0	1 = Field light 1 } Model 6 only
	Bit 1	1 = Field light 2 } Model 6 only
	Bit 2	1 = Field light 3 } Model 6 only
	Bit 3	1 = Field light 4 } Model 6 only
	Bit 4	1 = Field light 5 } Model 6 only
	Bit 5	1 = Field light 6 } Model 6 only
	Bit 6	1 = Field light 7 } Model 6 only
	Bit 7	1 = Field light 8 } Model 6 only

Note 3:

1. Alphameric length = Length minus one
2. Numeric length = length minus one in the numeric portion; number of decimal positions in the zone portion

Note 4:

Bit 0	1 = Look-ahead field
Bits 2-3	00 = Field name
	01 = Array name
	10 = Table name
Bits 4-5	00 = Table sequence not specified
	01 = Table is descending or field name is UDATE related field
	10 = Table is ascending or, if RLABL specified, field name or mask and displacement or indicator
Bit 6	0 = Length is alphameric (see Note 10)
	1 = Length is numeric
Bit 7	1 = Shorter array
	During phase $RPHC, 1 = Field is used

* Indicates an addition or change to a previously defined area.

** Either phase $RPGW or $RPGY is loaded depending on whether an object-time table or a compile-time table has been loaded.

Figure 3-6. Calculation Compressions (Part 10 of 10)

Item Number	Byte Length	Defined/ Modified	Description
128	1	$RPEO	Compression length in binary
129	1	$RPEO	Indicates presence of an item in the compression.
130	1		Item 129 Bit 0 = 1 = Item number 132 Bit 1 = 1 = Item number 133 Bit 2 = 1 = Item number 134 Bit 3 = 1 = Item number 135 Bit 4 = 1 = Item number 136 Bit 5 = 1 = Item number 137 Bit 6 = 1 = Item number 138 Bit 7 = 1 = Item number 139 Item 130 Bit 0 = 1 = Item number 140 Bit 1 = 1 = Item number 141 Bit 2 = 1 = Item number 142 Bits 3-5 = 100 = Item number 143 110 = Item number 144 (2-byte extension) 101 = Item number 145 (4-byte extension) Bit 6 = 1 = Item number 146 Bit 7 = 1 = Item number 147
131	1		Item 131 Bit 0 = 1 = Item number 148 Bit 1 = 1 = Item number 149 Bits 2-7 = Not used
132	4	$RPEO	Byte 0, Bit 0 = 1 = Invalid compression Bit 1 = 1 = Invalid Filename (col 7-14) Bit 2 = 1 = Invalid Field Name (col 32-37) Bit 3 = 1 = Invalid constant (col 45-70) Bit 4 = 1 = Field specification
		$RPGV	*Bytes 2-3. Number of elements in the table/array
		$RPGW/ $RPGY**	*Byte 1, *Bit 4 = 1 = Field name resolved *Bit 5 = 1 = Index resolved *Bit 7 = 1 = Symbol in table (set by phase $RPHA)
		$RPHC	*Byte 1, *Bit 4 = 1 = Field name resolved *Bit 5 = 1 = Index resolved *Bit 7 = 1 = Symbol in table (set by phase $RPHA)
		$RPJO	*Byte 1, Bit 0 = 1 = 4th tier print for full array greater than 96 (MFCU)
		$RPJO	Byte 1, Bits 0-3 = High Print head used (MFCM)
		$RPLB	*Byte 0, *Bit 1 = 1 = Fourth tier print needed *Bit 2 = 1 = Add file (first field line only) *Bit 3 = 1 = *PRINT follows (first field line only *Bit 4 = 1 = First field line *Bit 5 = 1 = Subsegment fields code *Bit 6 = 1 = Last field line *Bit 7 = 1 = Last record line
		$RPLG	Byte 0, Bit 1 is set to 0.
		$RPLR	*Byte 0, Bits 0-3 = Length minus one of the parameters to be moved to the DTF
		$RPLV	*Byte 0, X'80' = Output Field Move can be combined with the next output field move
		$RPMA	*Byte 0, X'80' = Output Field Move can be combined with the next output field move *Byte 1. Length of object code
		$RPMM	*Bytes 2-3. Address of object code block to be generated for this compression
		$RPPO	*Byte 0, Bit 6 is set to 0 *Bytes 2-3. Address of object code block for MFCU punch and print

Figure 3-7. Output-format Compressions (Part 1 of 6)

Item Number	Byte Length	Defined/ Modified	Description	
133	1	$RPEO	Bits 0-1	00 = Blank or invalid type (col 15) 01 = Heading or detail records 10 = Total records 11 = Records to be written during calculation time (E in col 15)
			Bit 2	1 = ADD (col 16-18)
			Bit 3	1 = AND (col 14-16)
			Bit 4	1 = OR (col 14-15)
			Bits 5-7	000 = No entry in col 16 001 = Eject (Model 6 only) 010 = Fetch overflow 011 = Stacker 5 (Model 15 only) 100 = Stacker 4 (Model 10 or 12) 101 = Stacker 1 (Model 10 or 12) 110 = Stacker 2 (Model 10 or 12) 111 = Stacker 3 (Model 10 or 12)
		$RPMA	Bit 6	1 = For ledger card (Model 6 only)
134	1	$RPEO	Bit 5	1 = Not (col 23)
			Bit 6	1 = Not (col 26)
			Bit 7	1 = Not (col 19)
		$RPGW/ $RPGY**	*Bit 1	0 = Object-time table name 1 = Compile-time table name
		$RPLB	*Bit 0	1 = No field lines follow
			*Bits 1-2	01 = 1P indicator 10 = LR indicator 11 = Overflow indicator
			*Bit 3	1 = First record for output type
			*Bit 4	1 = Last record or field for output record type
135	1	$RPEO	Bit 0	1 = PAGE, PAGE1, or PAGE2 (col 32-37)
			Bit 1	1 = Blank After (col 39)
			Bits 2-3	00 = Unpacked or alphameric (col 44) 01 = Packed 10 = Binary
			Bit 4	1 = *PLACE (col 32-37)
			Bit 5	1 = Ledger card control identification (N in col 40)
			Bit 6	1 = *PRINT (col 32-37) Model 10 or 12
			Bit 7	1 = MFCU print (* col 40) Model 10 or 12
		$RPGB	Bit 7	1 = MFCM print (Model 15 only)
		$RPJO	High print position on record (MFCM record specifications only).	
		$RPLR	Parameter address if more than one byte of Data Management parameters	
		$RPLG	X'00' = no parameter required X'01' = parameter required	
136	2	$RPEO	Space before/after (col 17-18)	
			Bits 0-3	X'0' – X'9' = Space before entry X'F' = No entry
			Bits 4-7	X'0' – X'9' = Space after entry X'F' = No entry
		$RPJO	Length of edited output (field specification only); high end position on record for VLR on tape (record specification only, Item 135 and byte 0 of item 136 used); high punch position on record (MFCM record specifications only).	
		$RPLR	Parameter address (item number 135 and Byte 0 of item number 136)	
		$RPLV	X'00' = no parameter required X'01' = parameter required	
137	1	$RPEO	Skip before in binary (col 19-20) (optional)	

Figure 3-7. Output-format Compressions (Part 2 of 6)

Item Number	Byte Length	Defined/ Modified	Description
138	1	$RPEO	Skip after in binary (col 20-21) (optional)
139	15	$RPEO	Filename (col 7-14) (optional)
		$RPGI	Byte 0, Bit 0 = 1 = MFCU print Bits 3-7 = Sequence number
			Bytes 1-2 = DTF address

Byte 3

Model 6

- X'10' = CONSOLE (printer-keyboard)
- X'12' = KEYBORD (keyboard)
- X'40' = DISKET
- X'5F' = Special IOS routine
- X'80' = BSCA
- X'84' = BSCA (BSCA with first-time logic)
- X'90' = CRT (display station)
- X'A0' = DISK (disk unit)
- X'E1' = TRACTR1 (tractor 1)
- X'E2' = TRACTR2 (tractor 2)
- X'E9' = LEDGER (ledger card device)
- X'F1' = DATA96 (data recorder)
- X'FF' = Invalid entry

Model 10

- X'10' = CONSOLE (printer-keyboard)
- X'40' = DISKET
- X'50' = READ42 (1442 Card Read-Punch)
- X'5F' = Special IOS routine
- X'60' = TAPE
- X'80' = BSCA
- X'84' = BSCA with first-time logic
- X'A0' = DISK (disk unit)
- X'C0' = DISK45
- X'E0' = PRINTER (line printer — carriage 1)
- X'E8' = PRINTR2 (line printer — carriage 2)
- X'F0' = MFCU1 (MFCU primary hopper)
- X'F8' = MFCU2 (MFCU secondary hopper)
- X'FF' = Invalid entry

Model 12

- X'10' = Console
- X'40' = DISKET
- X'50' = READ42 (1442 Card Read-Punch)
- X'5F' = Special
- X'60' = Tape
- X'80' = BSCA
- X'84' = BSCA (first time logic)
- X'A0' = DISK (5444 simulation area)
- X'C0' = DISK40 or DISK45 (3340 main data area)
- X'E0' = PRINTER — carriage 1
- X'E8' = PRINTR2 — carriage 2
- X'F0' = MFCU1 — primary hopper
- X'F8' = MFCU2 — secondary hopper
- X'FF' = Invalid entry

Model 15

- X'01 = Device Independent Input
- X'02' = Device Independent Output
- X'18' = CRT77
- X'40' = DISKET
- X'50' = READ42
- X'58' = READ01
- X'5F' = SPECIAL IOS routine
- X'60' = TAPE
- X'80' = BSCA
- X'A0' = DISK44
- X'C0' = DISK45, DISK40
- X'E0' = PRINTER
- X'E4' = PRINT84
- X'F0' = MFCU1
- X'F2' = MFCM1
- X'F4' = MFCM2
- X'F8' = MFCU2
- X'FF' = Invalid entry

Byte 4

Bits 0-3
- 0001 = Primary file
- 0010 = Secondary file
- 1000 = Chained file
- 1001 = Demand file
- 1010 = Record Address file

Bits 4-5
- 00 = No specified sequence
- 01 = Descending sequence
- 10 = Ascending sequence

Bit 6
- 1 = End of file

Bit 7
- 1 = File translate

Figure 3-7. Output-format Compressions (Part 3 of 6)

Item Number	Byte Length	Defined/ Modified	Description
			Byte 5 Bits 0-1 00 = Sequential file 01 = Indexed file 10 = Direct file 11 = ADDROUT file Bits 2-3 00 = Consecutive processing 01 = Random processing 10 = Indexed file processed sequentially by key 11 = Indexed file processed sequentially within limits Bits 4-6 001 = Display file 011 = Update file 100 = Combined file 101 = Regular output file 110 = Input file 111 = Unordered sequence Bit 7 1 = Addition specified
			Byte 6 Bit 0 0 = Variable format 1 = Fixed format Bit 1 0 = Unblocked file format 1 = Blocked file format Bits 2-3 01 = Extension or line counter specifications 10 = External indicators Bits 4-6 000 = Indexed file processed consecutively 010 = Indexed key (alphameric) 011 = Indexed key (packed) 100 = Record identification (disk address) 110 = Record number Bit 7 1 = Dual I/O
			Byte 7 Overflow mask
			Bytes 8-9 Record length
			Bytes 10-11 IOCB address
		$RPPJ	Bytes 12-13. Address of OPCR routine
140	2	$RPEO	Output indicator (col 24-25) (optional)
		$RPGF	Mask and displacement
		$RPLB	For output record lines, 1P indicators are zeroed out for heading and detail output, or LR indicators are zeroed on LR output. Overflow indicators are always zeroed out.
141	2	$RPEO	Output indicator (col 27-28) (optional)
		$RPGF	Mask and displacement
		$RPLB	For output record lines, 1P indicators are zeroed out for heading and detail output; LR indicators are zeroed out for LR output. Overflow indicators are always zeroed out.
142	2	$RPEO	Output indicator (col 30-31) (optional)
		$RPGF	Mask and displacement
		$RPLB	For output record lines, 1P indicators are zeroed out for heading and detail output; LR indicators are zeroed out for LR output. Overflow indicators are always zeroed out.

Note: Items 143, 144, and 145 are related as follows:
143-field name (6 bytes)
143-array name (6 bytes)
143 and 144-array with constant index
143 and 145-array with variable index

Figure 3-7. Output-format Compressions (Part 4 of 6)

Item Number	Byte Length	Defined/ Modified	Description
143	6	$RPEO	Field name (col 32-37) (optional)
		$RPGW/ $RPGY**	If an array name, the following is set: Byte 0 = 1. Alphameric length = length minus one 2. Numeric length = length minus one in the numeric portion; number of decimal positions in the zone portion Byte 1. See note 1 Bytes 2-3. Address of rightmost byte of last element for an array or an array with a variable index, or address of rightmost byte of indexed element for an array with a constant index. Bytes 4-5. DTT address
		$RPHC	If a field name, the following is set: Byte 0. 1. Alphameric length = length minus one 2. Numeric length = length minus one in the numeric portion; number of decimal positions in the zone portion Byte 1. See note 1 Bytes 2-3. Address of rightmost byte of field Bytes 4-5. X'0000'
		$RPJO	*Bytes 2-3. For arrays, address of rightmost byte of first element
		$RPLV	*Bytes 0-1. Cumulative length of combined output fields move *Byte 2, bit 6 is set to zero
144	2	$RPGW/ $RPGY**	For an array with a constant index, the following is set: Bytes 0-1. Numeric value of the array index
145	4	$RPHC	For an array with a variable index, the following is set: Byte 0. 1. Alphameric length = length minus one 2. Numeric length = length minus one in the numeric portion; number of decimal positions in the zone portion Byte 1. See note 1 Bytes 2-3. Address of rightmost byte of field
146	1	$RPEO	Edit codes (col 38) (optional)
		$RPHT	Bit 0 0 = Edit pattern 2 (no thousands demarcation) 1 = Edit pattern 1 (thousands demarcation) Bit 1 1 = Date field Bit 2 0 = Print zeros 1 = Print blanks for zero balance Bit 3 1 = CR for negative balance Bit 4 1 = Minus sign for negative balance Bit 5 1 = Asterisk fill Bit 6 1 = Floating dollar sign Bit 7 1 = Edit code Z (col 38)
147	2	$RPEO	End position in binary of the output record (col 40-43); End position in "Tier/End Position Notation" for print only on MFCM files.
		$RPGB	Converted end position for Print Only on MFCM files.
		$RPJO*	End position minus 1 of the output record
148	2		Reserved
149	7-25 (Variable length item)	$RPEO	Byte 0 = length of constant or edit word without quotes Bytes 1-24 = Constant or edit word (col 45-70)

Figure 3-7. Output-format Compressions (Part 5 of 6)

Item Number	Byte Length	Defined/ Modified	Description
		$RPLN	Byte 0 = Length minus one Bytes 1-24. If edit word: Byte 3 X'01' = CR for negative balance X'10' = Zero suppression X'20' = Floating dollar sign X'40' = Fixed dollar sign X'80' = Asterisk fill Byte 4 Zero suppression or asterisk fill length minus one Byte 5 Length minus one of start of edit word to end of status Byte 6 Length minus one of start of edit word to start of status
		$RPLR	Bytes 0-1 = Address of constant or edit word
		$RPLV	Byte 0 = cumulative length of combined Output Fields moves if a constant is present Bytes 1-24 = If constant, address is adjusted by the length added to it

Note 1 (Items 143, 145):

Bit 0 1 = Look-ahead field
Bits 2-3 00 = Field name
01 = Array name
10 = Table name
Bits 4-5 00 = Table sequence not specified
01 = Table is descending or field name is UDATE related field
10 = Table is ascending or, if RLABL specified field name or mask and displacement of indicator
Bit 6 0 = Length if alphameric
1 = Length is numeric

* Indicates an addition or change to a previously defined area.
** Either phase $RPGW or $RPGY is loaded depending on whether an object-time table or a compile-time table has been loaded.

Figure 3-7. Output-format Compressions (Part 6 of 6)

Item Number	Byte Length	Defined/ Modified	Description		
128	1	$RPEE	Compression length in binary		
129	1	$RPEE	Indicates presence of item in compression when bits are on (Bit=1)		
			Item 129	Bit 0	1 = Item number 132
				Bit 1	1 = Item number 133
				Bit 2	1 = Item number 134
				Bit 3	1 = Item number 135
				Bit 4	1 = Item number 136
				Bit 5	1 = Item number 137
				Bit 6	1 = Item number 138
				Bit 7	1 = Item number 139
130	1		Item 130	Bits 0-3	1000 = Item number 140
					1100 = Item number 140 and 141 (1-byte extension)
					1010 = Item number 140 and 142 (2-byte extension)
					1001 = Item number 140 and 143 (6-byte extension)
				Bits 4-7	1000 = Item number 144
					1100 = Item number 144 and 145 (1-byte extension)
					1010 = Item number 144 and 146 (2-byte extension)
					1001 = Item number 144 and 147
131	1		Item 131	Bits 0-3	1000 = Item number 148
					1100 = Item number 148 and 149 (1-byte extension)
					1010 = Item number 148 and 150 (2-byte extension)
					1001 = Item number 148 and 151 (6-byte extension)
				Bit 4	1 = Item number 152
				Bit 5	1 = Item number 153
				Bit 6	1 = Item number 154
				Bit 7	1 = Item number 155

Figure 3-8. Telecommunications Compressions (Part 1 of 5)

Item Number	Byte Length	Defined/ Modified	Description	
132	2	$RPEE	Byte 0, Bit 0	1 = Invalid specification
			Bit 1	1 = This station ID invalid
			Bit 2	1 = Remote station ID invalid
			Bit 3	1 = Dial Number invalid
			Bit 4	1 = Filename missing or invalid
			Bit 5	1 = Polling characters invalid
			Bit 6	1 = Addressing characters invalid
133	1	$RPEE	Byte 0, Bits 0-1	00 = Transmitter (col 16)
				01 = Receiver
				10 = Not applicable
				11 = Error
			Bits 2-3	00 = Blank (col 17)
				01 = Tributary
				10 = Not applicable
				11 = Error
			Bits 4-5	00 = N/Blank (col 19)
				01 = Yes, transparent feature
				10 = Not applicable
				11 = Error
			Bits 6-7	00 = Blank (col 60)
				01 = Last file processed
				10 = Not applicable
				11 = Error

Figure 3-8. Telecommunications Compressions (Part 2 of 5)

Item Number	Byte Length	Defined/ Modified	Description
134	1	$RPEE	Byte 0, Bits 0-4 Unused Bits 5-7 000 = Explicit or symbolic entry for autocall (col 20) 001 = Autoanswer specified 010 = Manual answer 100 = Manual call 101 = Blank 111 = Error
135	1	$RPEE	Byte 0, Bits 0-1 00 = Point to point nonswitched or blank (col 15) 01 = Switched 10 = Multipoint for nonswitched 11 = Error Bit 2 0 = EBCDIC, blank or error (col 18) 1 = ASCII Bit 3 0 = Blank (col 52) 1 = Interblock check characters (ITB) or error
136	2	$RPEE	Wait time in binary (col 55-57)
137	13	$RPEE	Filename (col 7-14)
138	2	$RPEE	Transmission terminator indicator (col 53-54)
139	2	$RPEE	Record available indicator (col 58-59)
140	6	$RPEE	This stations ID (col 33-39)
		$RPGW/ $RPGY**	If an array name, the following is set: Byte 0. See note 1 Byte 1. See note 2 Bytes 2-3. Address of rightmost byte of first element or indexed element Bytes 4-5. DTT address
		$RPHC	If a field name, the following is set: Byte 0. See note 1 Byte 1. See note 2 Bytes 2-3. Address of rightmost byte of field Bytes 4-5. X'0000'
		$RPLN	If a literal, the following is set: Byte 0. See note 1 Byte 1. See note 2
		$RPLR	Bytes 2-3. Address of rightmost byte of literal Bytes 4-5. These bytes are not used
141	1	$RPEE	This station ID (col 39)
142	2	$RPGW/ $RPGY**	If an array with a constant index, numeric value of the index is set
143	6	$RPHC	For an array with a variable index, the following is set: Byte 0. See note 1 Byte 1. See note 2 Bytes 2-3. Address of rightmost byte of index Bytes 4-5. X'0000'

Figure 3-8. Telecommunications Compressions (Part 2 of 4)

Item Number	Byte Length	Defined/ Modified	Description
144	6	$RPEE	Remote station ID (col 41-47)
		$RPGW/ $RPGY**	If an array name, the following is set: Byte 0. See note 1 Byte 1. See note 2 Bytes 2-3. Address of rightmost byte of first element or indexed element Bytes 4-5. DTT address
		$RPHC	If a field name, the following is set: Byte 0. See note 1 Byte 1. See note 2 Bytes 2-3. Address of rightmost byte of field Bytes 4-5. X'0000'
		$RPLN	If a literal, the following is set: Byte 0. See note 1 Byte 1. See note 2
		$RPLR	Bytes 2-3. Address of rightmost byte of literal Bytes 4-5. These bytes are not used
145	1	$RPEE	Remote station ID (col 47)
146	2	$RPGW/ $RPGY**	If an array with a constant index, numeric value of the index is set
147	6	$RPHC	For an array with a variable index, the following is set: Byte 0. See note 1 Byte 1. See note 2 Bytes 2-3. Address of rightmost byte of index Bytes 4-5. X'0000'
148	11	$RPEE	Dial number (col 21-31)
		$RPGW/ $RPGY**	If an array name, the following is set: Byte 0. See note 1 Byte 1. See note 2 Bytes 2-3. Address of rightmost byte of first element or indexed element Bytes 4-5. DTT address
		$RPHC	If a field name, the following is set: Byte 0. See note 1 Byte 1. See note 2 Bytes 2-3. Address of rightmost byte of field Bytes 4-5. X'0000'
		$RPLN	If a literal, the following is set: Byte 0. See note 1 Byte 1. See note 2
		$RPLR	Bytes 2-3. Address of rightmost byte of literal Bytes 4-5. These bytes are not used
149	1	$RPEE	Dial number (extension 1)
150	2	$RPGW/ $RPGY**	If an array with a constant index, numeric value of the index is set
151	6	$RPHC	For an array with a variable index, the following is set: Byte 0. See note 1 Byte 1. See note 2 Bytes 2-3. Address of rightmost byte of index Bytes 4-5. X'0000'

Figure 3-8. Telecommunications Compressions (Part 3 of 4)

Item Number	Byte Length	Defined/ Modified	Description
152	1	$RPEE	Polling characters
153	1	$RPEE	Addressing characters
154	4	$RPEE	Remote terminal (col 48-51) 2770 = 2770 channel 1 2771 = 2770 channel 1 2772 = 2770 channel 2 2773 = 2770 channel 3 2774 = 2770 channel 4 2780 = 2780
		$RPEW	
			Byte 0 Compressed Terminal value X'40' = 2770 channel 4 X'50' = 2770 channel 1 X'60' = 2770 channel 2 X'70' = 2770 channel 3 X'80' = 2780
			Byte 1 Compressed Device value 2770 X'11' = 2213-1 X'12' = 2213-2 X'13' = 545-3 X'14' = 545-4 X'15' = 2502-1 X'16' = 2502-2 X'17' = 5496-1 X'18' = 5496-2 2780 X'01' = 1442-1 X'02' = 1442-2 X'03' = 1443
			Byte 2 Device Type X'01' = reader X'02' = punch X'03' = printer
			Byte 3 Device Code
155	6	$RPEE	Remote Device 1443 = IBM 1443 Printer 1442-1 = IBM 1442 Card Read/Punch (card read) 1442-2 = IBM 1442 Card Read/Punch (card punch) 2213-1 = IBM 2213 Printer, Model 1 2213-2 = IBM 2213 Printer, Model 2 0545-3 = IBM 0545 Card Punch, Model 3 0545-4 = IBM 0545 Card Punch, Model 4 2502-1 = IBM 2502 Card Reader, Model 1 2502-2 = IBM 2502 Card Reader, Model 2 5496-1 = IBM 5496 Data Recorder (card read) 5496-2 = IBM 5496 Data Recorder (card punch)

* Indicates an addition or change to a previously defined area.

Note 1 (Items 140, 143, 144, 147, 148, 151):

1. Alphameric length = Length minus one
2. Numeric length = length minus one in the numeric portion; number of decimal positions in the zone portion

Note 2 (Items 140, 143, 144, 147, 148, 151):

Bit 0	1 = Look-ahead field
Bits 2-3	00 = Field name 01 = Array name 10 = Table name
Bits 4-5	00 = Table sequence not specified 01 = Table is descending or, if RLABL specified, field name or mask and displacement of indicator
Bit 6	0 = Length is alphameric (see Note 1) 1 = Length is numeric

Figure 3-8. Telecommunications Compressions (Part 5 of 5)

ALTERNATE COLLATING SEQUENCE, FILE TRANSLATE, AND COMPILE-TIME TABLE/ARRAY COMPRESSIONS

At compile time, these three tables are compressed by phase $RPFA into two compression types. The alternate collating sequence and file translate tables form one compression and the compile-time tables/arrays form another compression. The first byte of each 97-byte compression contains the length of the compression; the remaining bytes contain the record specified. The compression block table (CZATAB) tells the type of compression and its address.

CHAIN TABLE

This table is built by phases $RPPN and $RPRW and used by phase $RPMP. The table begins with a 2-byte field which contains the number of entries in the table and ends with a 4-byte dummy entry to show the last byte of the subsegment. Each entry in the table is four bytes long:

Byte	*Contents*
0	File sequence number
1-2	Start address of the subsegment
3	X'08' = Set Resulting Indicators subroutine X'10' = TAG subroutine X'20' = Convert to Decimal subroutine X'80' = Unpack subroutine

COMPILE-TIME SYMBOL TABLE

The 17-byte compile-time symbol table is built by phase $RPGY for each compile-time table/array:

Byte	*Bit*	*Contents*
0-5		Table/array name
6-7		Number of elements in the table/array
8-9		Number of elements per input record
10-11		Element length
12	0	1 = Look-ahead field
	2-3	00 = Field name 01 = Array name 10 = Table name
	4-5	00 = Table sequence not specified 01 = Table is descending or field name UDATE related field 10 = Table is ascending or, if RLABL is specified, field name or mask and displacement of the indicator
	6	0 = Length is alphameric 1 = Length is numeric
13-14		First element address (rightmost byte)
15-16		DTT address

DATA MANAGEMENT ENTRY POINTS AND MODULE NAMES COMPRESSIONS

Phase $RPFA builds compressions of data management entry points and module names. These compressions are used by later phases to generate branches to data management. The first byte of each compression is the length of the compression. This is followed by a series of 2-byte internal entry points and 2-byte translated module names. The last two bytes of each compression contains X'EEEE'.

The 2-byte entry point contains the following:

Byte	*Contents*
0 (Bits 0-3)	1000 = Sequential file processed consecutively 1001 = Indexed file processed randomly by key 1010 = Direct (disk address) file 1011 = Direct (record number) file 1100 = Indexed file processed sequentially by key 1101 = Indexed file processed sequentially within limits
0 (Bits 4-7)	0001 = Input file 0010 = Output file 0011 = Update/combined file 0100 = Add/print file 0101 = Input + add file 0110 = Output + add file 0111 = Update/combined + add file 1000 = Output unordered file 1010 = Output chain file

1 (Model 6)

X'18' = CONSOLE (printer-keyboard)
X'14' = KEYBOARD (keyboard)
X'40' = DISKET
X'80' = BSCA
X'84' = BSCA (BSCA with first-time logic)
X'94' = CRT (display station)
X'A0' = DISK (disk unit)
X'A1' = Multivolume Disk
X'E4' = TRACTR1 (tractor 1)
X'E4' = TRACTR2 (tractor 2)
X'E8' = LEDGER (ledger card device)
X'F4' = DATA96 (data recorder)
X'FF' = Invalid entry

1 (Model 10)

X'10' = CONSOLE (printer-keyboard)
X'40' = DISKET
X'50' = READ42 (1442)
X'60' = TAPE
X'80' = BSCA
X'84' = BSCA with first-time logic
X'A0' = DISK (5444)
X'A1' = Multivolume DISK (5444)
X'C0' = DISK45
X'C1' = Multivolume DISK45
X'E0' = PRINTER (line printer-carriage 1)
X'E1' = PRINTR2 (line printer-carriage 2)
X'F0' = MFCU1 (primary hopper)
X'F0' = MFCU2 (secondary hopper)

2 (Model 12)

X'10' = Console
X'40' = DISKET
X'50' = READ42 (1442 Card Read-Punch)
X'5F' = Special
X'60' = Tape
X'80' = BSCA
X'84' = BSCA (first time logic)
X'A0' = DISK (5444 simulation area)
X'C0' = DISK40 or DISK45 (3340 main data area)
X'E0' = PRINTER – carriage 1
X'E8' = PRINTER – carriage 2
X'F0' = MFCU1 – primary hopper
X'F8' = MFCU2 – secondary hopper
X'FF' = Invalid entry

1 (Model 15)

X'01' = Device independent input
X'02' = Device independent output
X'1C' = CRT77
X'40' = DISKET
X'50' = READ42
X'58' = READ01
X'60' = TAPE
X'80' = BSCA
X'84' = BSCA with first-time logic
X'A0' = DISK (5444)
X'A1' = Multivolume DISK (5444)
X'C0' = DISK45, DISK40
X'C1' = Multivolume DISK45
X'E0' = PRINTER
X'E4' = PRINT84
X'F0' = MFCU1
X'F0' = MFCU2
X'FC' = MFCM
X'FF' = Invalid Entry

External module names are formed from a 16-character EBCDIC alphabet. Each character is translated into a hexadecimal equivalent when the module name is compressed. In this way, a 4-byte module name is compressed into two bytes. The EBCDIC characters used in module names and the corresponding internal hexadecimal characters are as follows:

EBCDIC Character in Module Name	*Corresponding Internal Hexadecimal Character*
A	0
B	1
C	2
D	3
F	4
G	5
H	6
I	7
L	8
M	9
O	A
P	B
R	C
S	D
T	E
U	F

On Model 15 Program Number 5704-RG2, when external buffers are called, different data management modules are called for disk devices.

The external buffer data management modules have the same names as their corresponding internal buffer data management modules except for the first alphabetic character. (The name is changed in phase $RPRZ.) If external buffers are specified, disk data management modules with the first character of 'C' are changed to 'W', a first character of 'D' is changed to 'Y', and a first character of 'I' is changed to 'X'. Until phase $RPRZ is run, the translated name of the equivalent data management module is carried in the compression.

Example:

Disk file, external buffers specified
Internal buffers data management module name:
$$DFIM
Translated name in compression: X'3479'
Changed data management module name in $RPRZ:
$$YFIM

ERROR FILE

The error file can contain two different types of error information. During the Input and Compression phases, the error file is 64 bytes long. Each bit, going from left to right, represents an error number. When a bit is on, an error message must be printed for the corresponding error number. For example, if the eighth bit is on, then error message 8 must be printed. See the *IBM System/3 Model 6 RPG II Reference Manual*, SC21-7517 or *IBM System/3 RPG II Reference Manual*, SC21-7504 for a list of the errors and the error messages.

During the Diagnostic phases, the error file builds a 5-byte entry for each error found. The entry is in this format:

Byte	*Meaning*
0	Length of entry
1-2	Statement number of error
3-4	Binary number of error number

The compression block table (CZATAB) gives the address of the error file.

FILE INPUT/OUTPUT TABLE

This table is created by phase $RPGH and remains in storage to be used by the remaining Assign phases.

There is one 33-byte entry in the table for each file in the order defined on the file description specifications except the primary file entry is first. The format of this table is:

Byte	*Bit*	*Contents*
0-7		Symbolic filename
8	0	1 = Look-ahead field
	1	1 = *PRINT or * in column 40 (Models 10 and 12)
	2	1 = *PRINT used (Models 10 and 12)
	3	1 = From Filename with address placed in COMMON at COMBKC
	4	1 = To Filename with address placed in COMMON at COMBLC
	5	1 = Line counter specifications for this file
	6	1 = Printer type file or BSCA printer file
9		Sequence number of file description compression in which filename is defined
10-11		IOCB address
12		Device type
13	0-3	0001 = Primary file 0010 = Secondary file 1000 = Chain file 1001 = Demand file 1010 = Record address file 1100 = Table
	4-5	00 = No sequence 10 = Ascending file 01 = Descending file
	6	1 = End of file (col 17)
	7	1 = File translate
14	0-1	00 = Sequential file 01 = Indexed file 10 = Direct file 11 = ADDROUT file
	2-3	00 = Consecutive processing 01 = Random processing 10 = Indexed file processed sequentially by key 11 = Indexed file processed sequentially within limits
	4-6	110 = Input file 100 = Combined file 011 = Update file 101 = Ordered output specified 111 = Unordered output specified 001 = Display file
	7	1 = Add records
15	0	0 = Variable record 1 = Fixed record
	1	0 = Unblocked file 1 = Blocked file
	2-3	01 = Extension code specified (col 39) 10 = External indicator used to condition file
	4-6	000 = Consecutive processing 010 = Key (alphameric) 011 = Packed key 100 = By ADDROUT file 110 = By relative record number
	7	1 = Dual I/O

Byte	*Bit*	*Contents*
16-17		Block length
18-19		Record length
20-21		Key start location
22		Key length
23		Overflow indicator mask
24		External indicator
25-26		Index in storage
27		Number of extents
28		Operation code and parameters
29		DTF length
30-31		DTF address
32		Unreferenced

FILENAME TABLE

This table is built by phase $RPGV and used by phase $RPJK. The table contains one 12-byte entry for each filename specified in this format:

Byte	*Bit*	*Contents*
0-7		Filename
8-9		Record length
10-11	0	1 = Filename not used
	1	1 = Extension specified but not found
	2	1 = Line counter specified but not found
	3	1 = File sequence specified
	4	1 = Match fields
	5-6	00 = Blank 10 = Chain file 01 = Display file 11 = Demand file
	7	1 = Input specifications not found for input file
	8	1 = Output specifications not found for combined or update file or add was specified.
	9	1 = Ledger card device
	10	TRACTR1 or TRACTR2
	11-15	Statement number

FINAL SEGMENT LIST (MODELS 6, 10, AND 12)

$RPSB takes the 16-byte segment list from $RPSP in the following format:

Byte	*Bit*	*Contents*
0	0-1	Type 00 = Subroutine 01 = Table 10 = Mainline segment 11 = Subsegment
	2	Segment is subsegmented or first of a substructure list if Type = 10
	3	User subroutine if Type = 00
	4-6	100 = Segment is main overlay or start of suboverlay 110 = Segment is contained in main overlay 101 = Segment is contained in sub-overlay
	7	Overlay uses both areas (on mainline only)
1		Overlay priority (X'FF' = duplicate segment)
2-3		Address of object code
4-5		Length of object code
6-7		Substructure pointer
8-9		Identifier for each segment
10-11		Duplicate chain
12-13		Volatile duplicate flag
14-15		Transfer vector size

Phase $RPSC creates a 10-byte segment list by taking the first 10-bytes of the final segment list. Bytes 6-7 are changed to contain the relocated address of object storage.

FINAL SEGMENT LIST (MODEL 15)

$RPRX builds the 16-byte segment list and $RPRZ adds to it. The format of 16-byte segment list is:

Byte	*Bit*	*Contents*
0		Type 00 = System subroutine 10 = User subroutine 20 = EXTRN to system subroutine 30 = EXTRN to user subroutine 40 = Constants 80 = RPG mainline A0 = EXTRN to ENTRY C0 = RPG subsegment
1		Priority or type of entry segment for an EXTRN if TYPE = A0, 30, or 20.

Byte	Bit	Contents
2-3		Address of object code or: – Entry point address if TYPE = A0, 30, or 20. – X'0000' if TYPE = 10 or 00.
4-5		Length of object code or: – ID of referenced segment if TYPE = A0, 30, or 20. – X'0000' if TYPE = 10 or 00.
6-7		Controlling identifier (a pointer which chains entries together), or ID if referencing segment if TYPE = A0, 30, or 20.
8-9		Identifier for the segment or X'0000' if TYPE = A0, 30, or 20.
10-13		Reserved
14-15		Name: – Compressed name from 16-character alphabet if TYPE = 00 or 20. – Last two characters of user subroutine name if TYPE = 10 or 30. – End + 1 address of referenced segment if TYPE = A0 and reference segment TYPE = 40. – End + 1 address of this segment if TYPE = 40, 80, or C0.

GENERAL STORAGE TABLE

The general storage table is built by phase $RPLN and contains constants, edit patterns, literals, and output DTF parameters. Phase $RPLR modifies and uses this table. The format of the general storage table is:

Byte	Bit	Contents
0		Length of table element
1-2		Statement number
3		Usage mask
	0	1 = Total calculations
	1	1 = Detail calculations
	2	1 = Program Close Mainline
	3	1 = LR and Overflow Control Mainline
	4	1 = Open Mainline
	6	1 = Total output
	7	1 = Detail output
4-5		Object code address (built by phase $RPLR)
6-29		Object code

INTERNAL SYMBOL TABLE (MODELS 6, 10, AND 12)

The internal symbol table built by phase $RPSE is used by phases $RPSG, $RPSI, and $RPSK. The 3-byte table format is:

Byte	Bit	Contents
0-1		Disk address of the first sector of sorted object code blocks
2		Number of sectors of sorted object code blocks
3	0	1 = Object code sorted
	1	1 = Mainline routine is an overlay
	2	1 = Mainline routine contains suboverlays
	3	1 = Mainline processed by final output phases
	4	0 = Root segment to be generated 1 = Overlays to be generated
	5	1 = Mainline completely in storage
	6	Not used
	7	1 = Number of sectors of sorted object code blocks exceeds 225

NAME TABLE

Phase $RPGU builds a 16-byte entry for each object-time table/array in the name table with this format:

Byte	*Bit*	*Contents*
0-5		Table/array name
6	0-3	Number of decimal positions for numeric fields
	4-7	Length minus one in binary of a table/array entry
7	0	1 = Look-ahead field
	2-3	00 = Field name 01 = Array name 10 = Table name
	4-5	00 = Table sequence not specified 01 = Table is descending or field name is UDATE related field 10 = Table is ascending or, if RLABL is specified, field name or mask and displacement of the indicator
	6	0 = Alphameric length minus one 1 = Number of decimal positions in zone portion; length minus one in numeric portion
	7	Not used
8-9		Address of first entry in the table/array
10-11		Address of second byte of the DTT
12-13		Number of entries in the table/array in binary
14-15		Number of statement in which the table/array is first defined

OBJECT CODE BLOCK

Object code blocks contain portions of sequenced object text generated by the Assign and Assemble phases. These blocks are written onto a disk work file ($SOURCE) during compilation. The overlay phases sort the object code blocks and generate text-RLD records required by the linkage editor. Object code blocks contain up to 255 bytes, in the following format:

Byte	*Contents*
0	Length of the object code block
1	Length of object text contained in the block
2-3	Address of the leftmost byte of the text
4-n	Object text
n+1	Length of the relocation dictionary (RLD). This entry is zero if no RLD is present
n+2-end	Relocation dictionary (RLD)

Each relocation dictionary entry contains two or four bytes, as follows:

Byte	*Contents*
0	X'01' = Transfer vector not allowed X'04' = Overlay code X'0C' = Branch to transfer vector X'30' = RLD address is in ROCA X'40' = Relocate address outside of current segment X'80' = External reference (RPG II code) (if on, Bytes 2 and 3 are present) X'90' = External reference (subroutine) X'C0' = External reference (user subroutine)
1	Displacement from the beginning of the text to the rightmost byte of address to be relocated
2-3	Translated external module name (see *Data Management Entry Points and Module Names Compressions*)

PHASE LOAD COMPRESSION

Phase $RPG builds 11-byte compressions of load information for each compiler phase. These compressions are used to eliminate the find every time a new phase is called. The format of these compressions is:

Byte	*Meaning*
0	Length of entry (constant X'0B')
1-2	Last two characters of the phase name
3-4	Cylinder/sector (location of phase on disk)
5	Number of disk sectors to be loaded
6-7	Load point
8	Displacement to first RLD
9-10	Entry point

SEGMENT LIST

This list is built by phases $RPLR, $RPMK, $RPMM, and $RPMP and is used by phases $RPRX and $RPSA (Models 6, 10, and 12). The segment list contains information about segments of code in this format:

Byte	*Contents*
0	X'F0' = EXIT subroutine specified X'C0' = Subroutine X'B0' = IPCR,OPCR, or constants X'A0' = Object code X'80' = Mainline code
1	Overlay priority with 0 = lowest priority to be overlaid (the last to be overlaid)
2-3	Address of object code or subroutine name
4-5	Length of object code
6-7	Controlling identifier (a pointer which chains entries together for easy identification by the Overlay phases)
8-9	Identifier for each segment

There is an entry for each mainline and an entry for each subsegment. If the subsegment belongs to more than one mainline, an entry is generated for each mainline to which the subsegment belongs.

SYMBOL TABLE

The symbol table built by phase $RPHA remains to be used by phases $RPHC and $RPHD. There is one 12-byte entry in the table for each field name in the order defined in the input and calculations specifications. The table format is:

Byte	*Bit*	*Contents*
0-5		Field name
6	0-3	Number of decimal positions for numeric field
	4-7	Field length minus one in binary
7	0	1 = Look-ahead field
	1	1 = Field unreferenced
	2-3	00 = Field name
	5	1 = UDATE related field
	6	0 = Alphameric length minus one 1 = Number of decimal positions in zone portion; length minus one in numeric portion
	7	Not used
8-9		Assigned address in the Root Segment
10-11		Number of the statement in which the field is first defined

TELECOMMUNICATIONS TABLE

Phase $RPGK builds a 33-byte telecommunications table in main storage behind the file input/output table for each telecommunications compression. For more information on telecommunications compressions, see Figure 3-8. The table format is:

Byte	*Contents*
0-2	Format bytes
3-7	Note bytes
8-9	Wait time
10-11	Permanent error indicator

Byte	*Contents*
12-13	Record available indicator
14-15	Address of the leftmost byte of this station ID entry
16	Number of bytes in this station ID entry
17-18	Address of the leftmost byte of remote station ID entry
19	Number of bytes of remote station ID entry
20-21	Address of leftmost byte of Dial Number entry
22	Number of bytes of Dial Number entry
23-24	Address or polling characters
25-26	Address of corresponding file input/output table entry
27	Remote device selected
28	Lines to space after
29	Lines to space before
30	Line count
31	Page size
32	Overflow line

TEXT-RLD RECORD

Models 6, 10, and 12

Text-RLD records are generated by overlay phase $RPSK from sorted object code blocks for use by the linkage editor. Each record is 64 bytes long in the following format:

Byte	*Contents*
0	T (denotes text-RLD record)
1	Length minus 1 of object text contained in the record
2-3	Address of the rightmost byte of object text in the record
4-63	Object text begins in byte 4; 1-byte RLD (relocation dictionary) entries are inserted beginning in byte 63 from right to left. Unused bytes (at least one) between text and RLD contain X'00'. RLD points to the right end of the address displaced from beginning of text.

Each RLD entry contains the displacement from the leftmost text byte in the record.

Model 15

See *IBM Model 15 System Services Logic Manual*, SY21-0034, for text-RLD record format for the Model 15.

DISK WORK AREAS

The RPG II Compiler requires that two disk work areas be provided, $WORK and $SOURCE. Figures 3-9 and 3-10 show the general usage of these areas. For more detailed information on any phases mentioned, see the description of these phases elsewhere in this manual.

Initiating Phase	Non-Overlay		Overlay	
	$WORK	$SOURCE	$WORK	$SOURCE
$$STAI*	Unused	Source data	Unused	Source data
$RPEA	Compressions		Compressions	
$RPGX		Object text blocks		Object text blocks
$RPRX	Initializes 16-byte long final segment list		Initializes 16-byte long final segment list	
$RPSC	Converts 16-byte final segment list to 10-byte (full) segment list		Converts 16-byte final segment list to 10-byte (full) segment list	
$RPSE	16-byte final segment list and 10-byte (full) segment list Puts out 10-byte segment list and sorted text for root phase Puts out 10-byte segment list and sorted text for each mainline segment		16-byte final segment list and 10-byte (full) segment list Puts out 10-byte segment list and sorted text for root phase Puts out 10-byte segment list and sorted text for each mainline segment	
$RPSG (called only for overlay)				10-byte segment list 10-byte segment list and sorted text for root phase (updated with overlay fetch routine, overlay fetch table, and transfer vectors) 10-byte segment list and sorted text for each mainline segment (updated with transfer vectors)
$RPSI**		Phase and Entry records	Phase and Entry records	
$RPSK**		Text-RLD records	Text-RLD records	
$RPSK (just prior to calling $LINKB)	Copy of $SOURCE (Phase, Entry, Text-RLD records)			

**Called once for every overlay generated

Figure 3-9. Use of $WORK and $SOURCE (Models 6, 10, and 12)

Initiating Phase	$WORK	$SOURCE
$RPEA	Compressions	Compressions
$RPGX		Object text blocks
$RPRX	Initializes 16-byte long final segment list.	
$RPRX	16-byte segment list	Final text block
$RPRY		Sorted and merged text blocks; Copy of 16-byte segment list
$RPRZ (just prior to calling $OLYNX)	Input to Overlay Linkage Editor: OPTNS–OPTIONS RECORD S–ESL RECORDS T–TEXT-RLD RECORDS S– T– R. Modules S– T– E–END RECORD	

Note: Usage of disk work areas is the same in Model 15 RPG II, whether or not overlays are used.

Figure 3-10. Use of $WORK and $SOURCE (Model 15)

This section describes the object program generated by the RPG II Compiler in terms of:

- RPG II processing cycle (control flow through object program)
- Object code generated for calculations
- Functions performed by subroutines
- Data areas used by the object program
- Overlay techniques employed by the RPG II Compiler

This section also includes sample dump analysis as an aid in examining areas of a storage dump of an RPG II program.

Flowchart Techniques

Appendix A in this publication explains the general flowcharting techniques used. The flowcharting conventions in this section differ from the general flowcharting techniques in these ways:

1. A striped process block signifies an operation performed by a routine which is flowcharted in this section.
2. A decision block labeled COMPILER**** signifies a compiler decision. THIS DECISION IS NOT ACTUALLY MADE IN THE OBJECT PROGRAM, BUT IT INDICATES THE CONDITIONAL PRESENCE OF SOME OBJECT CODE. A compiler decision indicates a choice of two or more alternatives, only one of which is present in a given section of object code.

The narrative description of the routines only supplements the flowcharts.

OVERALL OBJECT PROGRAM FLOW

Figure 4-1 shows the flow of the RPG II processing cycle.

Figure 4-2 shows the cycle in more detail.

DETAILED OBJECT PROGRAM FLOW

The object program generated by the RPG II Compiler is described in detail on Charts CA through CM. Parameter lists and local hold areas are described in the individual routine descriptions. Significant data areas are defined in *Data Areas* in this section.

Open Mainline (Chart CA)

The Open Mainline performs key functions which control the RPG II cycle. It opens files and initializes data areas, indicators, and switches to allow a new processing cycle to begin.

Input Processing Control (Chart CB)

The Input Processing Control routine (IPCR) handles processing of input between RPG II and data management. IPCR also handles some error recovery from data management.

Output Processing Control (Chart CC)

The Output Processing Control routine (OPCR) handles processing of output between RPG II and data management. OPCR also handles some error recovery from data management. For each file that has output, there is a 12-byte linkage in the program listing.

Output Fields and Records Code (Chart CD)

The Output Field and Records Code routine move fields to the output buffer in main storage.

Output Records

The object code generated by compile-time phase RPPS is divided into six overlay segments. These segments appear in different areas of the program logic flow. The segments in order of appearance are:

1. First Page Output routine

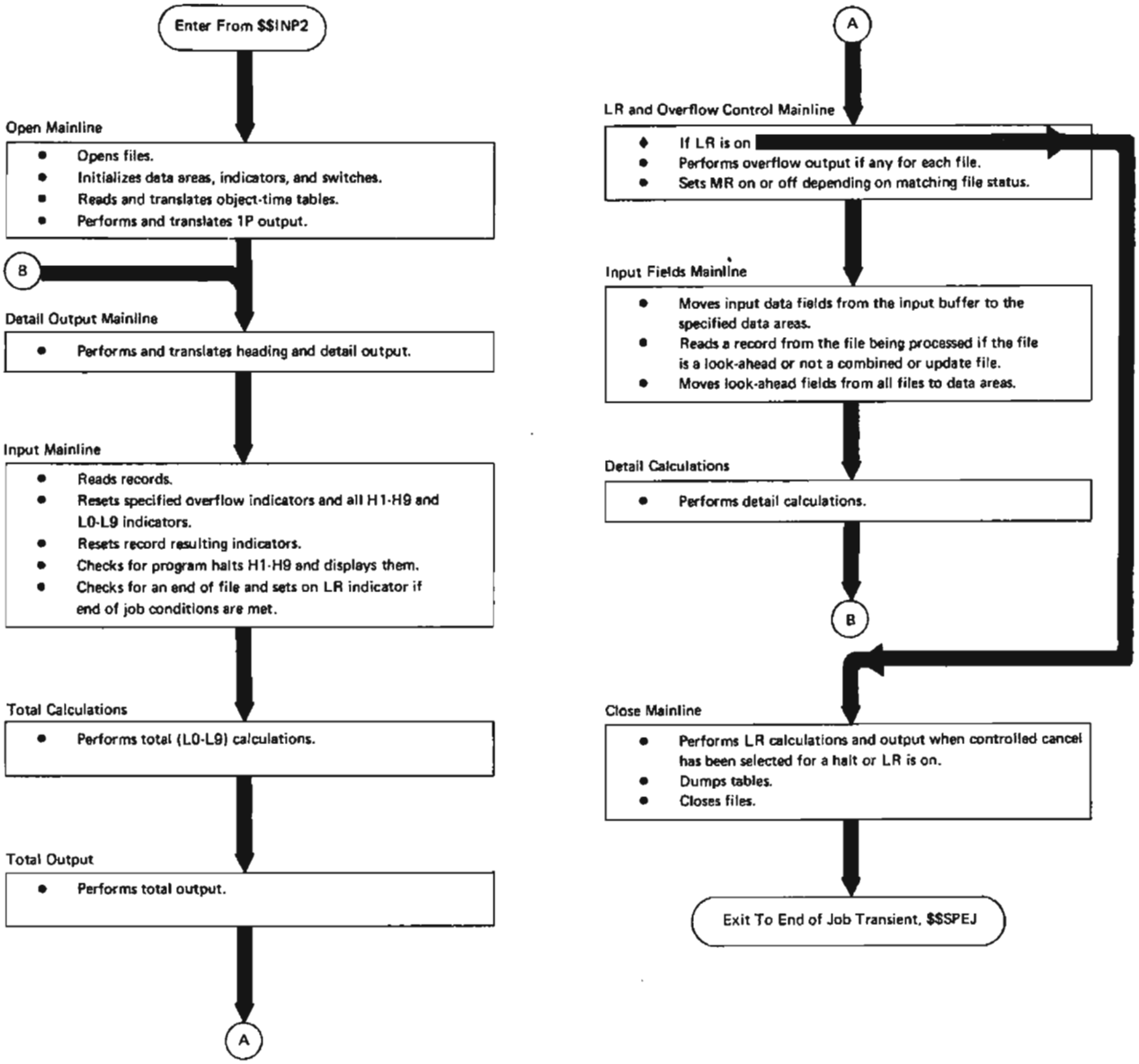


Figure 4-1. Object Program Flow

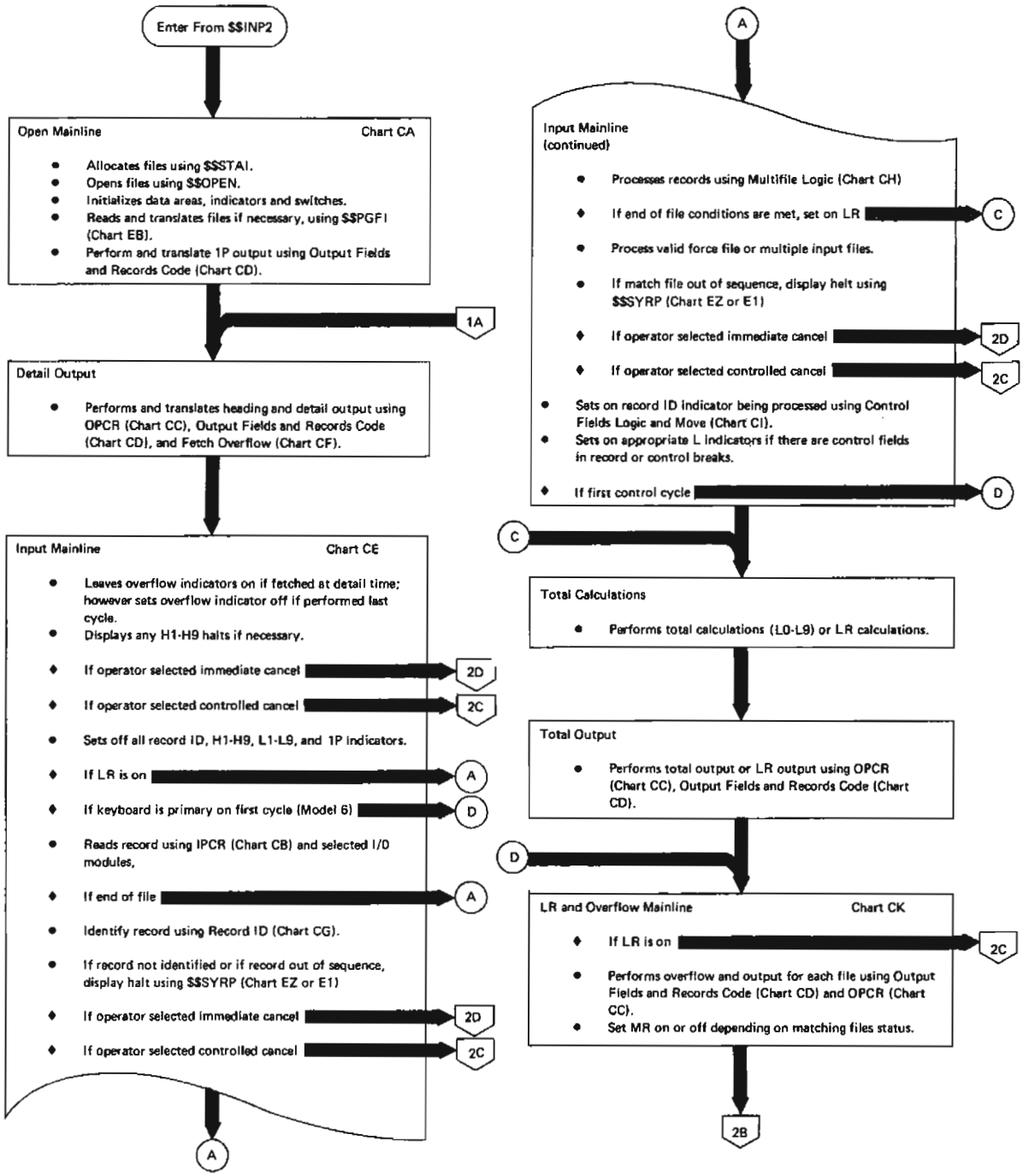


Figure 4-2 (Part 1 of 2). Intermediate Object Program Flow

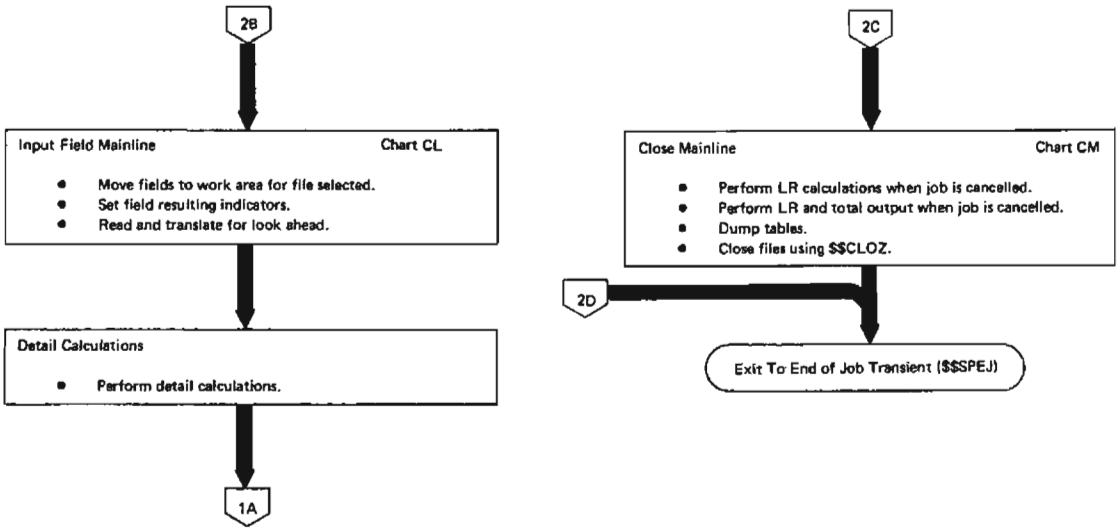


Figure 4-2. Intermediate Object Program Flow (Part 2 of 2)

2. Heading and Detail Output routine

3. Total Output routine

4. LR Output routine

5. Exception Output routine

6. Overflow Output routine (one segment per file).

The linkages between the segments are generated as follows. The First Page Output routine branches to the Detail Output routine, and the Detail Output routine branches to Input Mainline. The LR Output routine, the Exception Output routine, and each segment of the Overflow Output routine are closed subroutines which store ARR and return branches.

Record Indicators: Overflow indicators are not tested in the Overflow Output routine. This is done before the routine is called. LR and IP indicators are not tested in the First Page Output routine or the LR Output routine since first page and last record processing occurs once.

Output Fields

The addresses of output fields may be found as follows:

Table Elements: The last referenced element (indicated in the DTT) contains the addresses.

Array with No Index: Array loop control is used to increment processing throughout the array.

Array with Variable Index: The Array Index subroutine is used to find the desired field (see the *Library of Subroutines* for a description of the array index).

Array with Numeric Index: The address is calculated at compile time and used directly.

Punch: Punch fields are all moved and execute first before all fields to be printed.

**PLACE:* The highest previously used end position on other than *PLACE lines is used for the move. The length of the move is from the highest previously given end position to position 1.

PAGE: If no conditioning indicators are given, 1 is added to the field contents before the field is moved to the output buffer. If conditioning indicators are given and the conditions are met, the field is set to zero, 1 is added to it, and the field is moved to the output buffer. If conditioning indicators are given and the conditions are not met, 1 is added to the field and the field is moved to the output buffer. For 1P forms alignment, the PAGE fields are not incremented on the first 1P line.

Editing: For edit words and edit codes:

1. Edit patterns are moved to the output buffer.

2. The field is edited into the pattern.

3. If the field is positive, the status is blanked out for edit words.

4. For edit codes, if zero suppression is specified, then zeros suppress everything to the left of the decimal point. If the field has no value, it is zero suppressed through the decimal point on edit codes 2, 4, B, D, K, and M.

For Blank After: Alphameric fields are cleared to blanks; numeric fields are cleared to zeros.

Update FILES: An image of the update files is moved from the input buffer to the output buffer before output fields are moved in, except when adding to an update file. An image of the update files will not be moved to the output buffer in this case.

Input Mainline (Chart CE)

The Input Mainline routine generates code to get a record and checks for end of job. In addition, it performs these indicator functions:

1. Resets specified overflow indicators and all H1-H9 and L0-L9 indicators.

2. Resets record identifying indicators.

3. Checks for program halts (H1-H9) and displays them.

4. Checks for an LR indicator and, if LR is found, control is passed to Control Fields Logic and Move routine.

Fetch Overflow (Chart CF)

The Fetch Overflow routine determines if overflow has occurred for the file being processed. If overflow has occurred and is still pending, the overflow segment for the file being processed is called and the second internal indicator is set on. There is a fetch overflow routine for each overflow mainline.

Record ID (Chart CG)

The Record ID routine identifies a record, moves control parameters into the IOCB, and sequence checks the numeric sequence for all record types.

This routine also checks column 17 (Number) and column 18 (Option) of Input Specifications.

Multifile and Matching Records Logic (Chart CH)

Chart CH describes the three types of multifile and matching record logic: multifiles with no matching fields, multifiles with matching fields, and one input file with matching records.

Four areas are described as work and save areas. H2 is a work area located at the beginning of ROCA. H1 is a storage area which is assigned to ROCA if H1 is not greater than 72 bytes. If it is greater than 72 bytes, H1 is generated as part of the matching records code. S1 is a save area assigned to the Root Segment containing the matching values of the last selected file. It is used for sequence checking. S2 is a save area assigned to the Root Segment containing the match values of the last selected primary file. It is used to control the setting of the MR indicator.

The address of the primary file IOCB is stored at displacement X'98'-X'99' in ROCA, and the address of the last selected IOCB is stored in ROCA at displacement X'9A'-X'9B'. See *Data Areas, Match Field Save Areas* in this section for further discussion.

Control Fields Logic and Move (Chart CI)

The Control Fields Logic and Move routine determines whether to bypass total output and total calculations for the first RPG II cycle or bypass total output and total calculations when the first control break occurs. The routine also moves the control fields to a work area and compares them against the last set of control fields. If there is a control break, the appropriate level indicators and all lower L indicators are set on.

Chain and Read (Chart CJ)

The Chain and Read routine performs three functions for each file:

1. Identifies the input record.
2. Moves record control parameters to the IOCB.
3. Moves data fields from the data area to the output buffer.

LR and Overflow Control Mainline (Chart CK)

The LR and Overflow Control Mainline performs four functions:

1. Calls the Program Close Mainline if the LR indicator is on.
2. Tests for overflow and, if on, calls the appropriate Overflow routine.
3. Sets off all internal overflow indicators.
4. Sets the MR indicators.

Move Input Fields Mainline (Chart CL)

The Move Input Fields Mainline performs these major functions:

1. Moves input data fields from the input buffer to the specified data areas.
2. Reads a record from the file being processed if the file is a look-ahead and not a combined or update file.
3. Moves look-ahead fields from all files to data areas.

Object code generated to perform these functions is determined by the type of field:

Field Type	*Code Generated*
Alphameric	MVC instruction
Numeric	ZAZ instruction

Field Type	*Code Generated*
Pack	Linkage to unpack subroutine
Binary	Convert to Decimal subroutine
Alphameric array	MVC instruction for each 256 bytes of data
Numeric array	ZAZ instruction and loop control

Program Close Mainline (Chart CM)

The Program Close Mainline performs the necessary RPG II functions for end of job by processing LR calculations and LR output. It also closes files and dumps tables.

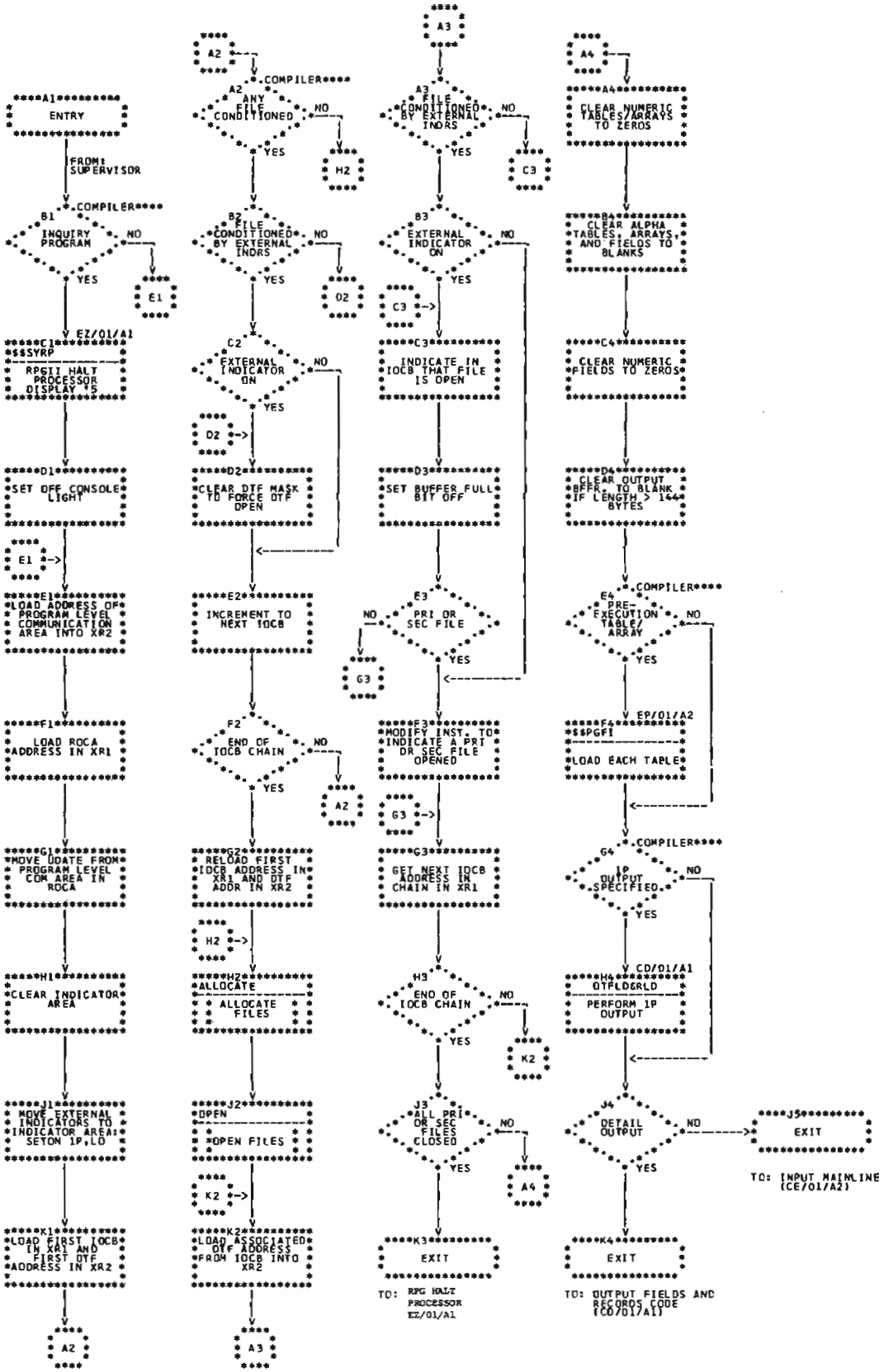


Chart CA. Open Mainline

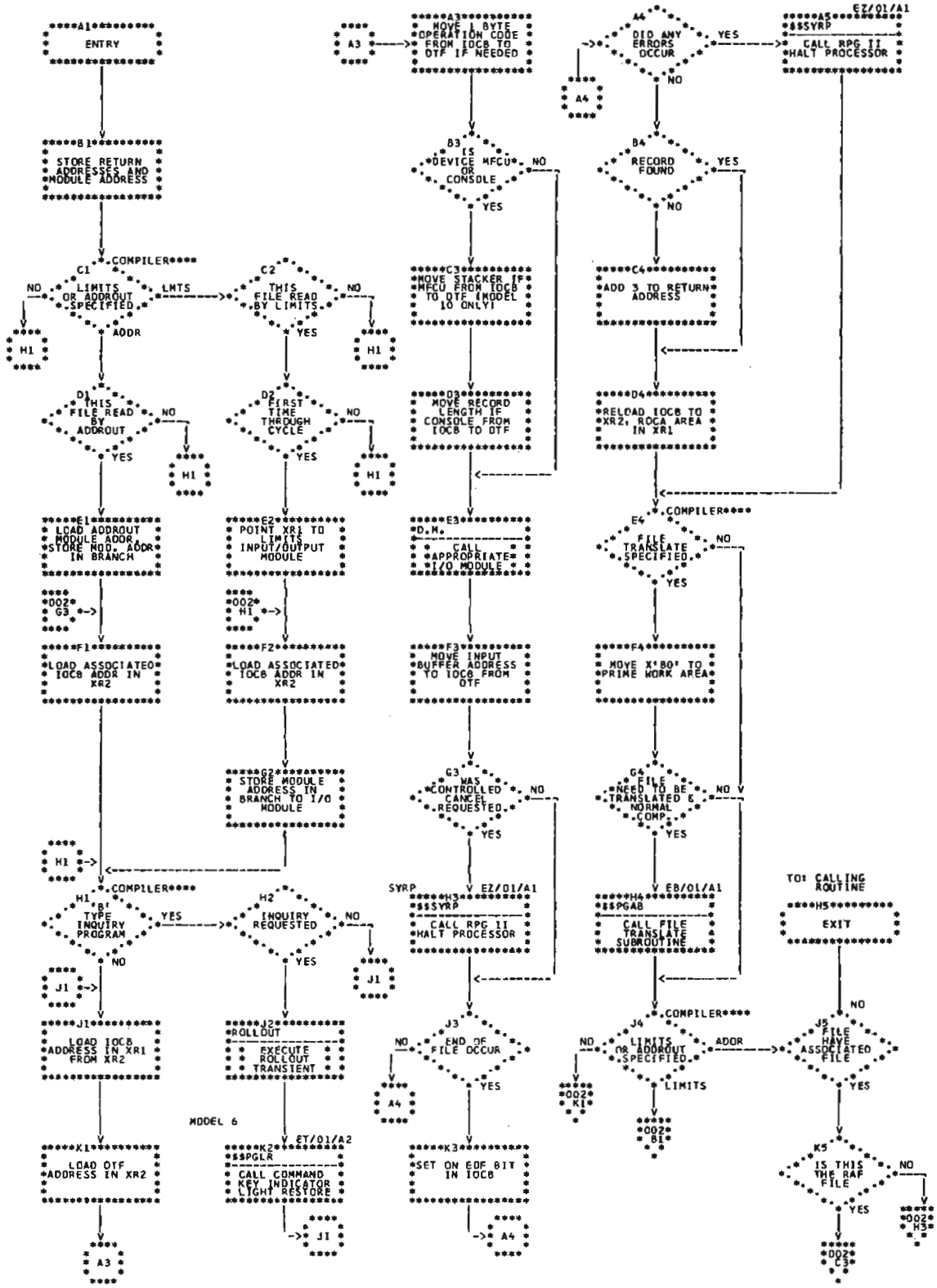


Chart CB (Part 1 of 2). Input Processing Control Routine

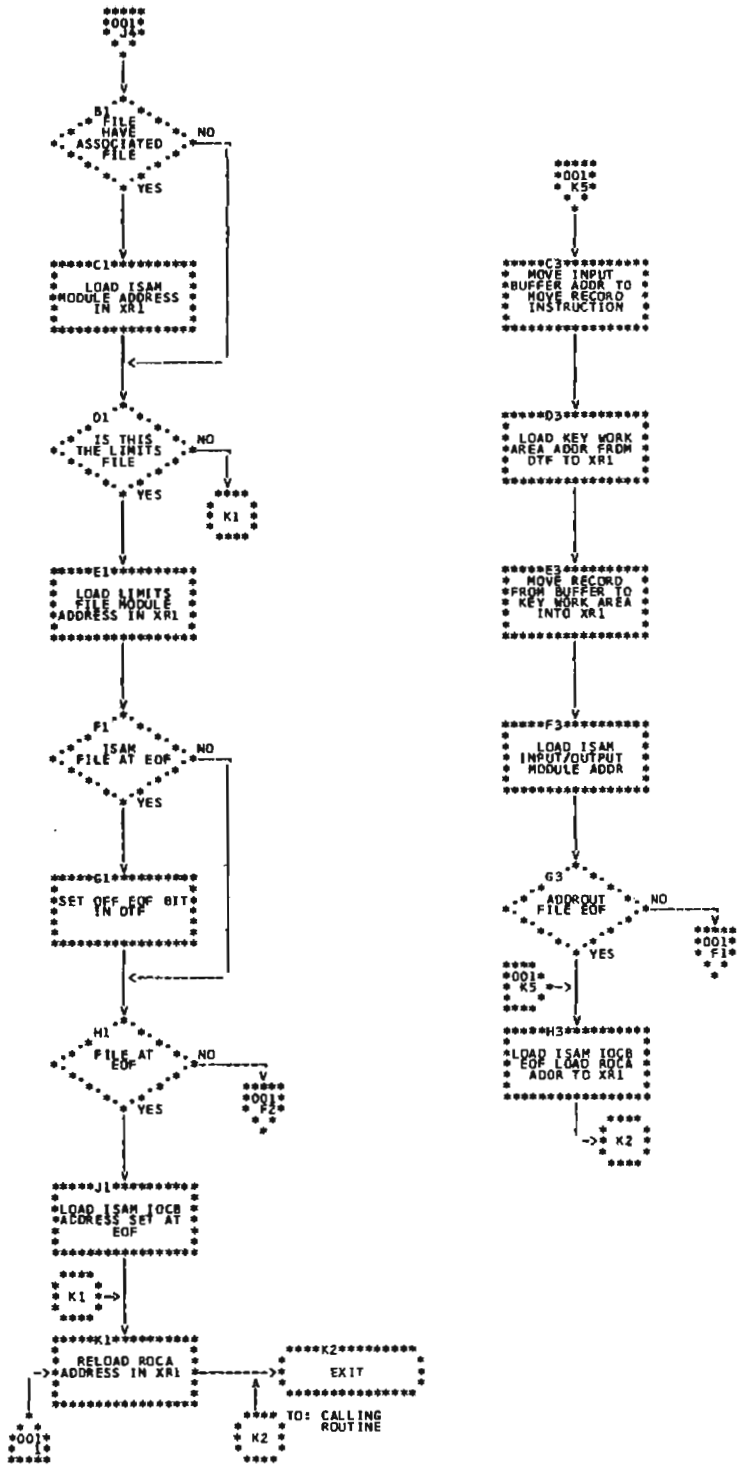


Chart CB (Part 2 of 2). Input Processing Control Routine

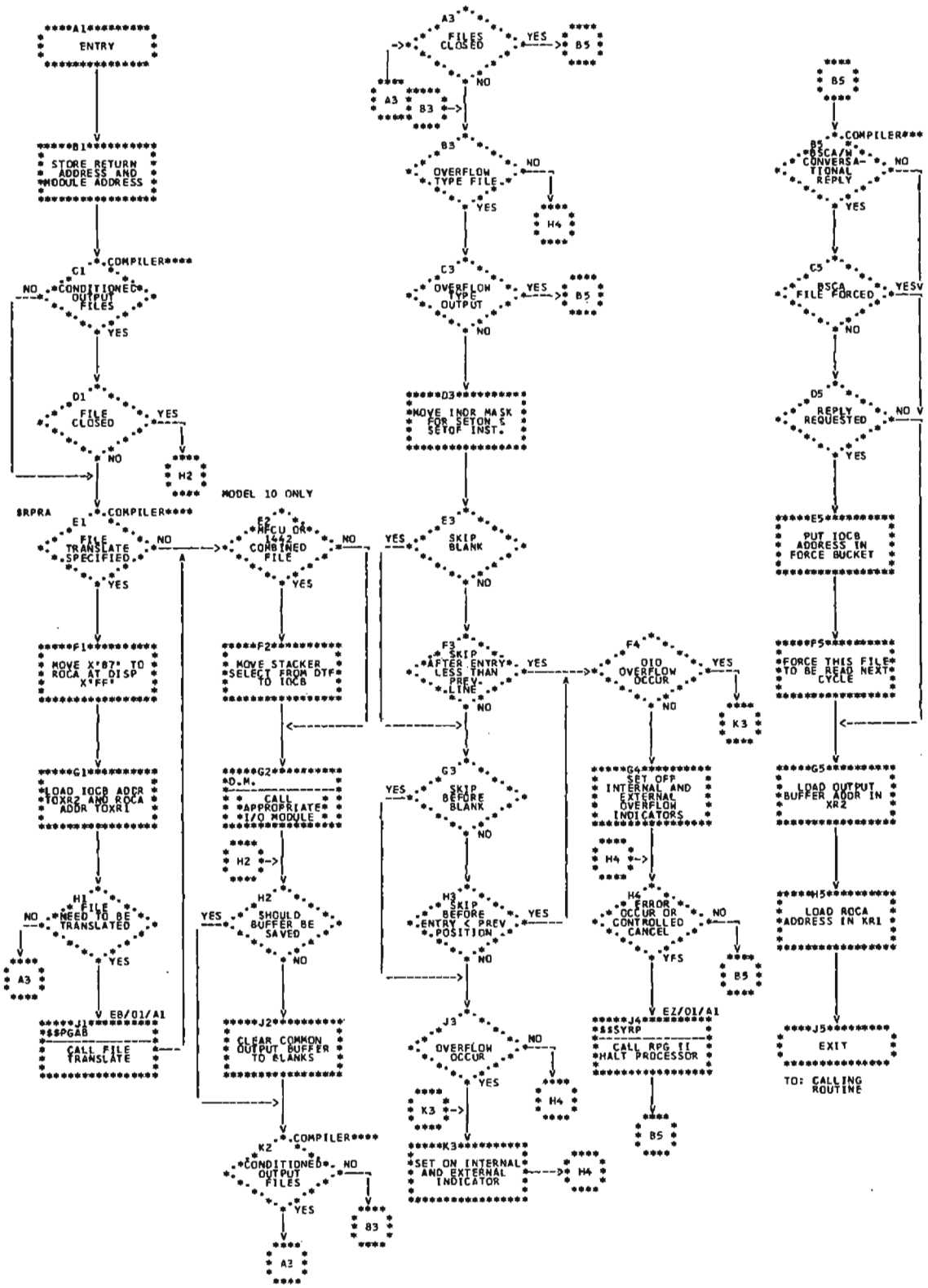


Chart CC. Output Processing Control Routine

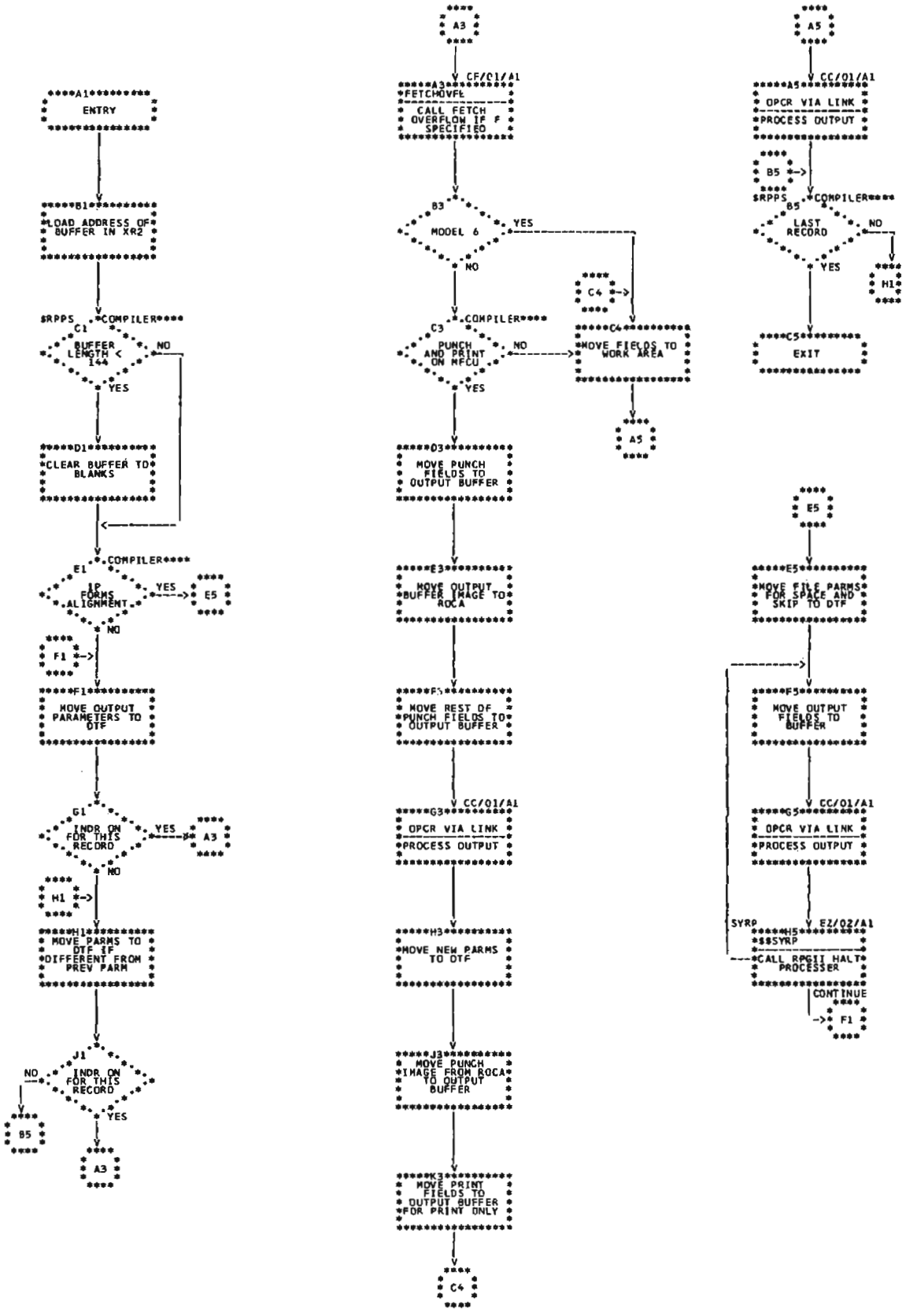


Chart CD. Output Fields and Records Code

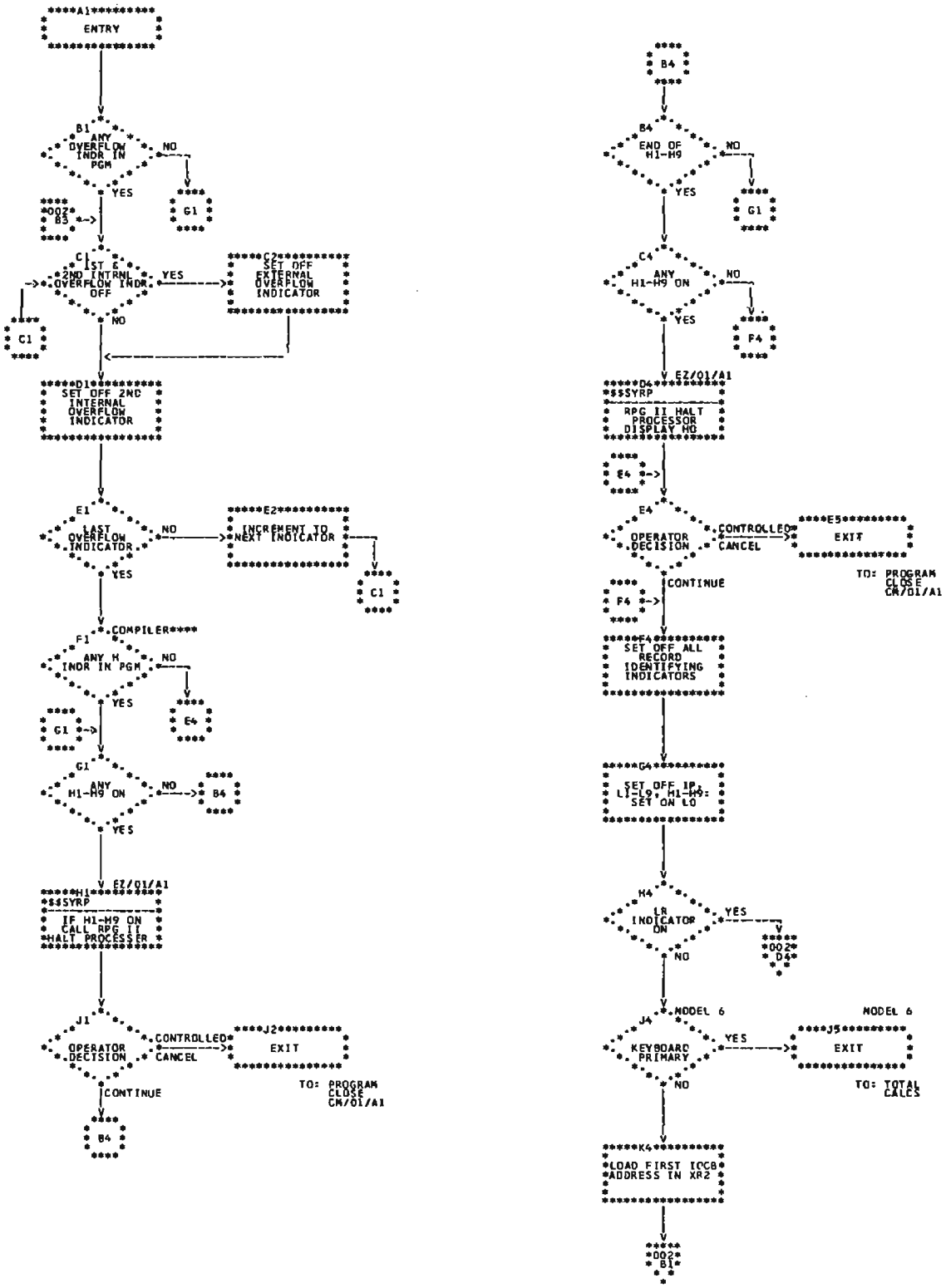


Chart CE (Part 1 of 2). Input Mainline

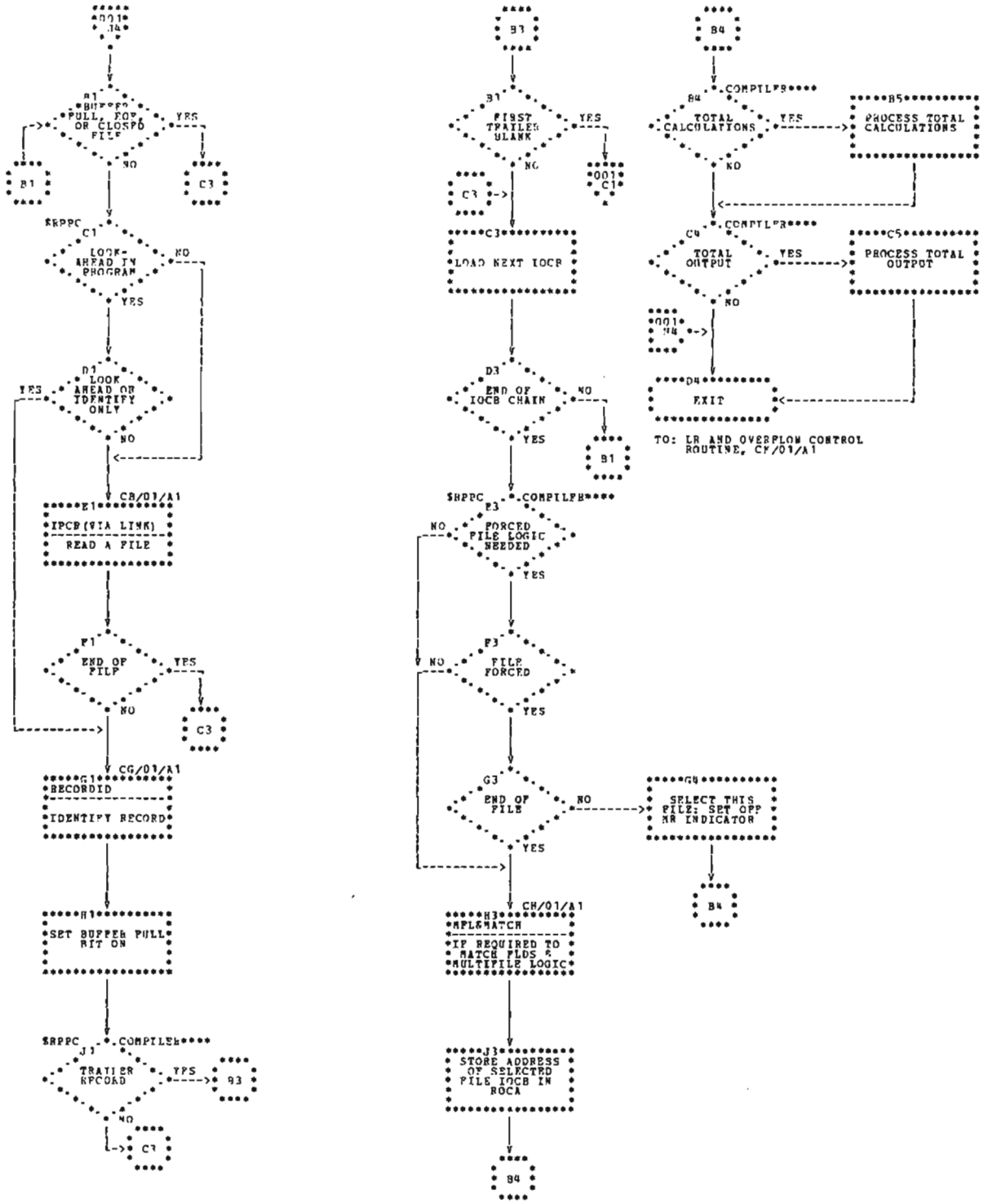


Chart CE (Part 2 of 2). Input Mainline

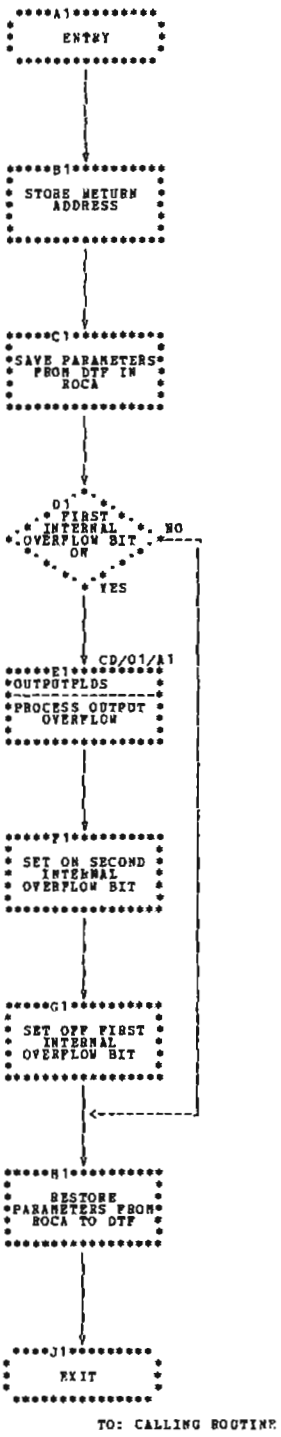


Chart CF. Fetch Overflow

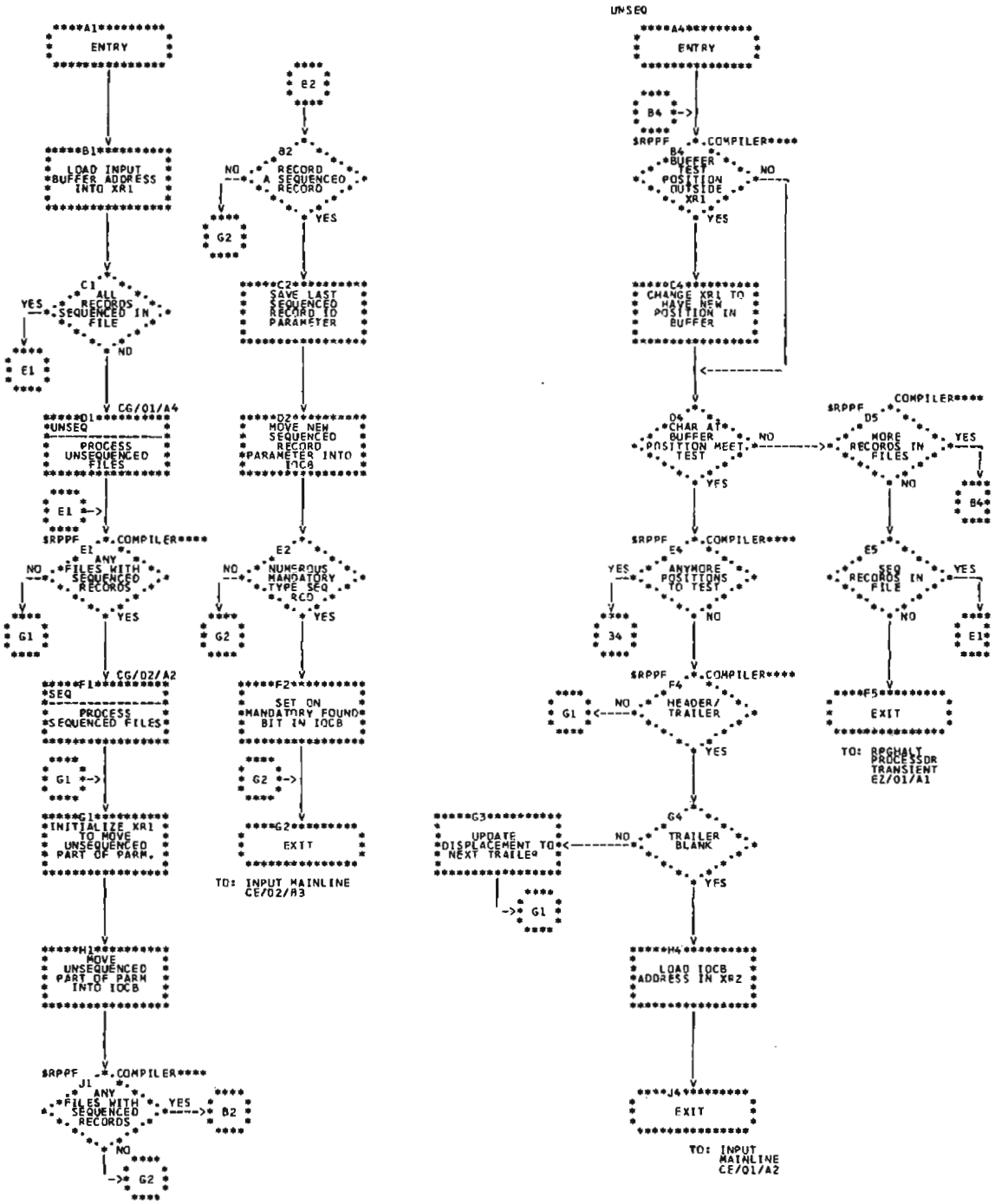


Chart CG (Part 1 of 2). Record ID

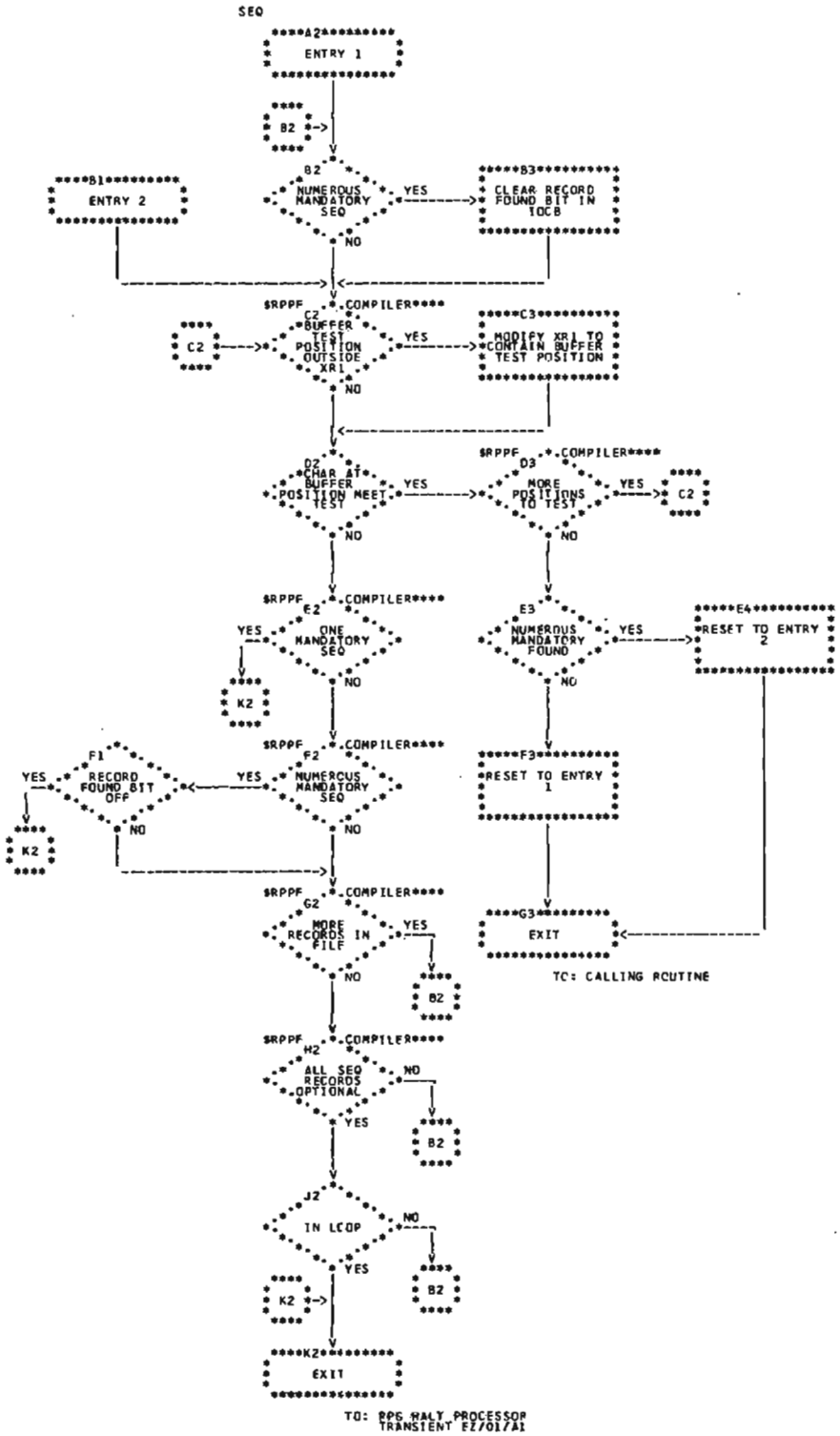


Chart CG (Part 2 of 2). Record ID

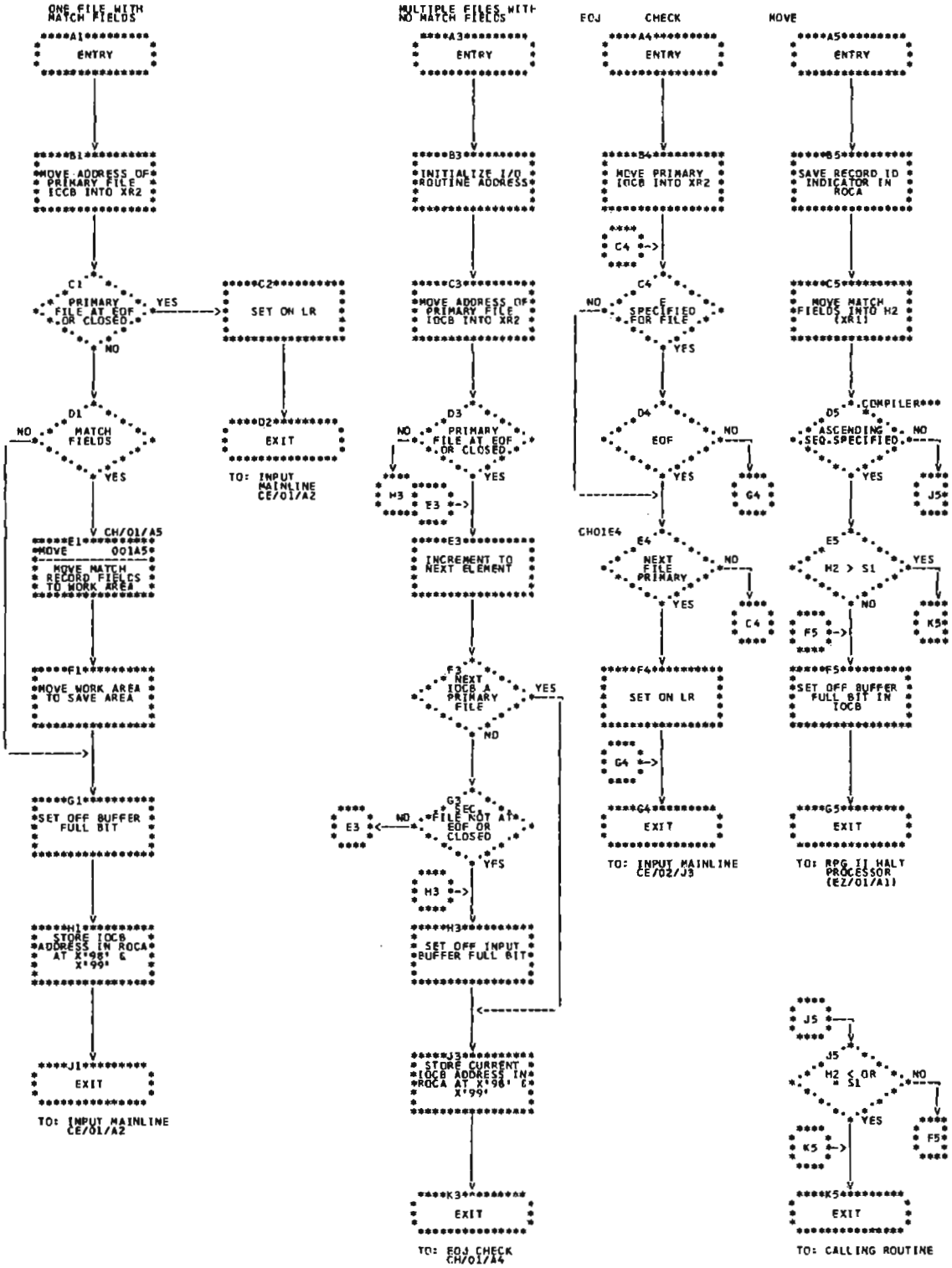


Chart CH (Part 1 of 2). Multifile Logic

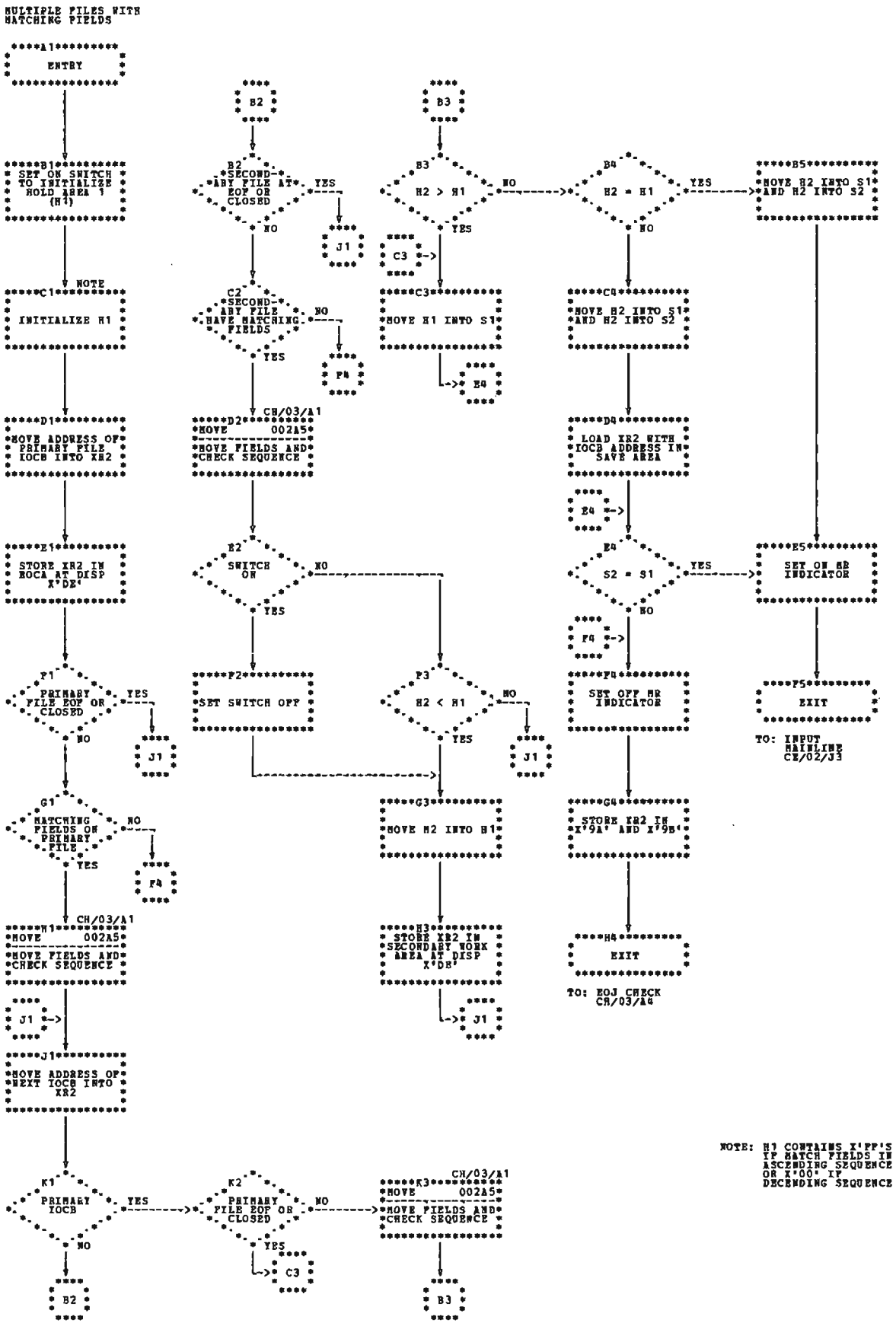
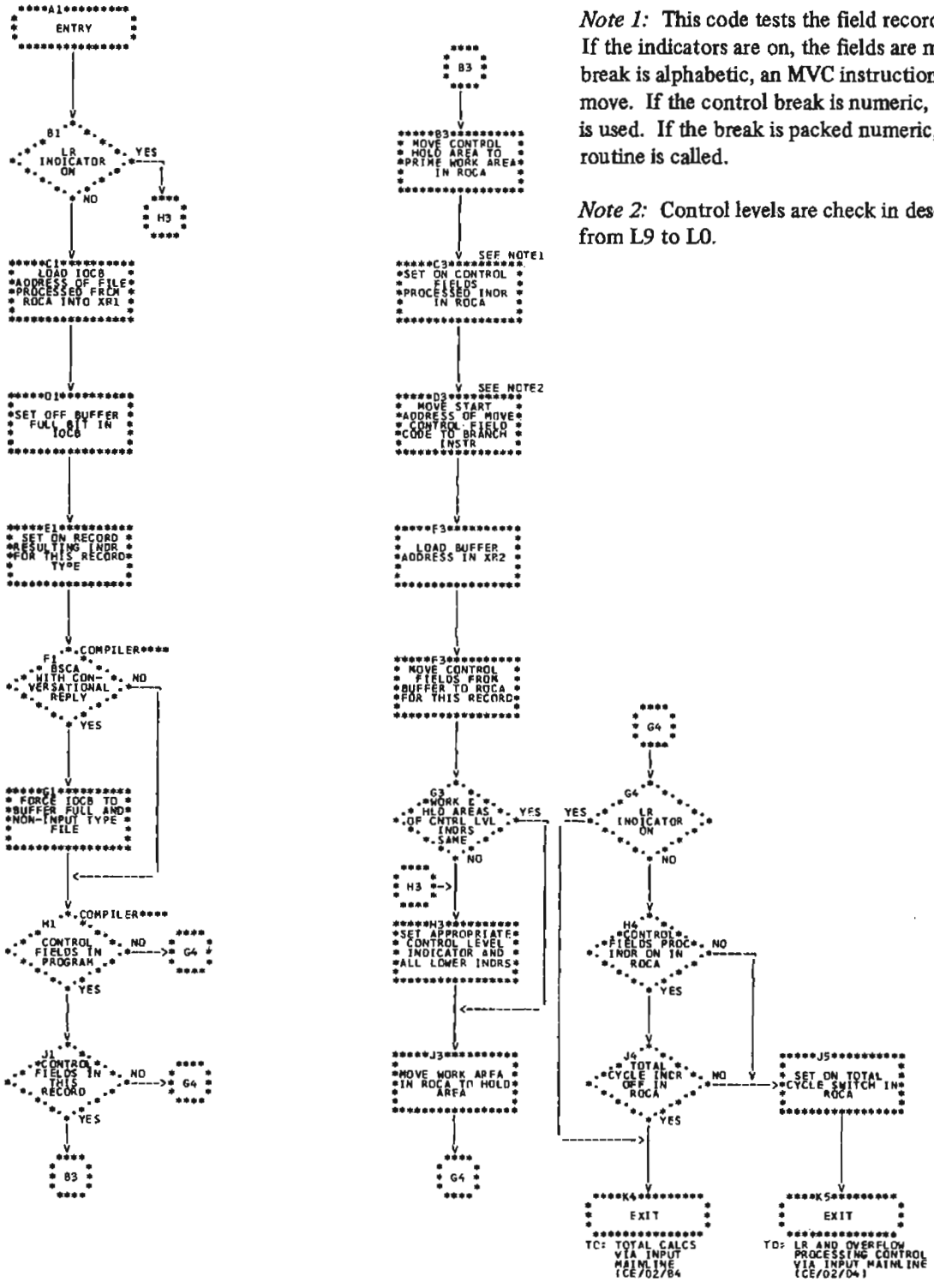


Chart CH (Part 2 of 2). Multifile Logic



Note 1: This code tests the field record relation indicator. If the indicators are on, the fields are moved. If the control break is alphabetic, an MVC instruction is used for the move. If the control break is numeric, a ZAZ instruction is used. If the break is packed numeric, the Unpack subroutine is called.

Note 2: Control levels are check in descending sequence from L9 to L0.

Chart CI. Control Fields Logic and Move

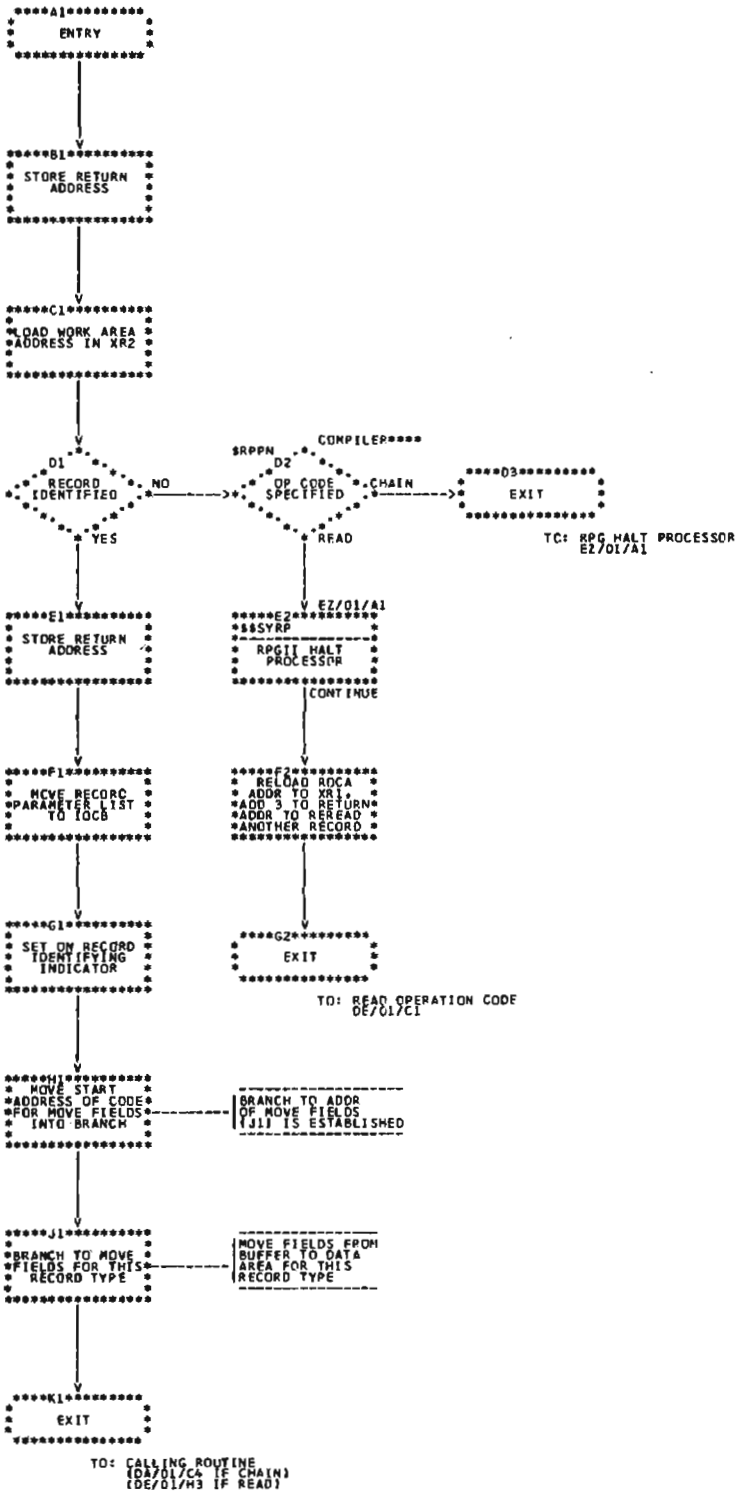


Chart CJ. Chain and Read

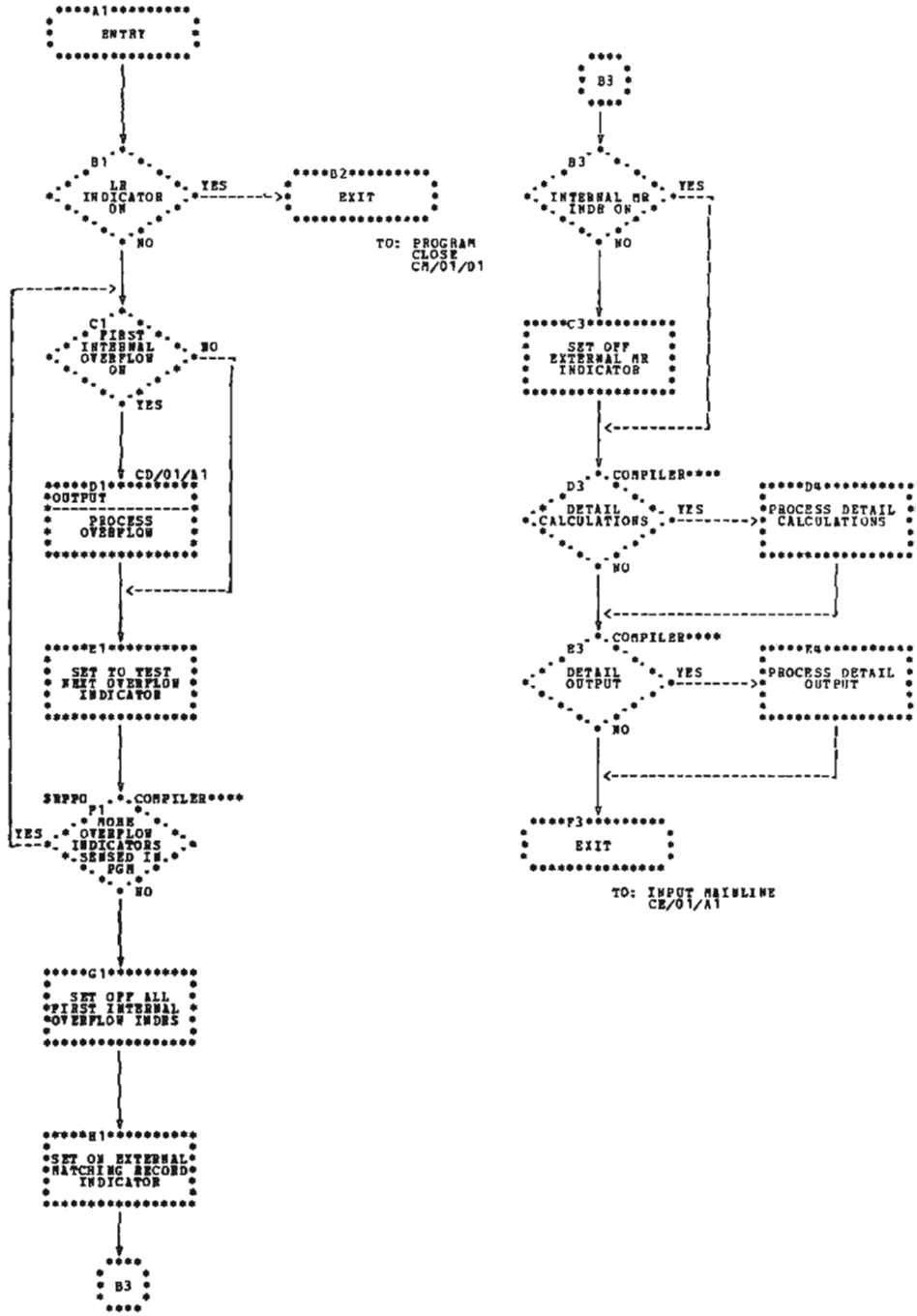


Chart CK. LR and Overflow Control Mainline

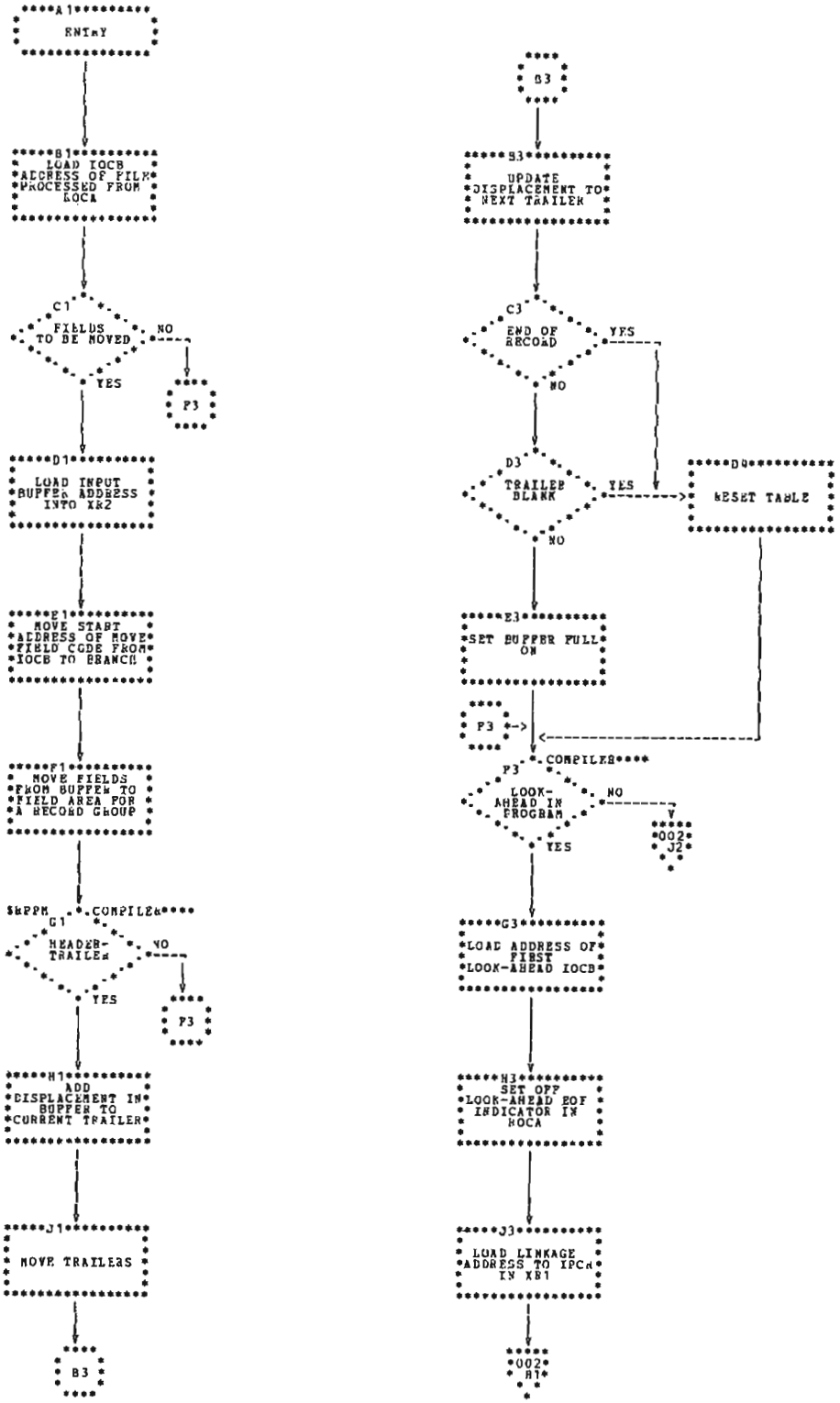


Chart CL (Part 1 of 2). Move Input Fields Mainline

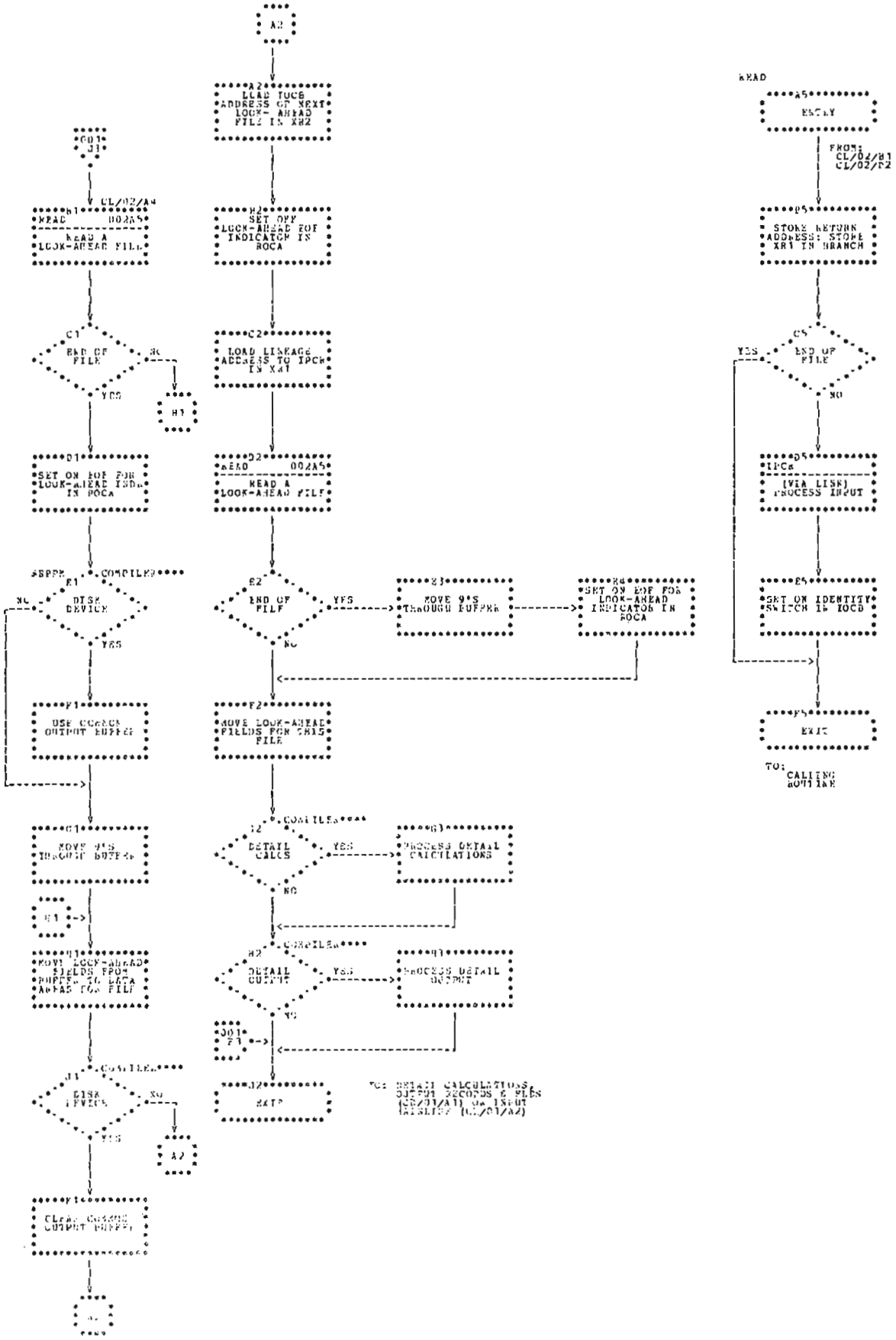


Chart CL (Part 2 of 2). Move Input Fields Mainline

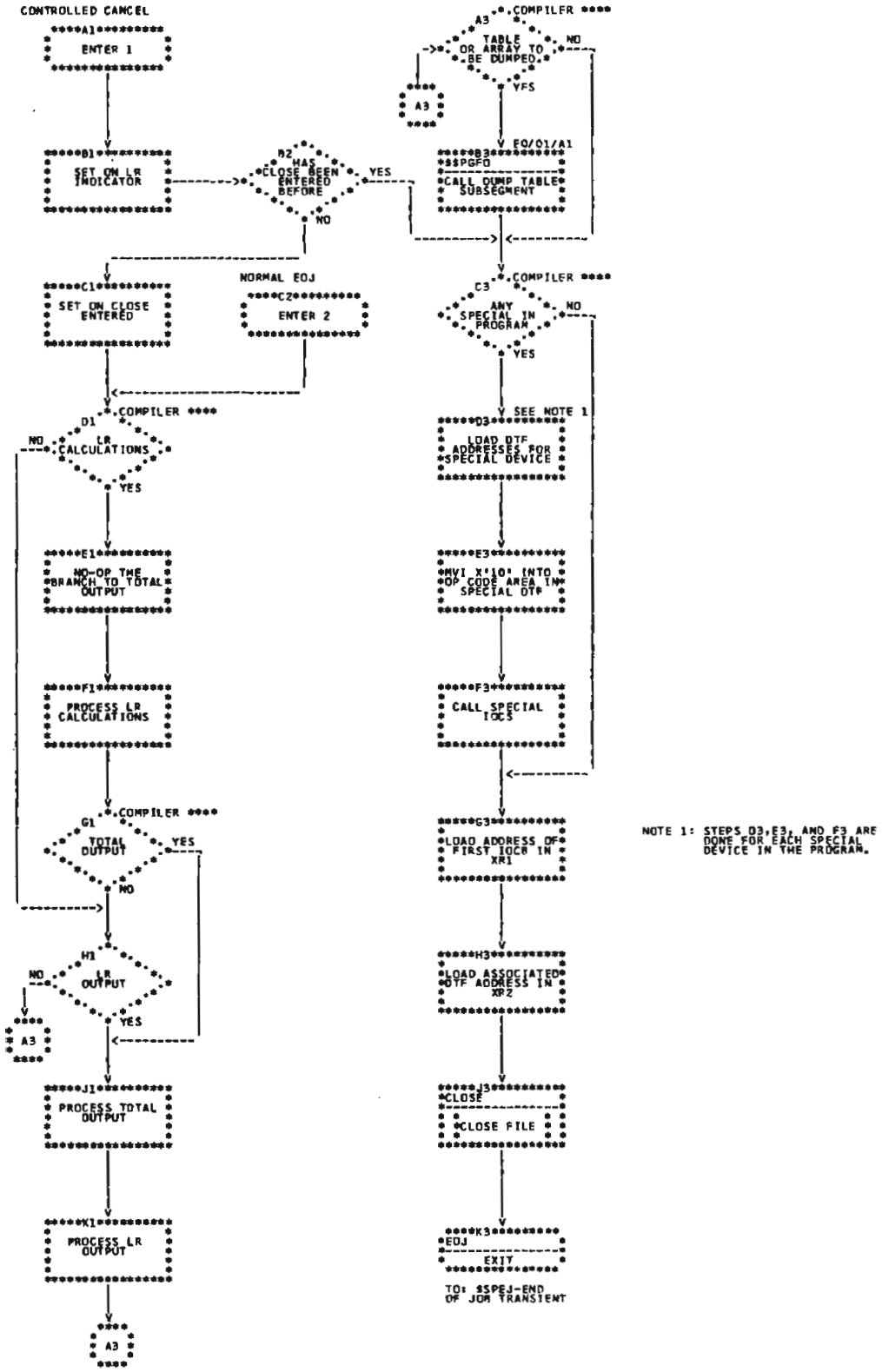


Chart CM. Program Close Mainline

CALCULATIONS OBJECT CODE

This section discusses the sequence of operations performed by the object code generated for each permissible RPG II calculation operation code. Knowledge of the following introductory information is required to understand the discussions of individual specifications.

Abbreviations Used in This Section

ARR	Address recall register
DF1	Number of decimal positions in Factor 1
DF2	Number of decimal positions in Factor 2
DRF	Number of decimal positions in Result Field
DTT	Define the table
F1	Factor 1 (refers to heading on RPG II Calculation sheet)
F2	Factor 2 (refers to heading on RPG II Calculation sheet)
IAR	Instruction address register
IOCB	Input/output control block
LF1	Length of Factor 1
LF2	Length of Factor 2
LRF	Length of Result Field
RF	Result Field (refers to heading on RPG II Calculation sheet)
ROCA	Reserved object communications area
XR1	Index register 1
XR2	Index register 2

Use of Entire Arrays

When any calculation specifies an unindexed array as a factor, the calculation operation must be performed on every element in that array. For example, when moving a field to an array, the field must be moved to every element in that array. If more than one array is used in an operation, processing is terminated when the end of the array with the fewest elements is reached.

Use of Tables

If tables are specified anywhere except as Factor 2 with a LOKUP operation, only the last looked-up element is used in the operation. If the table is referenced before a LOKUP operation, the first element in the table is used. For a description of LOKUP, see *Library of Subroutines* in this section.

Use of RPG II Subroutines

Library subroutines used by the object program are selected by Pre-Assemble phase $RPMP. During object program execution, linkage to each subroutine is done with an unconditional branch followed by a parameter list. For a flowchart and description of each subroutine, including the format of the parameter list, see *Library of Subroutines.*

Use of XR1

Index register 1 (XR1) contains the address of ROCA. It is always restored to that address if changed by any part of the object program.

Use of Conditioning Indicators

Conditioning indicators must be set as specified in the calculation specifications if each operation is to take place. Indicators are tested based on how they are specified in the Calculation Specifications sheet. If the indicator must be ON to perform the operation, a TBN instruction is generated to test the indicator. If the indicator must be OFF to perform the operation, then a TBF instruction is used to test the indicator. A JUMP FALSE instruction is used to bypass an operation. If there are any duplicate indicators, the operations they condition are bypassed with a JUMP FALSE. For example, if three adjacent indicators were:

```
01,02    ADD
01,02    SUB
02,03    ADD
```

The first group of indicators (01, 02) would be tested. But the second group of indicators (01, 02) would not be tested. They would be bypassed using a JUMP FALSE instruction, since the second set of indicators is the same as the first. The next indicator group tested would be 02, 03.

Obtaining Addresses Used in Calculations

The addresses of fields and constants are fixed and are available to the Assemble phases when the object program is generated. The individual discussions of calculation specifications make no reference to the procedure for obtaining the table element or array field addresses unless a different procedure is required. For example, a phase sets a bit on if the table element address or array pointer is moved into some instruction other than the mainline

instruction. The addresses of table elements and array fields must be obtained by the object program in the following manner.

Table Elements: Moves the address of the last-found (or last looked-up) table element from the DTT of the referenced table (see *Data Areas* for DTT description) into the address portion of the main instruction.

Array with Variable Index: Branches to the Array Index subroutine with parameters telling the subroutine to move the address of the specified element into the address portion of the main instruction.

Array with Integer Index: Treats this array like a field since the address of the desired or referenced element is calculated at compile time.

Entire Arrays: Processes every field in the array using a program loop to:

- Initialize the pointer to the array by placing the address of the first element of the array given in the DTT (see *Data Areas*) in either an index register used by the mainline instruction or in the mainline instructions.
- Process one array field.
- Add the length of one element to the array pointer.
- Compare the address contained in the array pointer to the address of the last field in the DTT.
- Branch back to the mainline instruction if all elements have not been processed.

Methods of Preserving Decimal Integrity

Aligning the Decimal Point: Decimal alignment is necessary only for addition or subtraction performed in the prime work area of ROCA. In addition or subtraction, the first decimal position is assumed to be at byte 17 in the prime work area.

The compiler generates instructions to initialize the prime work area. The prime work area is indexed by XR1 with a displacement computed to assure that the first decimal position is always in byte 17. The displacement is calculated by the formula, D = 16 + DF1 (D = 16, since byte 16 contains the last byte of the factor to the left of decimal place). For example, to generate the instruction to move a Factor 1 field which is eight digits long with four decimal positions (D = 16 + 4 = 20), the instruction generation would be:

```
ZAZ 20(16,XR1),F1(8)
```

Displacements for other factors are calculated in a similar manner. In multiplication and division, decimal integrity is maintained.

Half Adjusting: Half adjusting in an arithmetic operation is meaningful only in the following cases:

1. Addition/subtraction – when DRF<DF1 or DF2.
2. Multiplication – when DRF<DF1 + DF2.
3. Division – all cases.
4. Square root – all cases.

After the result is calculated in the prime work area, the one digit to the right of the sign position of the result is added to itself and the Result Field. For example, a Result Field five digits long with three decimal positions is calculated to be 17.6527. The sign position is the 2 so the half adjust operation is as follows:

```
 17.6527
        7
 -------
 17.6534
```

The position to the right of the sign is then dropped when the result is moved from the work area to the field.

Retaining the Sign: IBM System/3 arithmetic operations are performed on numeric fields in zoned decimal format (see *Calculation Specification Descriptions, SQRT).* The sign is always retained in the zone portion of the rightmost byte of the numeric field. Correct results from arithmetic operations depend on the proper location of the sign. Thus, before some arithmetic operations can be performed on fields of unequal lengths, the sign position of one of the fields must be adjusted (shifted left or right) to match the sign position of the second field. In the same way, if the calculated result contains a different number of decimal positions than the Result Field, the sign position of the result must be adjusted to match the sign position of the Result Field. An MZZ instruction is used to move the sign to its proper location. In general, if the Result Field contains more decimal positions than

either Factor 1 or Factor 2, the prime work area must be cleared to binary zeros before the factors are processed there.

Calculation Specification Descriptions

Calculation specifications are discussed in alphabetical order. Conditional instructions, present only in certain cases, are enclosed in brackets ([ ]). Notes are included where necessary.

ADD

The ADD specification causes Factor 1 and Factor 2 to be added together and the sum placed in the Result Field. The sequence of operations is:

[CLEAR the prime work area in ROCA to binary zeros if DRF is greater than both DF1 and DF2.]

[MOVE the longest factor to the prime work area using a ZAZ instruction. If the instruction which clears the prime work area is present, an AZ instruction is used instead of a ZAZ instruction.]

[ADJUST the sign of the moved factor to match the sign position:

1. The Result Field if either F1 or F2 = RF and DF1 = DF2.

2. The shortest factor if DF1 = DF2 = DRF.]

ADD the factors using an AZ instruction. If F1 or F2 = RF and the instruction which adjusts the sign is present, the factor in the prime work area is added directly to the Result Field; otherwise, the factors are added in the prime work area.

[HALF ADJUST the sum.]

[ADJUST the sign position of the sum to match the sign position of the Result Field.]

[MOVE the sum to the Result Field using a ZAZ instruction.]

Note: See *Methods of Preserving Decimal Integrity* for a discussion of sign adjustment and half adjusting.

BEGSR

The BEGSR (begin subroutine) specification signifies the beginning of a subroutine written by the RPG II user. The main instruction is:

STORE the ARR in the return branch generated by the ENDSR specification. This instruction is followed by the object code generated for the user-written subroutine.

BITON

The BITON specification causes the bits specified in Factor 2 to be set on in the Result Field. A 1-byte field name can be substituted for Factor 2. The bits from that byte are then used. A special set of control instructions are required if Factor 2 is not a literal and/or the Result Field is a table/array tag. The main instruction is:

SET bits on using an SBN instruction.

BITOF

The BITOF specification causes the bits specified in Factor 2 to be set off in the Result Field. A 1-byte name can be substituted for Factor 2. The bits from that byte are then used. A special set of control instructions are required if Factor 2 is not a literal and/or the Result Field is a table/array tag. The main instruction is:

SET bits off using an SBF instruction.

CHAIN (Chart DA)

The CHAIN specification allows direct access to a disk file during calculations in the program cycle. The sequence of operations is:

READ a record by branching to the Input Processing Control routine (IPCR).

BRANCH to the Record ID routine and Move Input Fields Mainline to identify the record and move fields.

COMP

The COMP (compare) specification causes Factor 1 to be compared to Factor 2. The factors must either be both alphameric or both numeric. If one alphameric field is

shorter than the other, the shorter field is padded to the right with blanks before the compare. Numeric fields are aligned according to the decimal point. Short fields are padded to the left and right with zeros.

If an alternate collating sequence is specified, the sequence of operation is:

[BRANCH to the Alternate Collating Sequence subroutine. The branch is followed by parameters (see *Library of Subroutines*). The alternate collating sequence subroutine performs the comparison; no other instructions are present for one COMP statement.]

Numeric fields are compared as follows:

[CLEAR the prime work area to binary zeros if DF1≠DF2.]

MOVE Factor 1 to the prime work area with a ZAZ instruction if D1=D2 and subtract Factor 2 from Factor 1. If DF1≠DF2, the factor with shorter decimals (either Factor 1 or Factor 2) is moved into the prime work area.

[ADJUST the sign position of the factor in the prime work area to match the sign position of the longer factor if the instruction which clears the prime work area is present.]

SET the condition register by subtracting the remaining factor from the factor in the prime work area.

BRANCH to the Set Resulting Indicators subroutine. The branch is follwed by parameters (see *Library of Subroutines*).

Alphameric fields of equal lengths are compared as follows:

COMPARE the two factors using a CLC instruction with the addresses of the factors.

Alphameric fields of unequal length are compared as follows:

LOAD XR2 with a pointer to a byte in the longest factor such that the number of bytes to the left of the pointer equals the number of bytes in the shorter factor.

COMPARE the shorter factor to the bytes in the longer factor to the left of the pointer using a CLC instruction.

JUMP not equal around the following instructions to the branch instruction.

TEST the first byte of the longer factor to the right of the pointer for a blank using a CLI instruction.

JUMP, if not a blank, around the following instruction.

TEST the remainder of the longer field for blanks using a CLC instruction.

Note: The condition register may be set by any of the three compare instructions.

BRANCH to the Set Resulting Indicators subroutine. The branch is follwed by parameters (see *Library of Subroutines*).

DEBUG

The DEBUG specification provides a source listing during program execution of:

1. All RPG II indicators that are on.
2. A literal for identification purposes (optional).
3. The contents of any one field (optional). The main instruction is:

BRANCH to the DEBUG subroutine. The branch is followed by parameters (see *Library of Subroutines*).

DIV

The DIV (divide) specification causes Factor 1 to be divided by Factor 2 with the quotient being placed in the Result Field. The sequence of operations is:

[PLACE the addresses of table elements or array fields in the ZAZ instructions which move the factors to the prime work area of ROCA.]

CLEAR the prime work area to binary zeros.

MOVE Factor 2 to the prime work area using a ZAZ instruction.

MOVE Factor 1 to the prime work area using a ZAZ instruction.

BRANCH to the Divide subroutine. No parameters follow the branch (see *Library of Subroutines*).

[HALF ADJUST the quotient.]

[ADJUST the sign position of the quotient to match the sign position of the Result Field.]

MOVE the quotient to the Result Field using a ZAZ instruction.

Note: See *Methods of Preserving Decimal Integrity* for a discussion of sign adjustment and half adjusting.

DSPLY (Chart DB - Models 6, 10, and 12; Chart EO - Model 15)

The DSPLY specification displays either Factor 1, Result Field, or both on a console device. If the Result Field is displayed, the operator is allowed to alter that field. The sequence of operation is as follows:

Models 6, 10, and 12: Branch to OPCR to:

1. Print DSPLY.
2. Display contents of Factor 1 if specified.
3. Display contents of Result Field if specified.
4. Place the reply into the Result Field if there is a reply.

Model 15:

1. Clear the prime work area of ROCA to binary zeros.
2. Move Factor 1 to bytes 0 – n of the prime work area if specified.
3. Move Result Field to bytes 36 – n of the prime work area if specified.
4. Set up 2-byte parameter list (see *Library of Subroutines*).
5. Branch to DSPLY subroutine.

Note: If the Result Field is alphameric, it is left justified; if the Result Field is numeric, it is right justified.

ENDSR

The ENDSR (end subroutine) specification signifies the end of a subroutine written by the RPG II user. The main instruction is:

BRANCH to the return address stored by the BEGSR statement. Object code generated for the user-written subroutine precedes this instruction.

EXCPT

The EXCPT (exception output) specification causes output to be performed during detail or total calculations. The main instruction is:

BRANCH to the output routine for all exception files.

Note: Records are built exactly as normal output records are built (see *Detailed Object Program Flow*).

EXIT

The EXIT specification causes a branch of the main routine to a subroutine. The EXIT specification can also be used with the RLABL specification (see *RLABL*). The main instruction is:

BRANCH to the subroutine specified. The subroutine name is resolved by the linkage editor.

EXSR

The EXSR (execute subroutine) specification causes a branch to be taken to a user-written subroutine for which the first and last instructions were generated by BEGSR and ENDSR statements. The main instruction generated is:

BRANCH to the subroutine entry point. The compiler places the subroutine entry point in the branch instruction.

FORCE (Chart DC)

The FORCE specification causes the next record for processing to be taken from the file specified as Factor 2. The sequence of operations is:

LOAD the address of the forced file IOCB into XR2.

TEST if file is at end of file. If so, jump around next statement.

STORE addreses of the forced IOCB and the file IOCB in the prime work area in ROCA.

GOTO

The GOTO specification causes a branch to be taken to the specified TAG or ENDSR address. The main instruction is:

BRANCH to the TAG or ENDSR address. The compiler places the TAG or ENDSR address in the branch instruction.

KEY (Model 6 Only)

The KEY specification allows fields to be keyed from the keyboard at calculation time. Alphameric fields are left justified, and numeric fields are right justified. Depending on what is specified, the operation is treated in one of three ways:

1. Display Mode – Place keyed field into main storage location and display on printer.
2. Store Only Mode – Place keyed field into main storage location (no display).
3. Manual Mode – Keyed field is displayed (doesn't go into main storage).

Each KEY specification requires the following code:

MOVE parameters to the DTF

LOAD IOCB address into XR2

BRANCH to IPCR

If there are resulting indicators, the following code is generated for a numeric field before the branch to the Set Resulting Indicator routine:

- *1-byte field*

  [LOAD the field address into XR2]

  [ZAZ the field to set the condition code]

- *Greater than 1-byte field*

  [LOAD the field address into XR2]

  [ADD to register so it points to right end minus 1 of field]

  [ZAZ the field to set the condition code]

  [JUMP past following DC]

  [DC a 2-byte constant of field length minus 2]

The following code is generated for an alphameric field:

- *1-byte field*

  [LOAD the field address into XR2]

  [CLI the field to set the condition code]

- *Greater than 1-byte field*

  [LOAD the field address into XR2]

  [ADD to register so it points to right end minus 1 of field]

  [CLI right byte of field to test for a blank]

  [JNE around next CLC]

  [JUMP past following DC]

  [DC a 2-byte constant of field length minus 2]

If the KEY field is an array element or a table element the following code is generated first:

[SLC to adjust the right hand element address (as determined by array control to the left hand element address) to be used by the rest of the KEY object code]

For SET/KEY combinations, see the description of the SET operation code in this section.

LOKUP (Chart DD)

The LOKUP (look-up) specification causes elements of the table or array specified as Factor 2 to be compared against a field specified as Factor 1 until the user-specified setting of the condition register is reached. The sequence of operations is:

SET off resulting indicators.

MOVE the address of the last found element to the LOKUP parameter list if Factor 1 is a table. If Factor 1 is an array with variable index:

1. Branch to the Array Index subroutine

2. Store the calculated array element address in the LOKUP parameter list.

BRANCH to the LOKUP subroutine.

SET on the appropriate resulting indicator if the desired conditions have been met.

MHHZO

The MHHZO (move high-high zone) specification causes the zone portion of the leftmost byte of Factor 2 to be moved to the zone portion of the leftmost byte of the Result Field. The sequence of operations is:

[OBTAIN the address of the leftmost byte of array fields or table elements. If only one factor is an array or table element:

1. Load XR2 with a negative constant, 1 minus L, where L is the length of the field.

2. Add the field address XR2. If both factors are arrays or table elements, the preceding procedure is used for Factor 2 along with these instructions:

   a. Load XR1 with a negative constant, 1 minus LRF.
   b. Add the address of the Result Field to XR1.]

MOVE the zone portion of the leftmost byte of Factor 2 to the zone portion of the leftmost byte of the Result Field using an MZZ instruction.

[RESTORE XR1.]

MHLZO

The MHLZO (move high-low zone) specification causes the zone portion of the leftmost byte of Factor 2 to be moved to the zone portion of the rightmost byte of the Result Field. The sequence of operations is:

[OBTAIN the address of the leftmost byte of Factor 2 as follows if Factor 2 is an array or table element:

1. Load XR2 with a negative constant, 1-LF2.

2. Add the field address to XR2.]

MOVE the zone portion of the leftmost byte of Factor 2 to the zone portion of the rightmost byte of the Result Field using an MZZ instruction.

MLHZO

The MLHZO (move low-high zone) specification causes the zone portion of the rightmost byte of Factor 2 to be moved to the zone portion of the leftmost byte of the Result Field. The sequence of operations is:

[OBTAIN the address of the Result Field as follows if the Result Field is an array or table element:

1. Load XR2 with a negative constant, 1 minus LRF.
2. Add the field address to XR2.]

MOVE the zone of the rightmost byte of Factor 2 to the zone of the leftmost byte of the Result Field using an MZZ instruction.

MLLZO

The MLLZO (move low-low zone) specification causes the zone portion of the rightmost byte of Factor 2 to be moved to the zone portion of the rightmost byte of the Result Field. The main instruction is:

MOVE the zone portion of the rightmost byte of Factor 2 to the zone portion of the rightmost byte of the Result Field using an MZZ instruction.

MOVE

The MOVE specification causes the specified Factor 2 to be moved into the Result Field. If the Result Field is shorter than Factor 2, the leftmost bytes of Factor 2 are truncated. The main instruction is:

MOVE Factor 2 to the Result Field using an MVC instruction.

Note: If the Result Field is numeric, a ZAZ instruction is used instead of an MVC instruction.

MOVEA

The MOVEA (move array) specification causes characters from Factor 2 to be moved to the leftmost positions of the Result Field. If the Result Field is shorter than Factor 2, the excess rightmost bytes of Factor 2 are not moved. If the Result Field is longer than Factor 2, the characters to the right of the data moved to the Result Field will remain unchanged. The sequence of operations is:

BUILD the ten-byte parameter list which follows the branch instruction (see *Library of Subroutines*).

BRANCH to the MOVEA subroutine.

SET the move length to the shorter of the Factor 2 and Result Field lengths.

DETERMINE the proper addresses.

MOVE Factor 2 to the Result Field, using an MVC instruction.

Note: Either Factor 2 or the Result Field must reference an alphameric array. However, both Factor 2 and the Result Field cannot reference the same array.

MOVEL

The MOVEL (move-left) specification causes characters from Factor 2 to be moved to the leftmost positions of the Result Field. If the Result Field is shorter than Factor 2, the excess rightmost bytes of Factor 2 are not moved. The sequence of operations is:

[ADJUST the address of the longer field to assure that the leftmost bytes of the Result Field are filled by the equal number of leftmost bytes of Factor 2 (LF2≠LRF):

1. Load XR2 with a negative constant, LS minus LL, where LS is the length of the shorter field and LL is the length of the longer field.

2. Add the address of the longer field to XR2.]

MOVE Factor 2 to the Result Field using an MVC instruction.

Note: If the Result Field is numeric, a ZAZ instruction is used instead of an MVC instruction.

MOVE the sign of Factor 2 to the sign position of the Result Field using an MZZ instruction if the Result Field is numeric and Factor 2 is longer than the Result Field.

[FORCE the sign position (zone portion of the rightmost byte) of the result field to an F zone using an SBN instruction, unless the sign of Factor 2 is minus; then FORCE the sign position of the result field to a D zone using a SBF instruction.]

MULT

The MULT (multiply) specification causes Factor 2 to be multiplied by Factor 1 with the product being placed in the Result Field. The sequence of operations is:

[PLACE the address of field, table, or array elements in the ZAZ instructions which move the factors to or from the prime work area in ROCA rather than in the main instruction itself.]

MOVE Factor 1 to the prime work area using a ZAZ instruction.

MOVE Factor 2 to the prime work area using a ZAZ instruction.

MOVE length parameter to the prime work area using a MVI instruction (Model 15 only).

BRANCH to the Multiply subroutine. No parameters follow the branch (see *Library of Subroutines*).

[HALF ADJUST the product.]

[ADJUST the sign position of the product to match the sign position of the Result Field.]

MOVE the product to the Result Field using a ZAZ instruction.

Note: See *Methods of Preserving Decimal Integrity* for a discussion of sign adjustment and half adjusting.

MVR

The MVR (move remainder) specification causes the remainder of the preceding DIV operation to be moved to the specified field. The sequence of operations is:

[ADJUST the sign position of the remainder to match the sign position of the Result Field.]

MOVE the remainder to the Result Field using a ZAZ instruction.

READ (Chart DE)

The READ operation code calls for immediate input from a demand file during calculations in the program cycle. The sequence of operations is:

READ a record by branching to the Input Processing Control routine (IPCR).

BRANCH to the Chain and Read subsegment to identify the record and move fields.

Note: If the continue option is taken on the unidentified-record halt, another record is read immediately from the demand file.

RLABL

The RLABL specification generates a 3-byte parameter list for each field and array/table and each indicator with IN followed by two letters of a valid indicator (example: INL3). The parameter list is used by the user-coded subroutine used with EXIT. The sequence of operations is:

BRANCH to a user-coded subroutine.

USE and check the RLABL parameter list.

RETURN to the next sequential instruction after the RLABL parameter list.

SET (Model 6)

The SET specification allows control of the printer through spacing, skipping, and positioning the print element and through setting command key indicators on or off. If more than one function is specified, the printer control code is generated before the command key code.

The following code is generated for printer control:

MOVE parameters to DTF

BRANCH to OPCR

The following code is generated if command key indicators are specified:

LOAD IOCB address into XR2

MOVE parameters to DTF

BRANCH TO IPCR

BRANCH to command key indicator set routine followed by a parameter pointing to the allowed command keys and the depressed command keys. See *Parameters to Data Management* in this section for more information about the parameters.

The following code is generated for ledger card eject:

MOVE eject parameter to the DTF

BRANCH to OPCR

Tab sets are built into a table and followed by KEY, which has a parameter pointing to the table.

The special SET/KEY combination will cause the allowed command keys from the SET to be placed in the KEY parameters. This eliminates the above command key code except for branch to the Indicator Set routine allowing both functions with one program start.

SETLL

The SETLL (set lower limit) specification causes the content of Factor 1 to be moved to the low key limits area associated with the file being processed by limits given in Factor 2. EOF conditions are set in the DTF to indicate to Data Management that a new set of limits must be defined on subsequent read operations to the specified file. An EOF condition is also set off in the appropriate IOCB for IPCR considerations. The sequence of operations is:

PACK the contents of Factor 1 (if the key is in unpacked format and requires packing) via a branch to the Pack routine.

MOVE Factor 1 to the low key area, using a MVC instruction.

MOVE the EOF condition to the DTF, using an MVI instruction and a X'42' mask.

SET off the End of File bits in the IOCB, using the SBF instruction and the X'80' mask.

SET on the indicator bits in the DTF to indicate to Data Management to check for EOF using an SBN instruction and the X'40' mask.

SETOF

The SETOF (set indicator off) specification causes the specified RPG II indicators to be set off. The main instruction is:

SET an indicator off by using an SBF instruction. One SBF instruction is present for each indicator specified in the SETOF specification if no optimization can be performed. If a command key indicator is set off, additional instructions LIO and DC are added for each indicator to turn off the associated command key light(s) on the console.

SETON

The SETON (set indicator on) specification causes the specified RPG II indicators to be set on. The main instruction is:

SET an indicator on using an SBN instruction. One SBN instruction is present for each indicator specified in the SETON specification if no optimization can be performed.

SQRT

The SQRT (square root) specification causes the square root of Factor 2 to be placed in the Result Field. The main instruction is:

BRANCH to the SQRT subroutine. The branch is followed by parameters (see *Library of Subroutines*). The parameters built by compiler phase $RPQF tell this subroutine how to adjust the size of the source field (Factor 2) to the size needed by the Result Field. For every decimal and whole number in the Result Field, two decimal places and whole numbers are needed in the source field.

SUB

The SUB (subtract) specification causes Factor 2 to be subtracted from Factor 1 with the result being moved to the Result Field. The sequence of operations is:

[CLEAR the prime work area in ROCA to binary zeros if DRF>DF1 and DF2.]

MOVE Factor 1 to the prime work area using a ZAZ instruction. If the instruction which clears the prime work area is present, an AZ instruction is used instead of a ZAZ instruction.

[ADJUST the sign position of Factor 1 to match the sign position of Factor 2.]

SUBTRACT Factor 2 from Factor 1 using an SZ instruction.

[ADJUST the sign position of the result to match the sign position of the Result Field.]

[MOVE the result of the Result Field using a ZAZ instruction.]

Note: See *Methods of Preserving Decimal Integrity* for a discussion of sign adjustment and half adjusting.

TAG

The TAG specification signifies a point to which any GOTO statement may branch. No object code is generated for this statement.

TESTB

The TESTB (test bits) specification causes the bits specified in Factor 2 to be tested in the Result Field. A 1-byte field name can be substituted for Factor 2. The bits from that byte are then used. A special set of control instructions are required if Factor 2 is other than a literal and/or if the Result Field is a table or array with variable index. If all bits are off, an indicator in columns 54-55 is set on. If all bits are on, an indicator in columns 58-59 is set on. If the bits are mixed, an indicator in columns 56-57 is set on. The main instructions are:

TEST bits on or off using a TBN or TBF instruction.

JUMP true or false (JT or JF).

SET on the RPG II indicators as specified using a SBN instruction. Indicators are set off if the condition does not exist.

TESTZ

The TESTZ (test zone) specification causes the leftmost byte of an alphameric Result Field to be tested. Resulting indicators are used to determine the results of the test. The zone portion of the + and A through I characters causes the plus indicator to be set on. The zone portion of the -(minus) and J through R characters causes the minus indicator to be set on. All other characters, when tested, cause the blank indicator to be set on. The sequence of operations is:

[OBTAIN the address of the array or table element specified as the Result Field as follows:

1. Load XR2 with a negative constant, 1 minus LRF.
2. Add the field address to XR2.]

MOVE the leftmost byte of the Result Field to the prime work area in ROCA.

BRANCH to the Test Zone subroutine. No parameters follow the branch (see *Library of Subroutines*).

TIME (Model 15 Only)

The TIME specification causes the TIME, or TIME and system DATE to be placed in a field, indexed array element, or table entry. Where it is placed depends on the 6 or 12 byte numeric result field. The sequence of operations is:

1. Point XR2 to the beginning of ROCA.
2. Move a flag byte of 0 to byte 0 of ROCA requesting the time and date in decimal units.
3. Place time and system date in bytes 1-12 of ROCA using Supervisor call with TIME of day RIB request (X'96').
4. Move the requested 6 or 12 bytes from ROCA to the result field.

XFOOT

The XFOOT (crossfoot) specification causes all fields in an array specified as Factor 2 to be added together. The result is placed in another field specified as the Result Field. The sequence of operations is:

CLEAR the prime work area to binary zeros.

ADD a field of the array to the prime work area.

Note: Normal array loop control is present. The main instruction is executed for each field in the array.

[HALF ADJUST the sum.]

[ADJUST the sign position of the sum to match the sign position of the Result Field.]

MOVE the sum to the Result Field using a ZAZ instruction.

Note: See *Methods of Preserving Decimal Integrity* for a discussion of sign adjustment and half adjusting.

Z-ADD

The Z-ADD (zero and add) specification causes Factor 2 to be added to the Result Field after the Result Field is set to decimal zeros. The sequence of operations is:

MOVE Factor 2 to the prime work area in ROCA using a ZAZ instruction.

[HALF ADJUST Factor 2.]

[ADJUST the sign position of Factor 2 to match the sign position of the Result Field.]

MOVE Factor 2 to the Result Field using a ZAZ instruction.

Note: See *Methods of Preserving Decimal Integrity* for a discussion of sign adjustment and half adjusting.

Z-SUB

The Z-SUB (zero and subtract) specification causes the negative value of Factor 2 to be placed in the Result Field. The sequence of operations is:

CLEAR the prime work area in ROCA to logical zeros.

SUBTRACT Factor 2 from the prime work area using an SZ instruction.

[HALF ADJUST the result.]

[ADJUST the sign position of the result to match the sign position of the Result Field.]

MOVE the result from the prime work area to the Result Field.

Note: See *Methods of Preserving Decimal Integrity* for a discussion of sign adjustment and half adjusting.

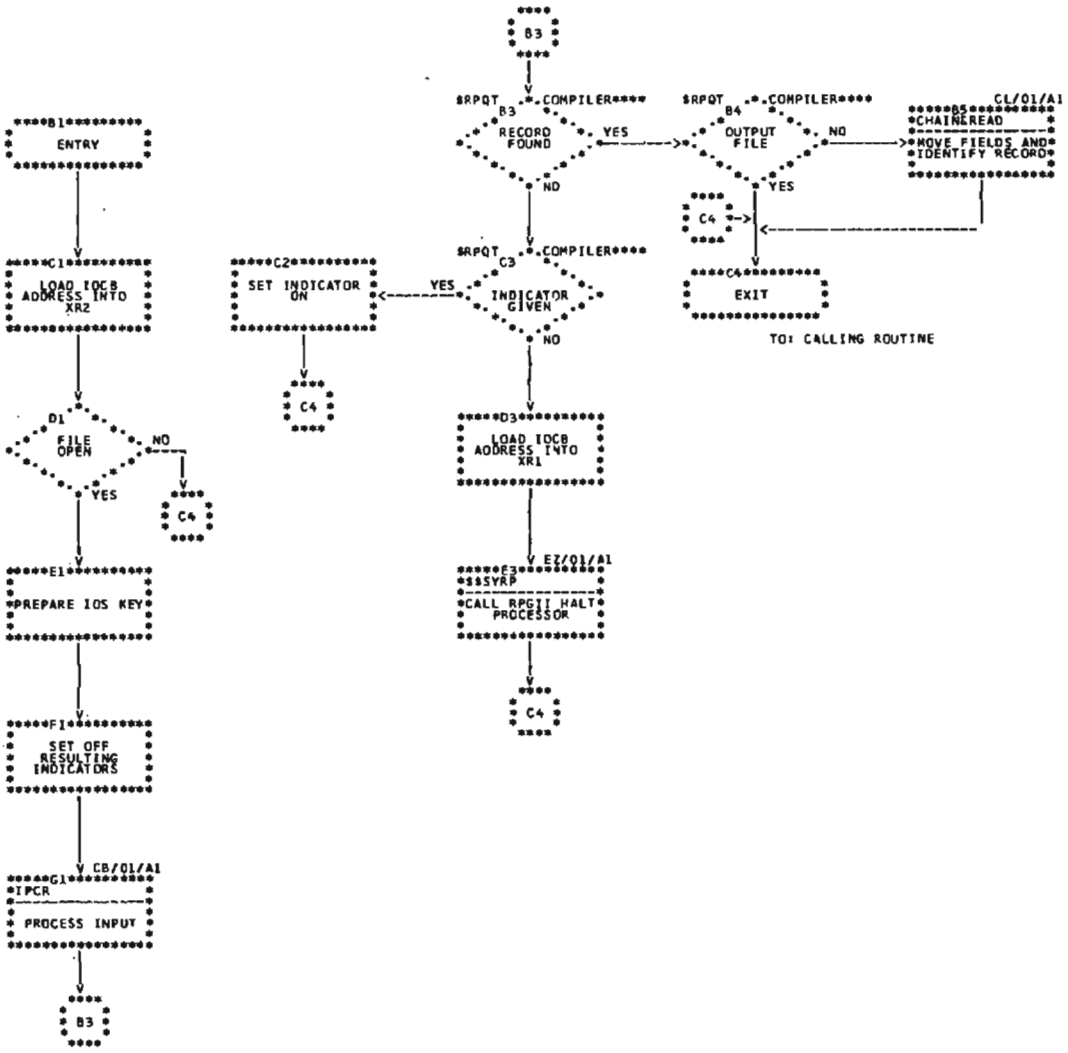


Chart DA. CHAIN Operation Code

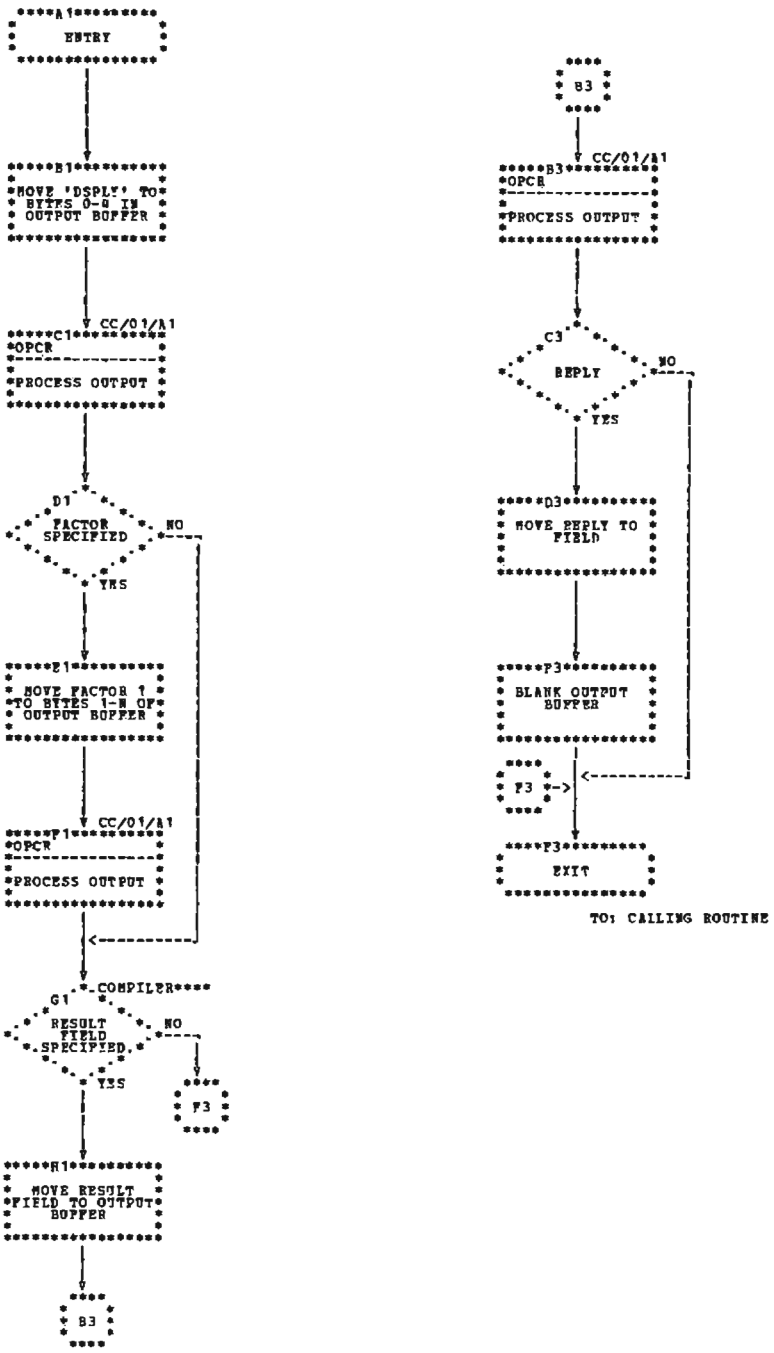


Chart DB. DSPLY Operation Code

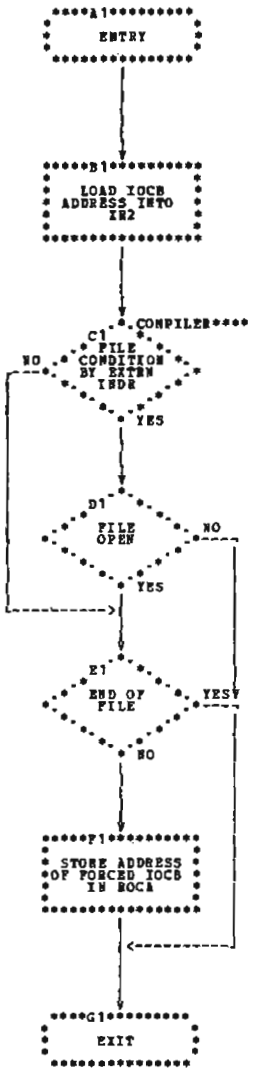


Chart DC. FORCE Operation Code

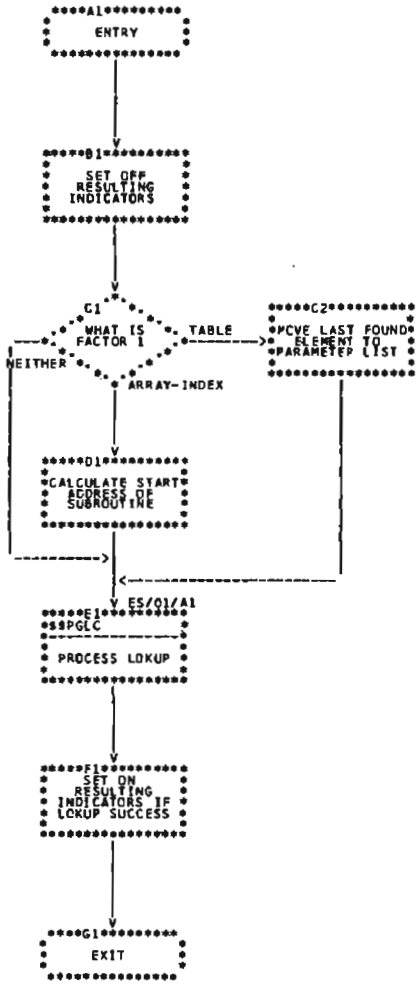


Chart DD. LOKUP Operation Code

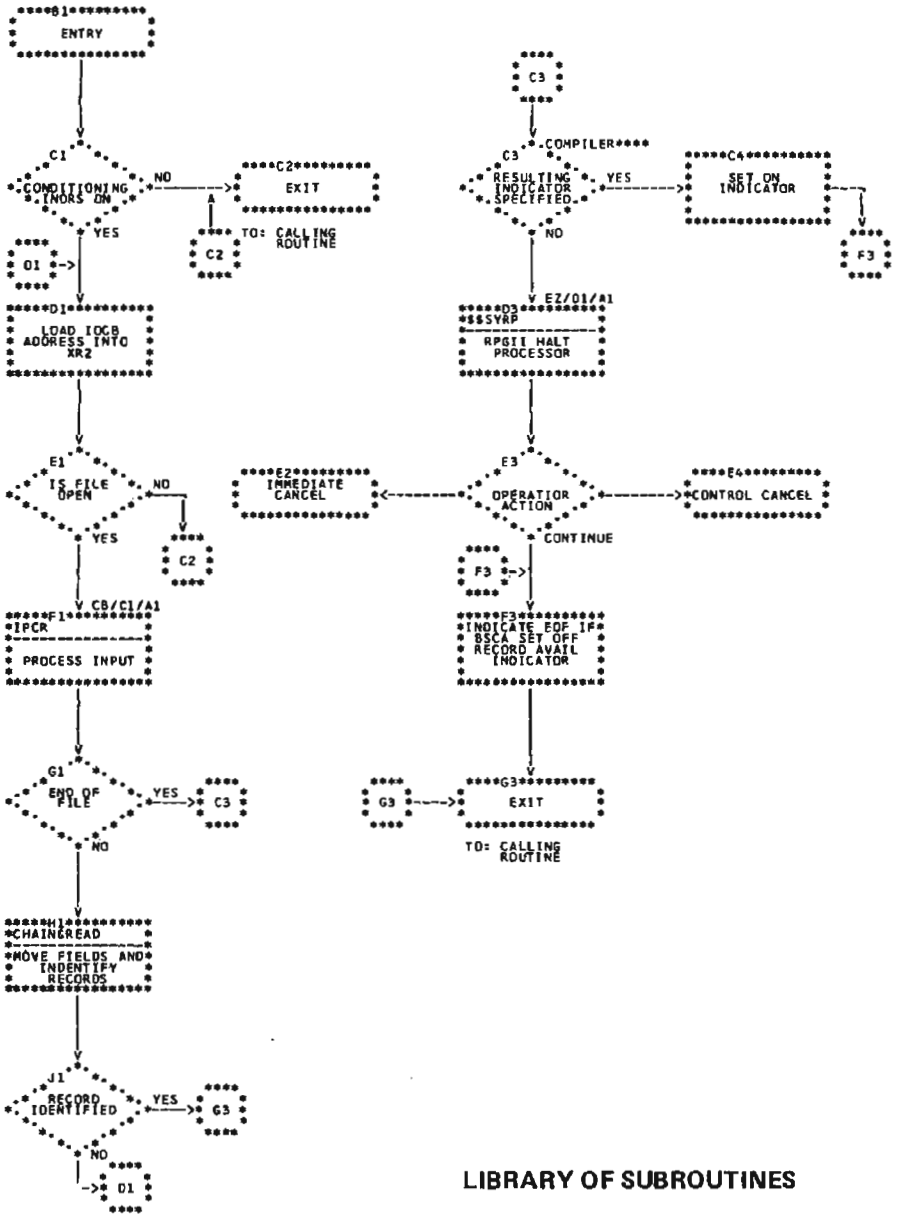


Chart DE. READ Operation Code

LIBRARY OF SUBROUTINES

Some RPG II functions and data conversions are performed by subroutines rather than by in-line sequential instructions. For example, multiplication is performed by the Multiply subroutine and table files are loaded by the Load Object Tables subroutine. The subroutines reside in the object library during compilation. They are selected by the Pre-Assemble and Assemble phases and included in the object program overlay segments by the Overlay phases.

The entry point to each subroutine is the last four characters of the phase name. The exit from each of the subroutine is to the calling routine. Figure 4-3 shows the object program flow of the subroutines.

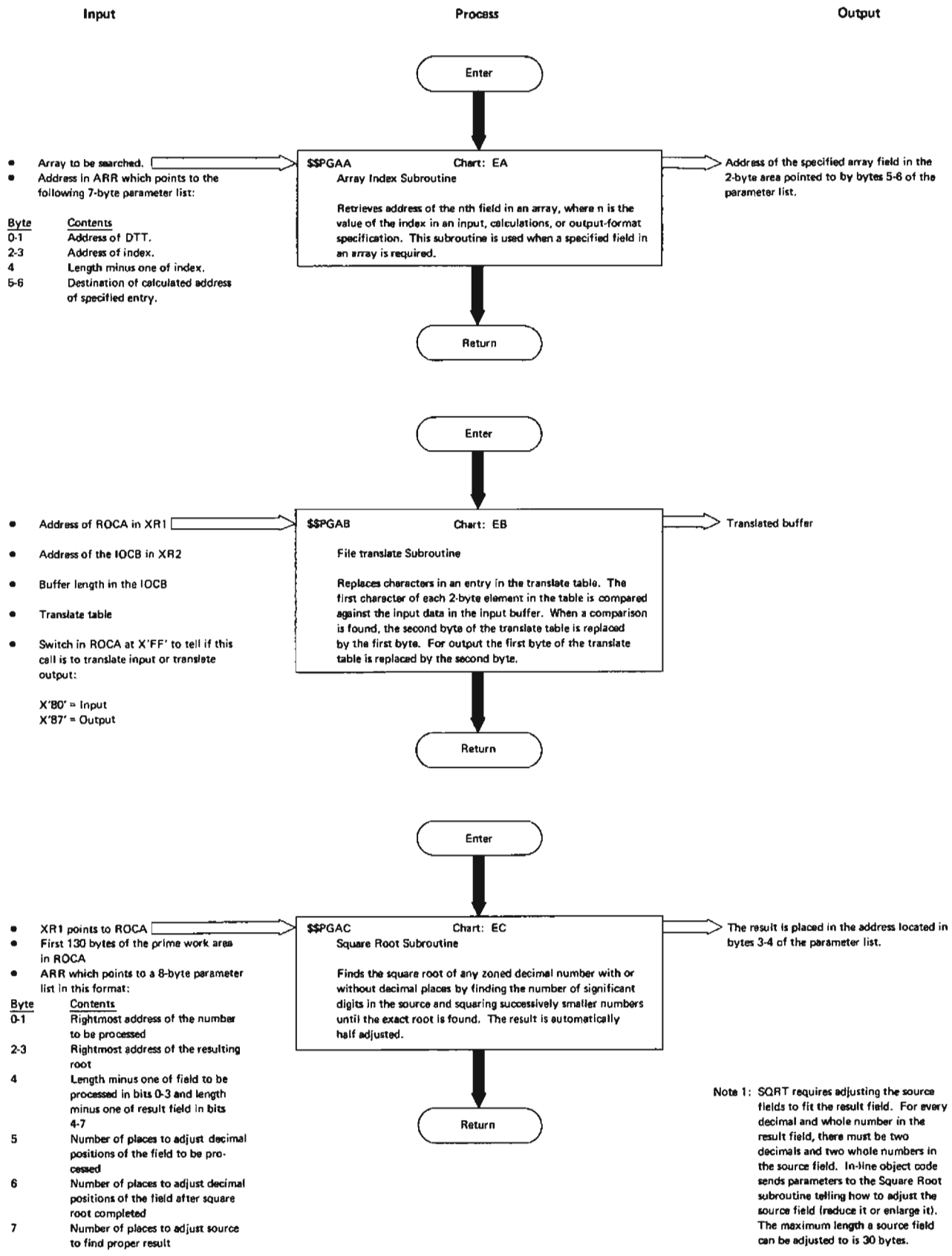


Figure 4-3 (Part 1 of 9). Library of Subroutines

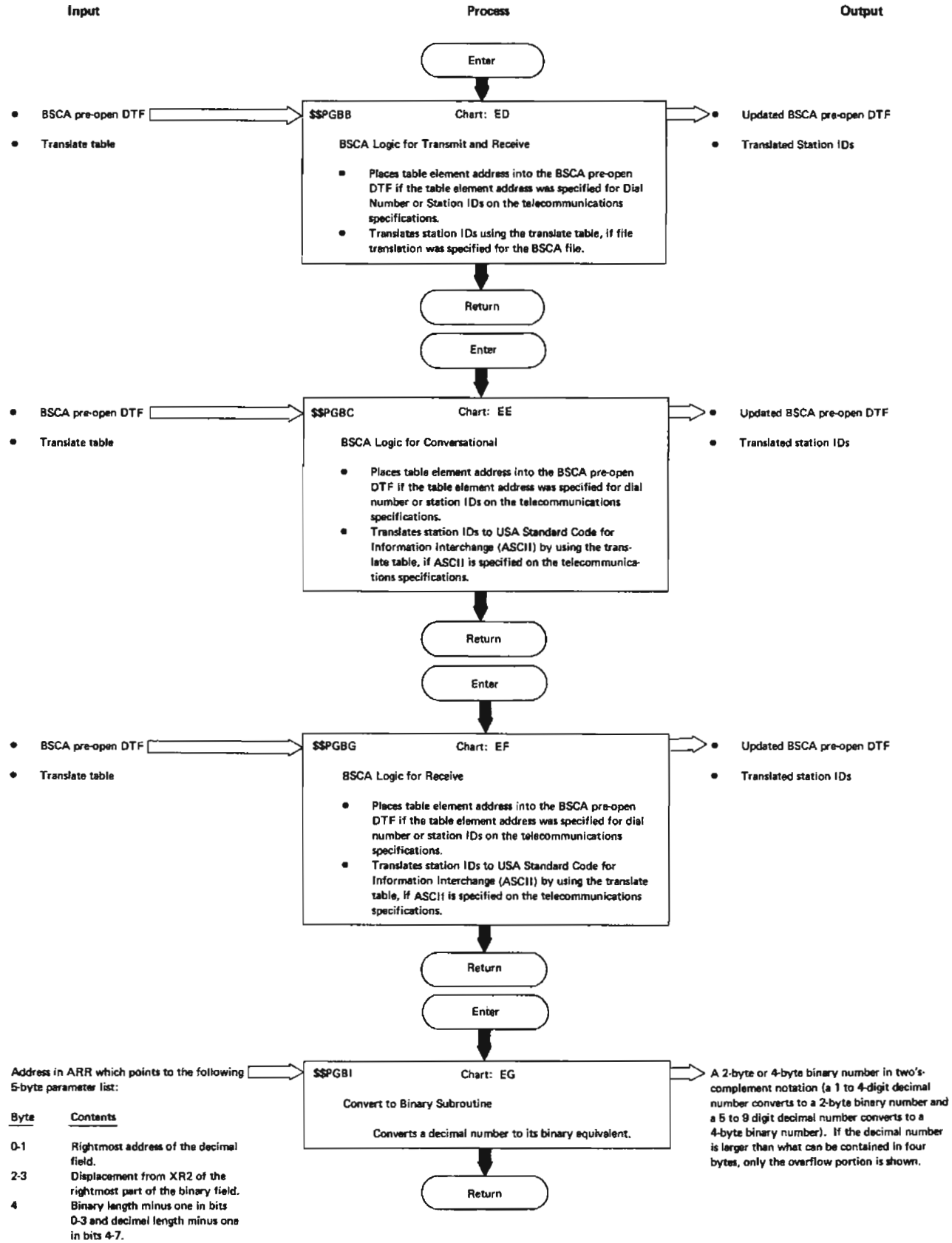


Figure 4-3 (Part 2 of 9). Library of Subroutines

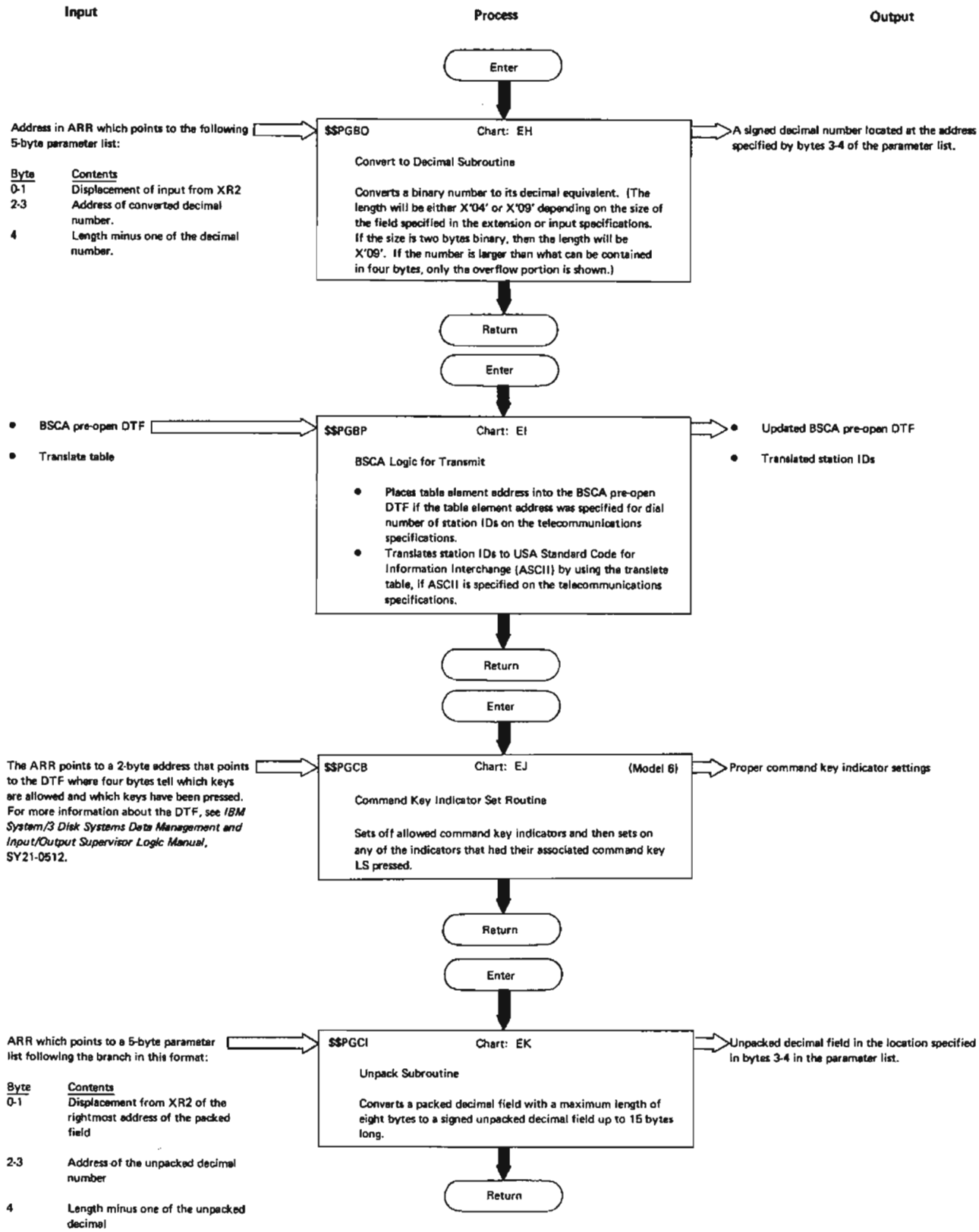


Figure 4-3 (Part 3 of 9). Library of Subroutines

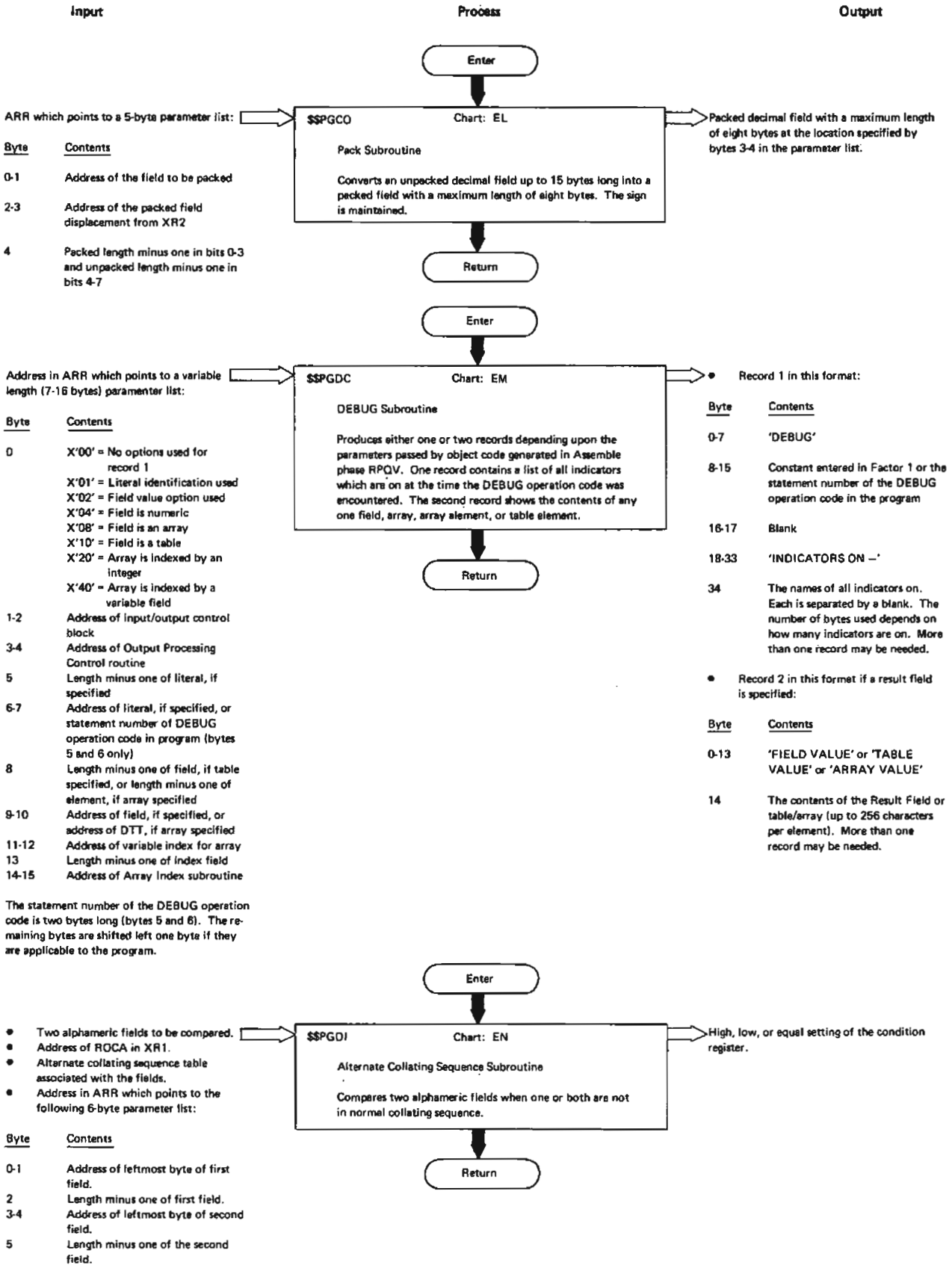


Figure 4-3 (Part 4 of 9). Library of Subroutines

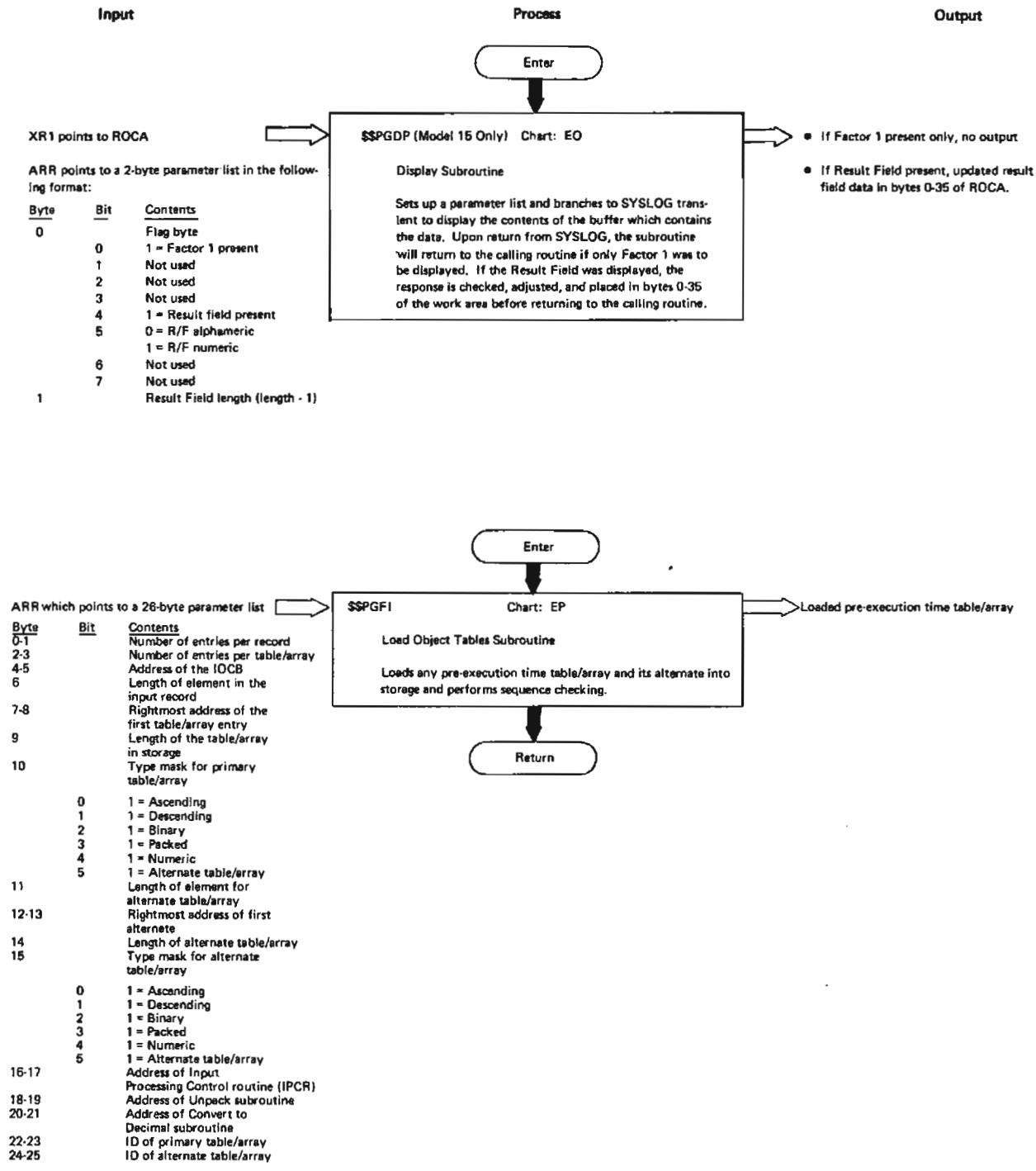


Figure 4-3 (Part 5 of 9). Library of Subroutines

Figure 4-3 (Part 6 of 9). Library of Subroutines

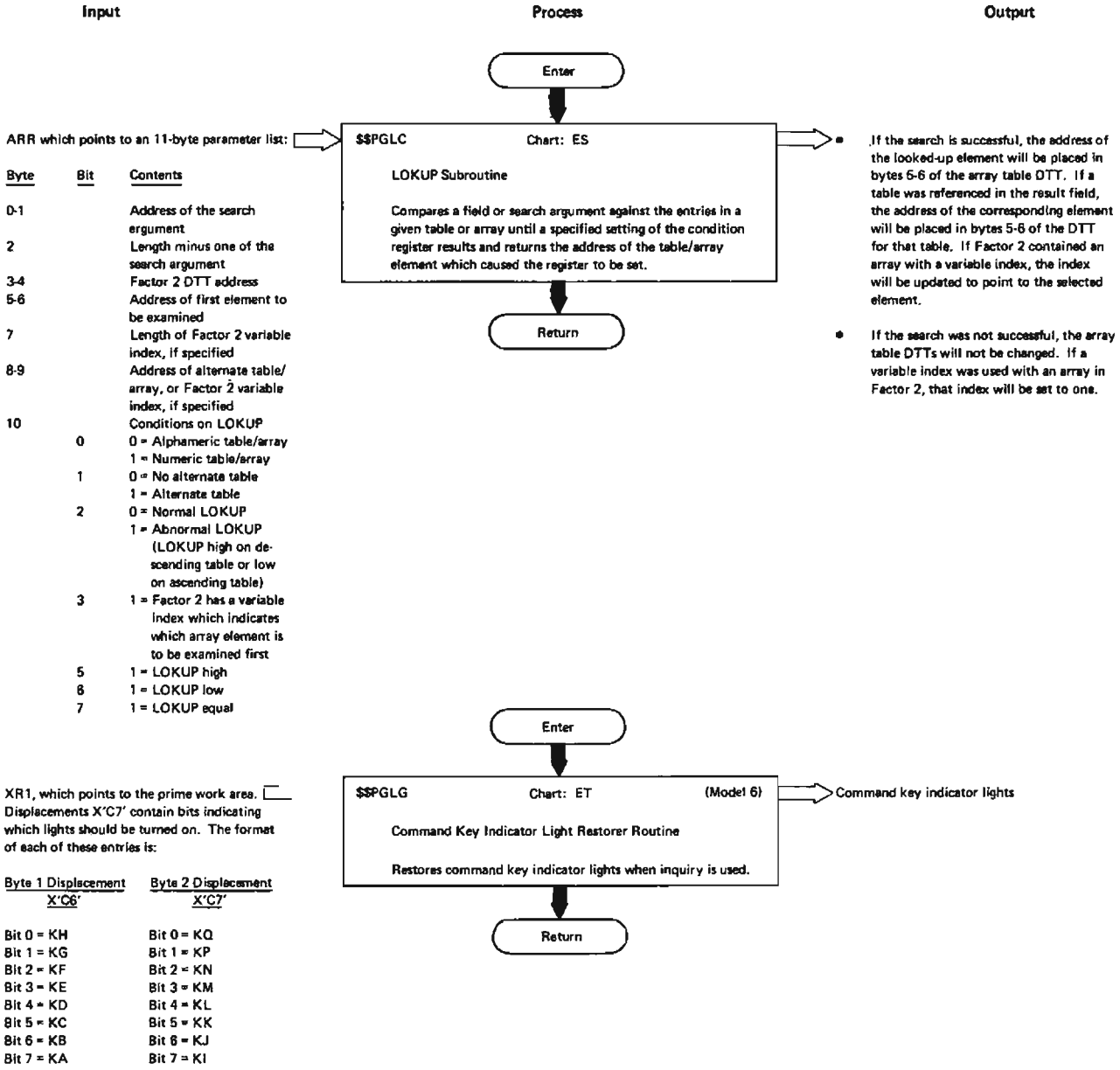


Figure 4-3 (Part 7 of 9). Library of Subroutines

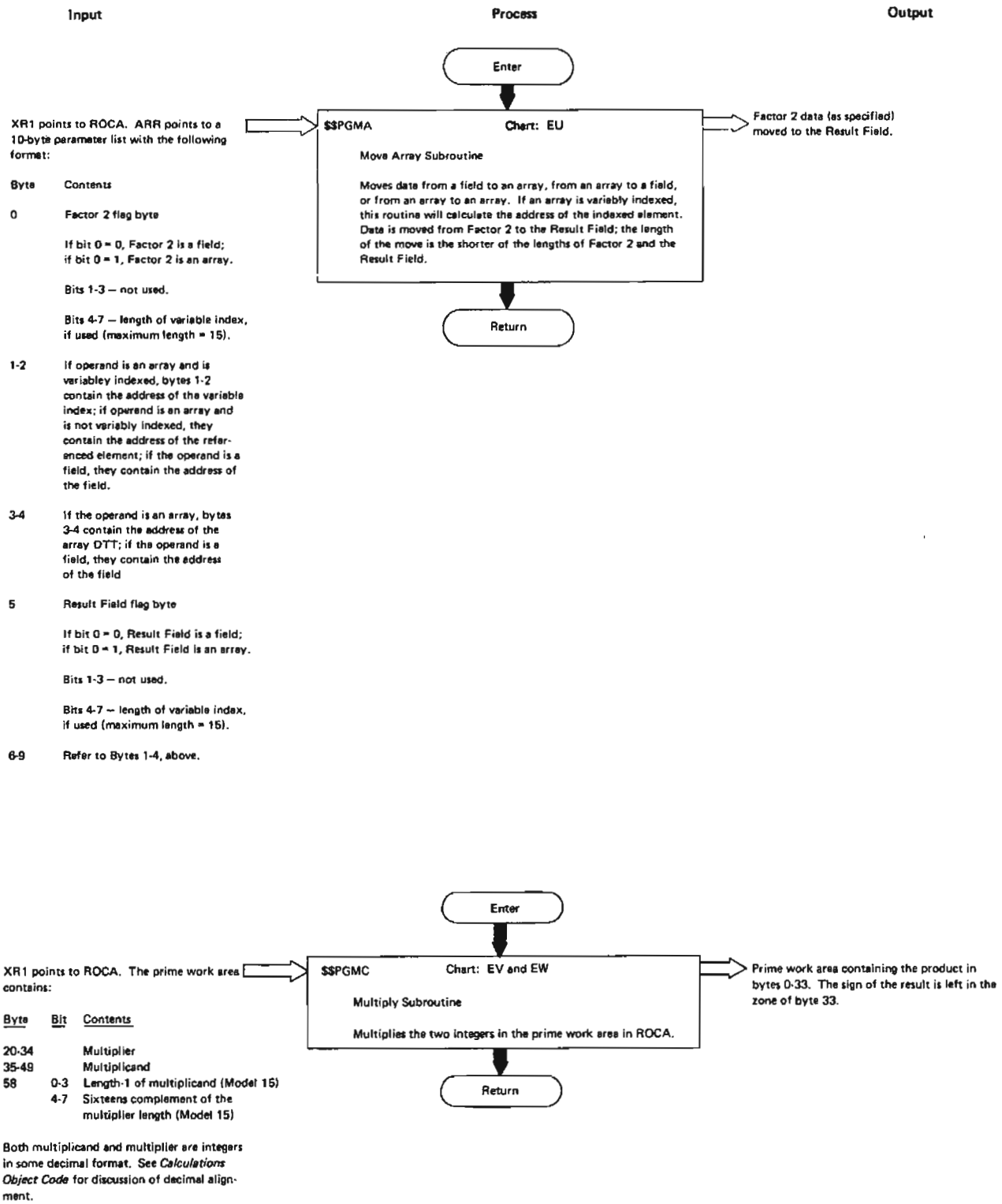


Figure 4-3 (Part 8 of 9). Library of Subroutines

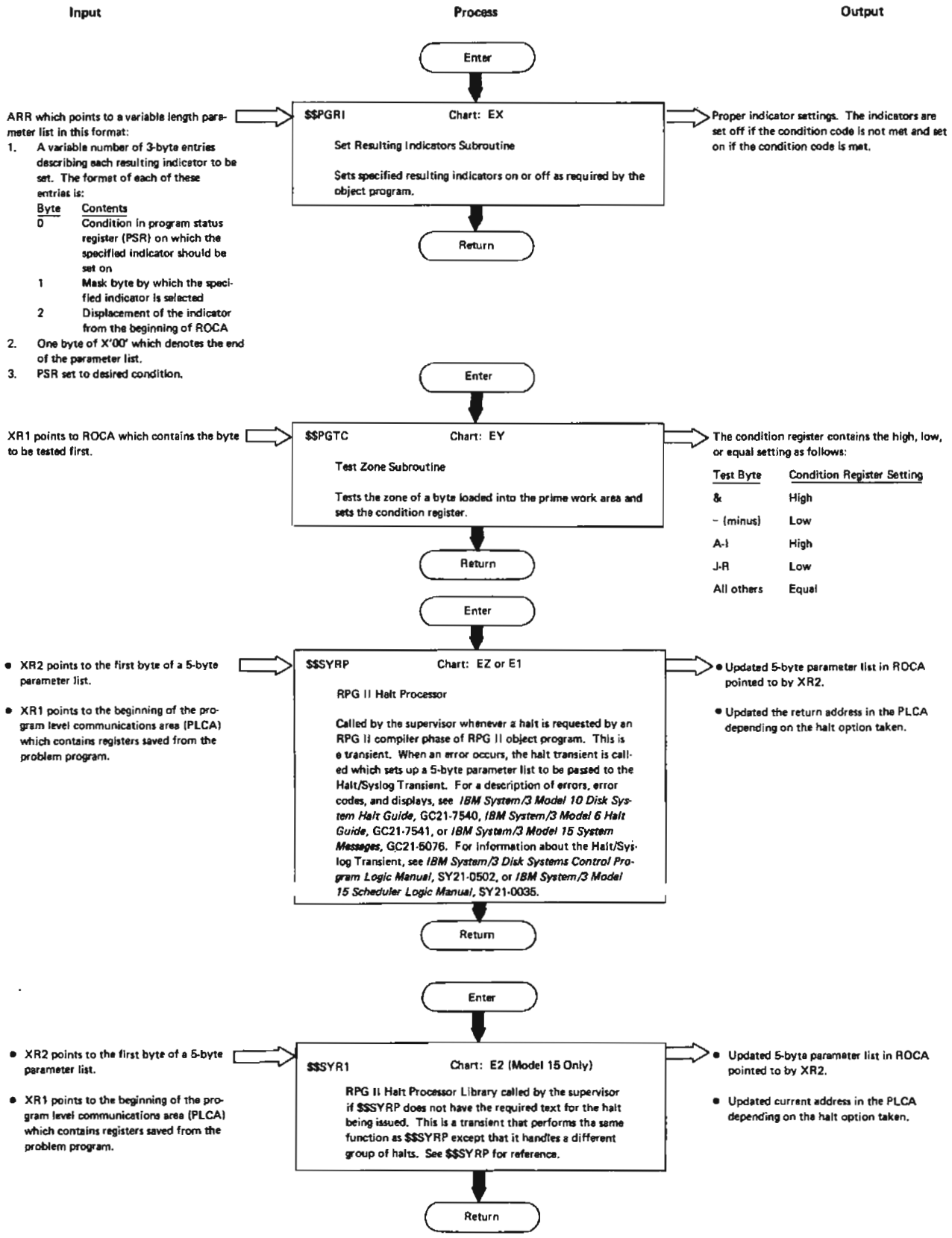


Figure 4-3 (Part 9 of 9). Library of Subroutines

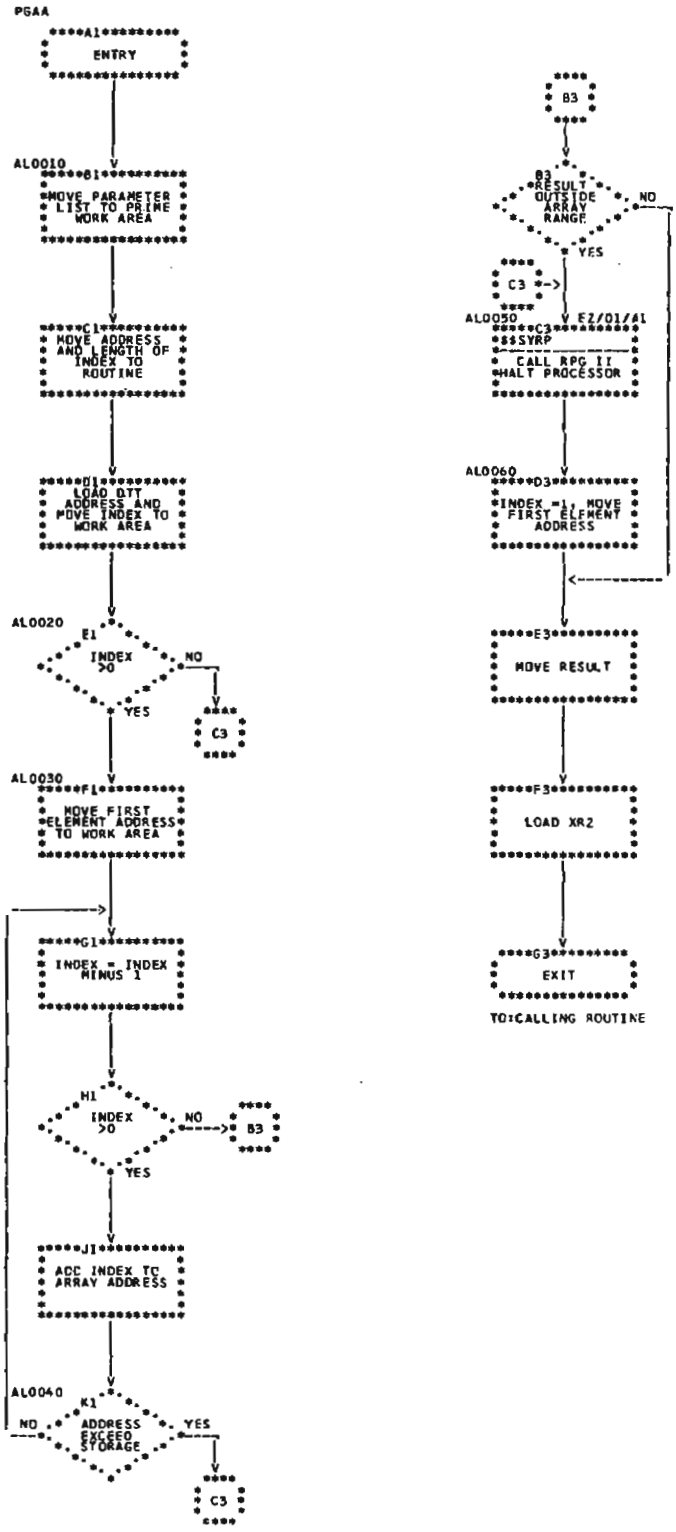


Chart EA. Array Index Subroutine

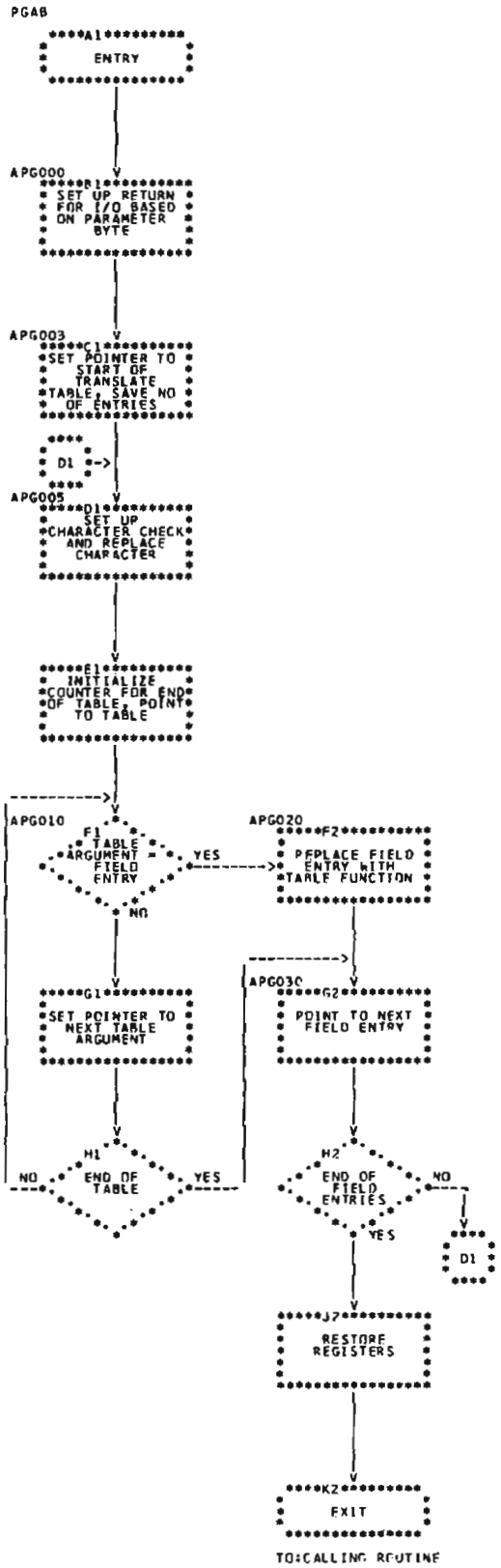


Chart EB. File Translate Subroutine

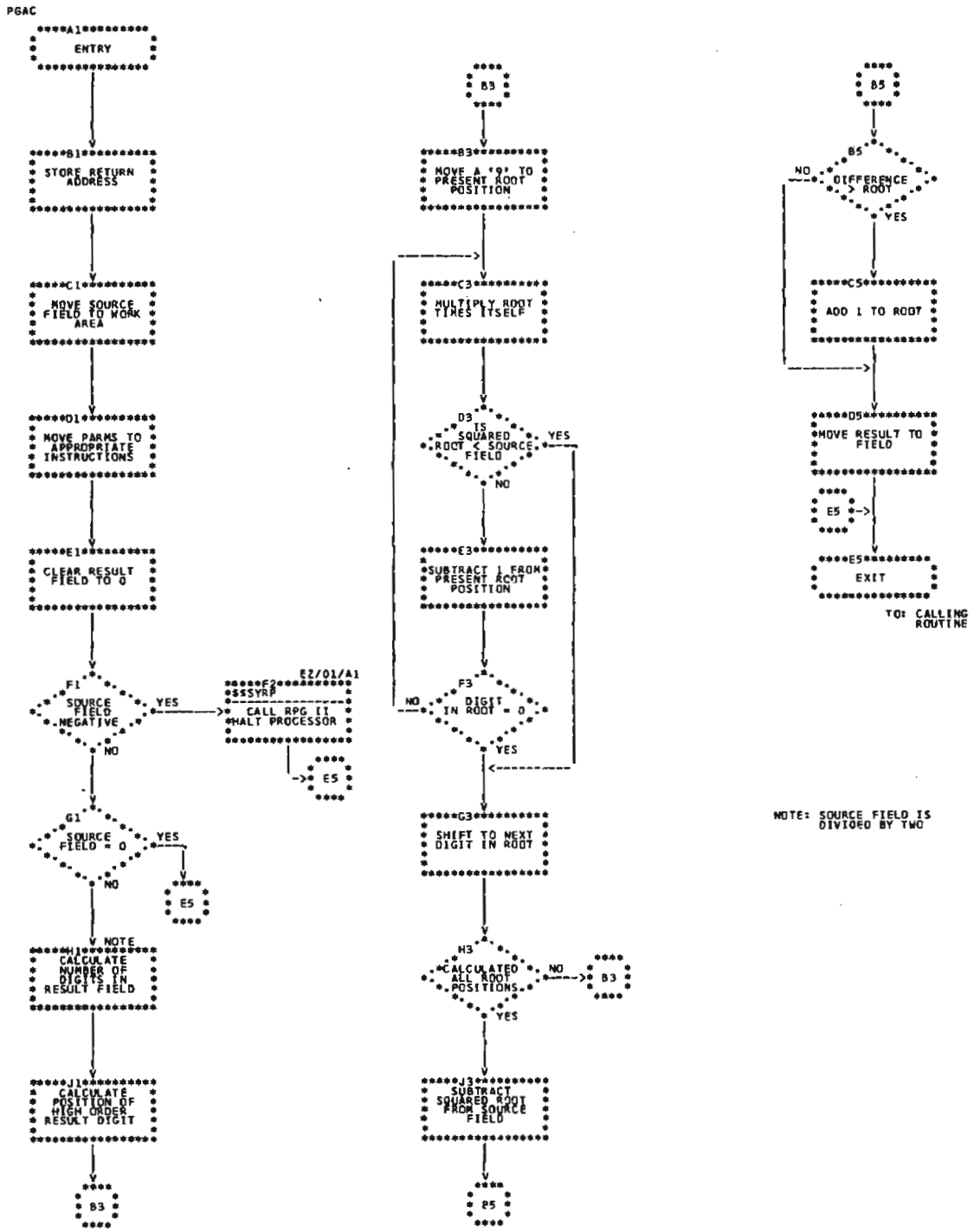


Chart EC. Square Root Subroutine

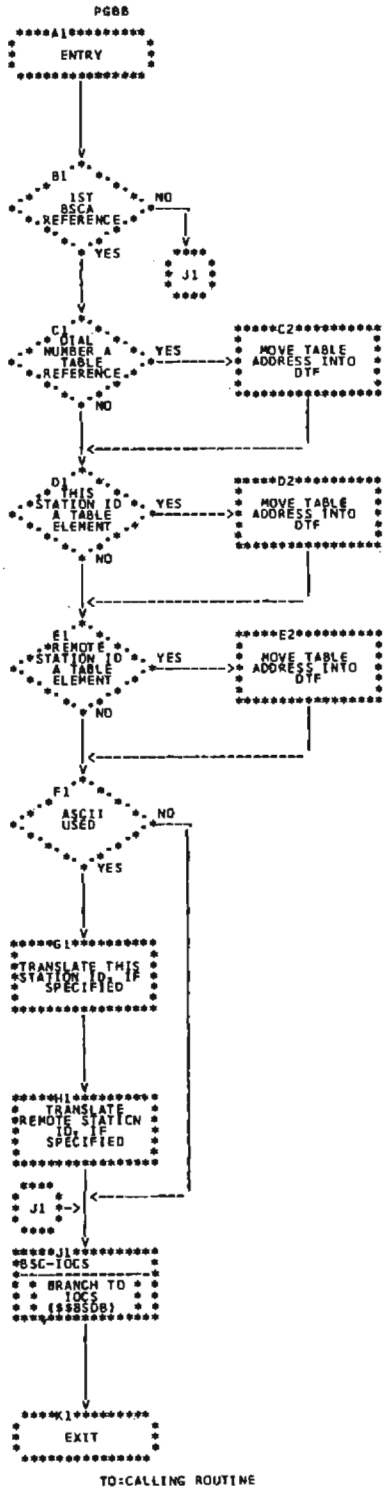


Chart ED. BSCA Logic for Transmit and Receive

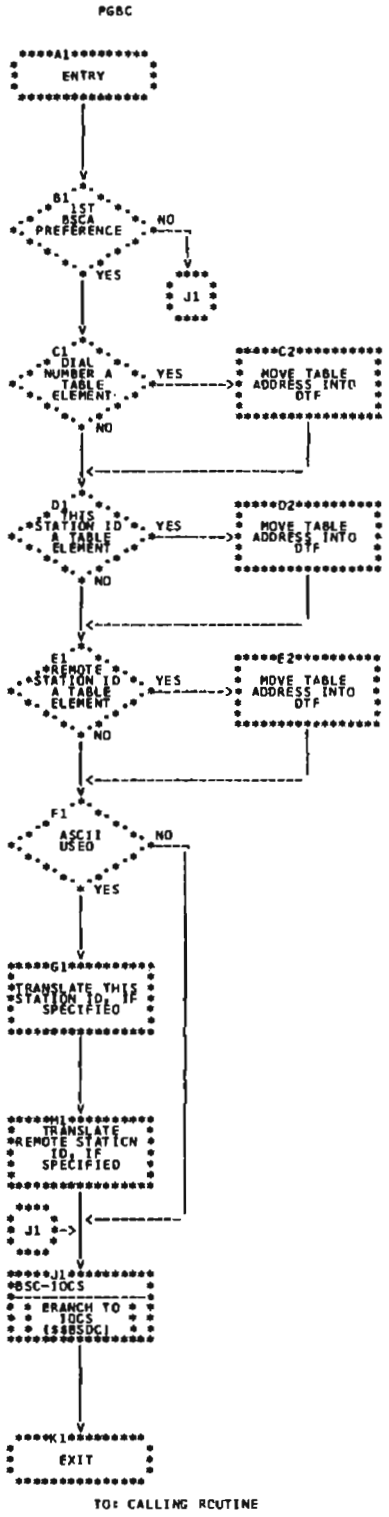


Chart EE. BSCA Logic for Conversational

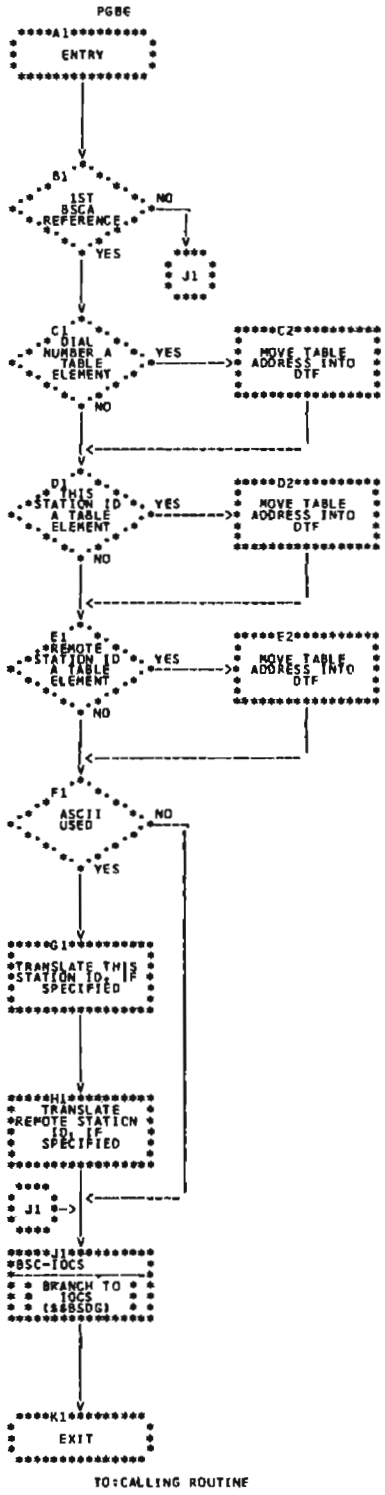


Chart EF. BSCA Logic for Receive

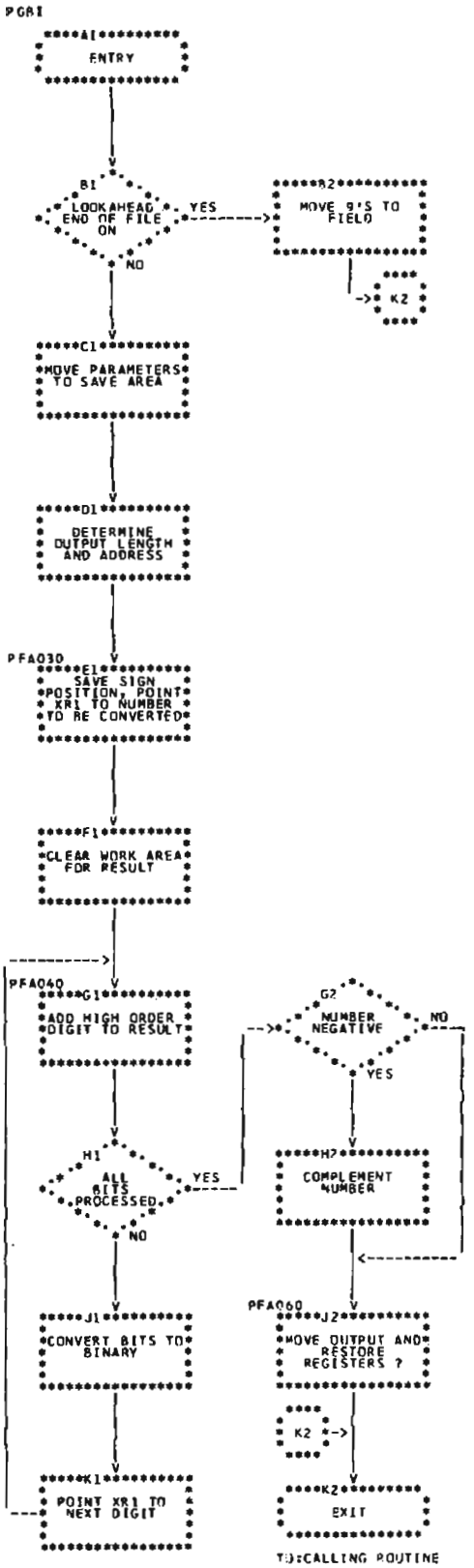


Chart EG. Convert to Binary Subroutine

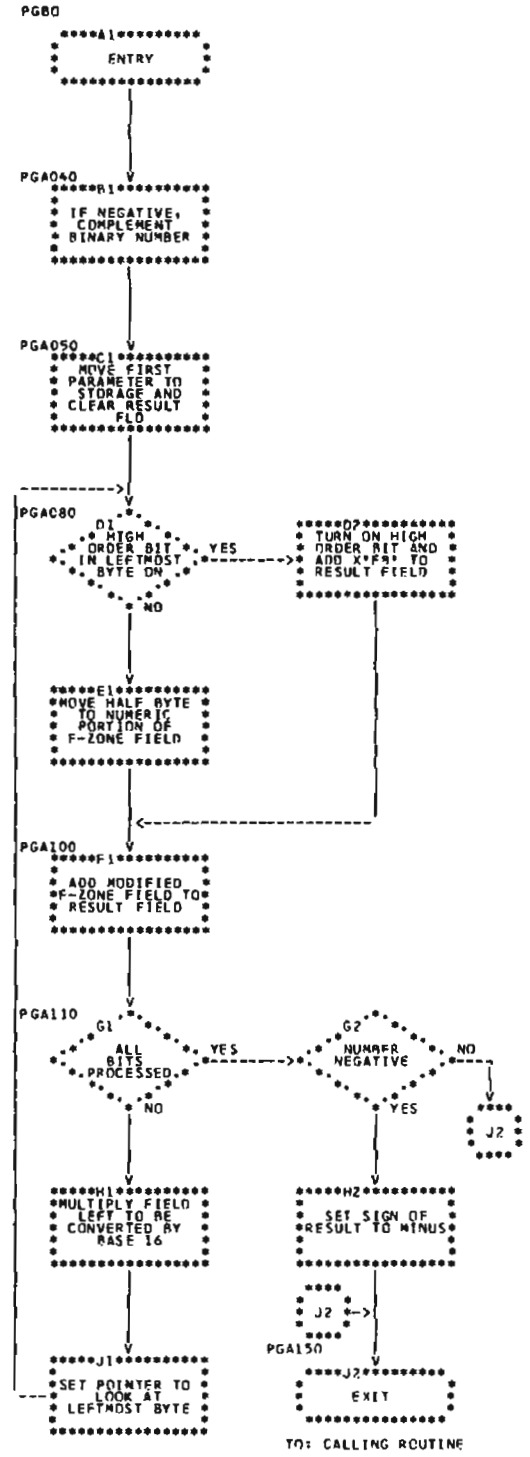


Chart EH. Convert to Decimal Subroutine

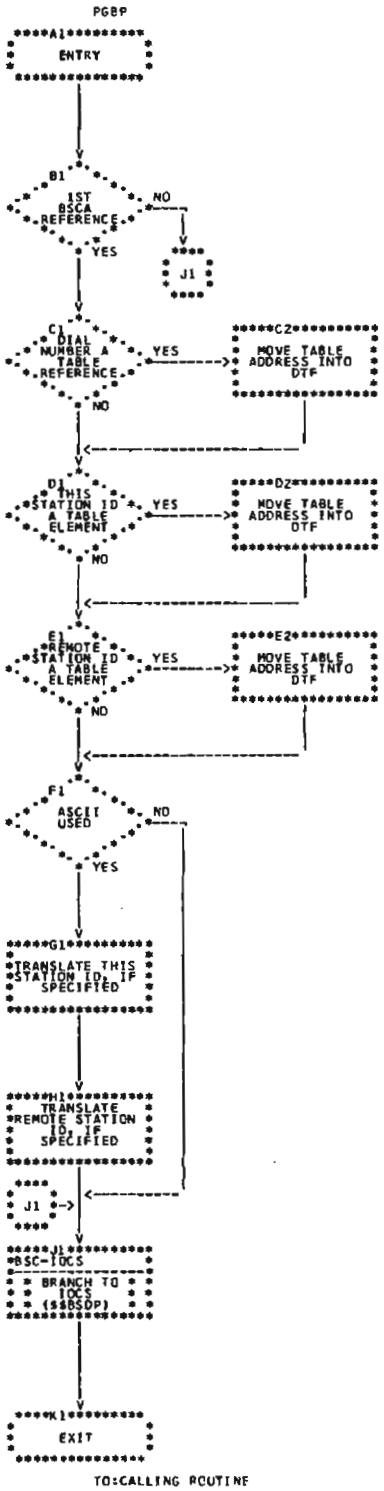


Chart EI. BSCA Logic for Transmit

Chart EJ. Command Key Indicator Set Routine (Model 6)

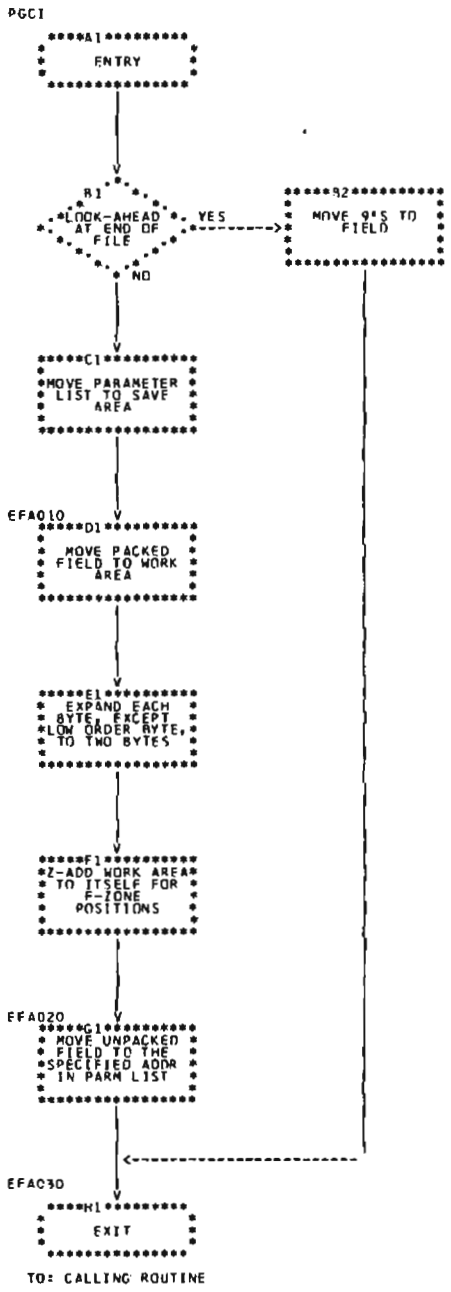


Chart EK. Unpack Subroutine

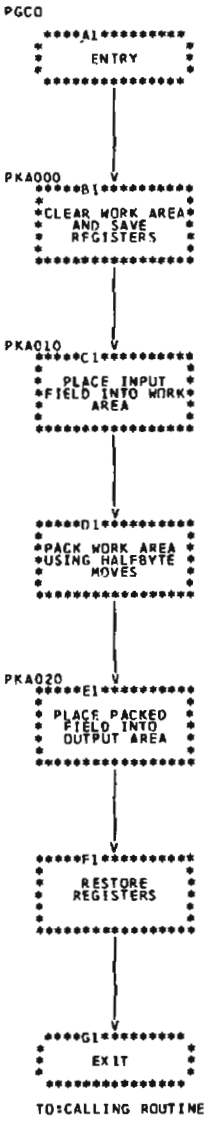


Chart EL. Pack Subroutine

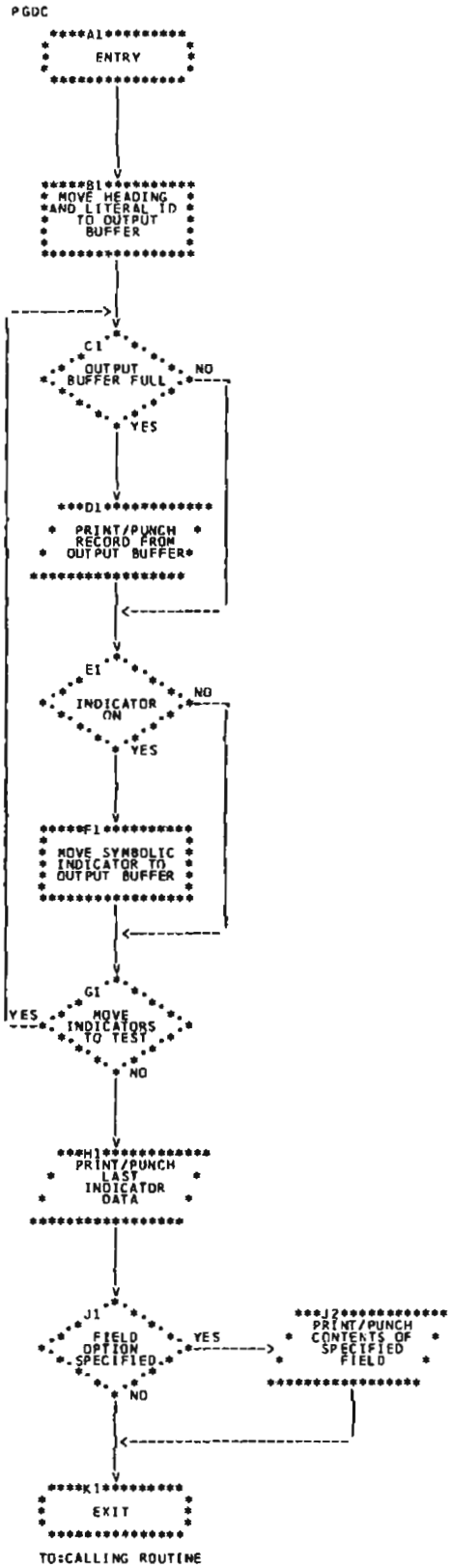


Chart EM. DEBUG Subroutine

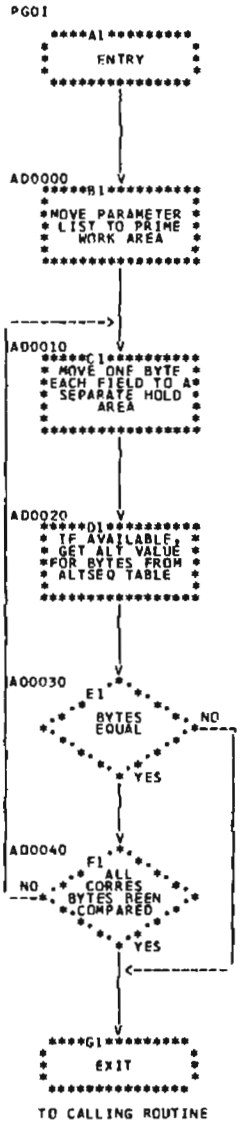


Chart EN. Alternate Collating Sequence Subroutine

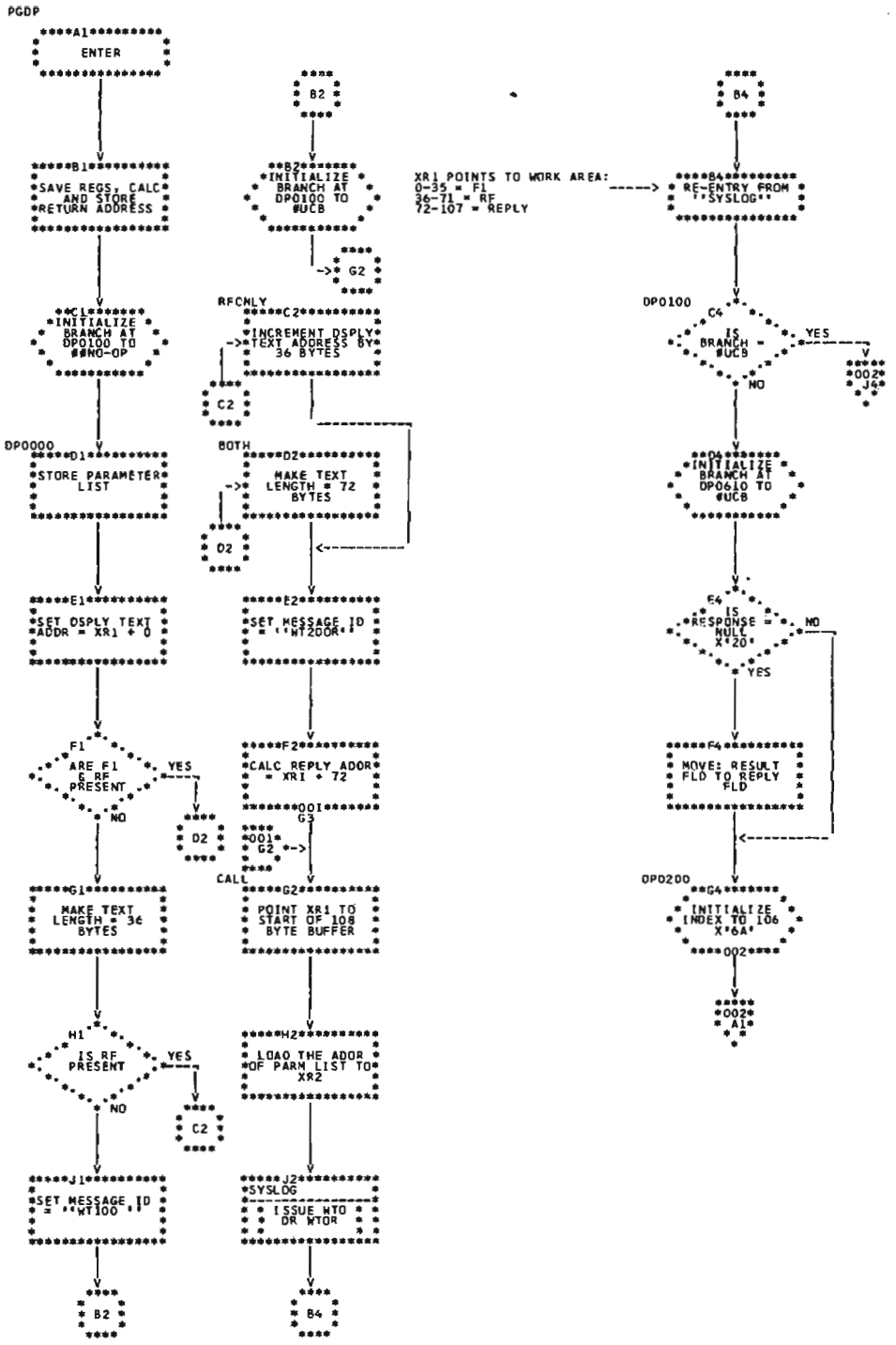


Chart EO (Part 1 of 2). Display Subroutine (Model 15)

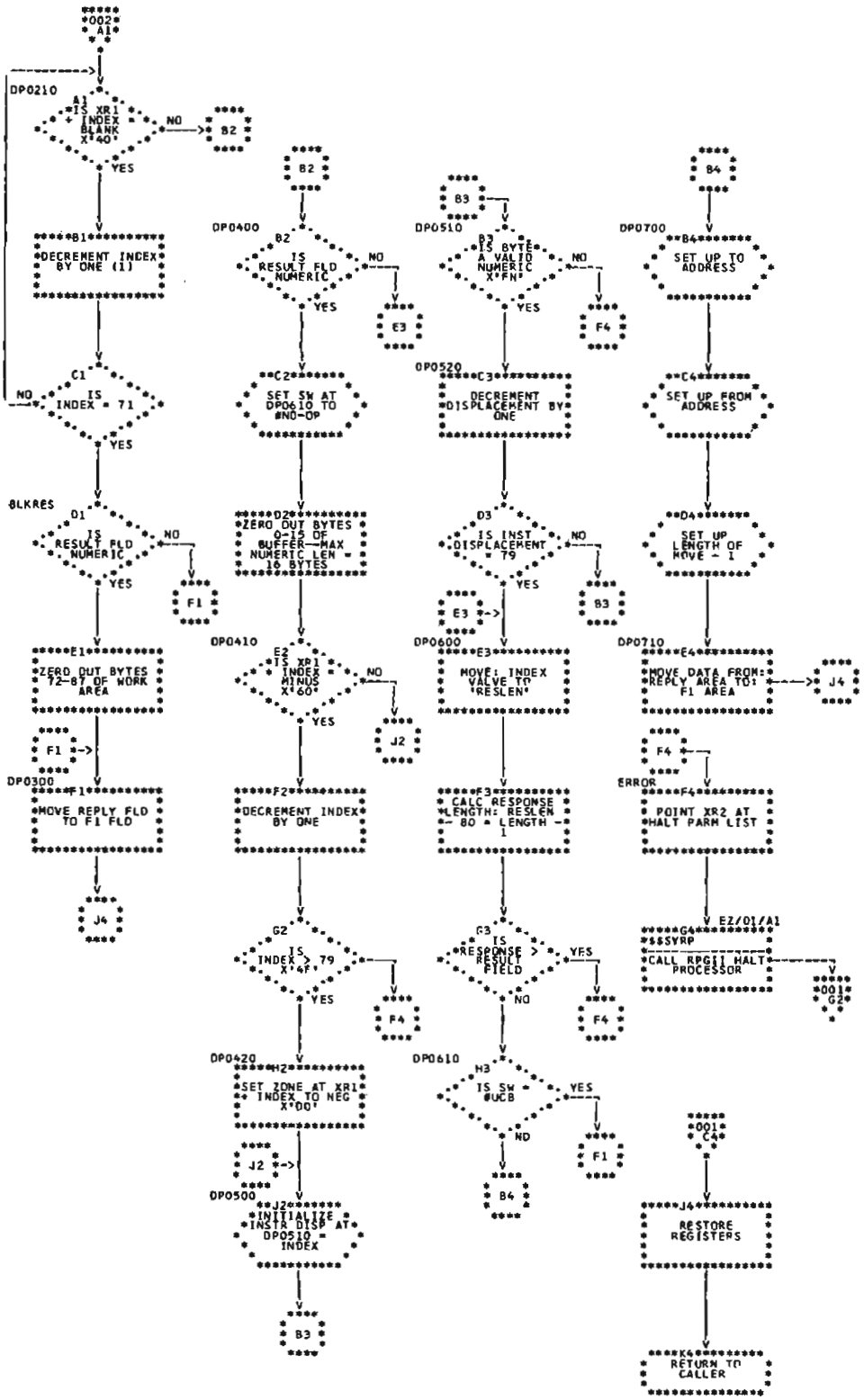


Chart EO (Part 2 of 2). Display Subroutine (Model 15)

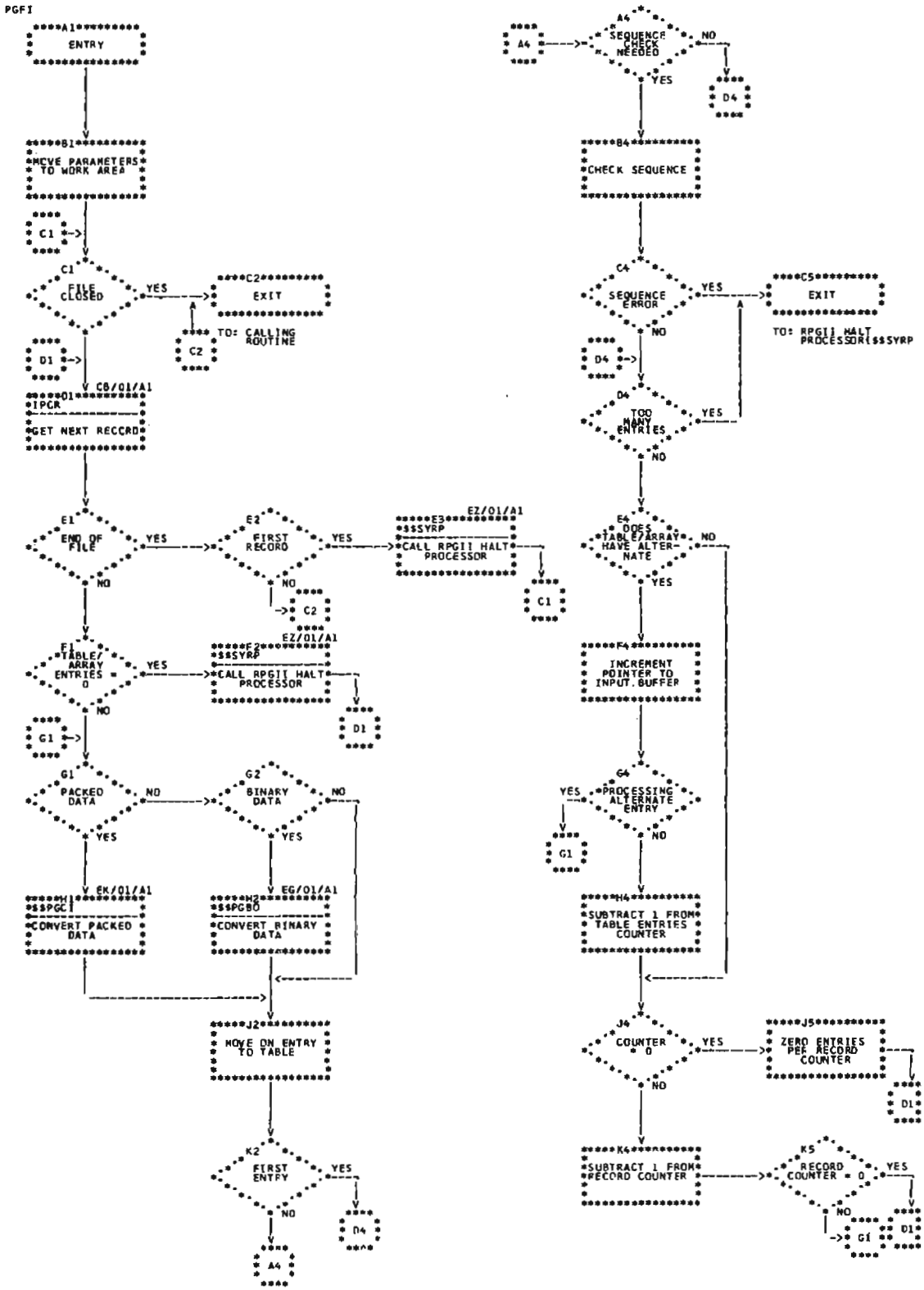


Chart EP. Load Object Tables Subroutine

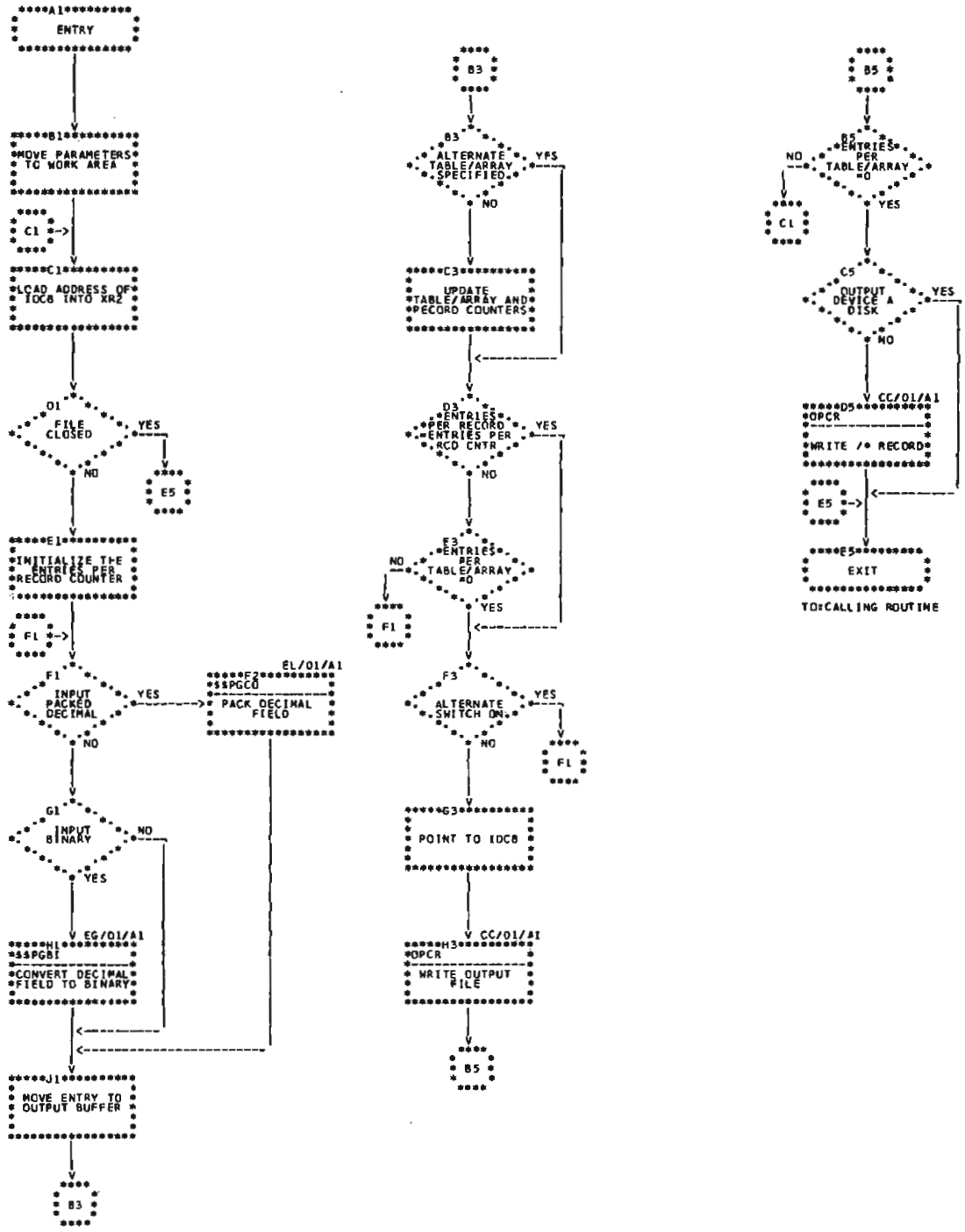


Chart EQ. Dump Object Table Subroutine

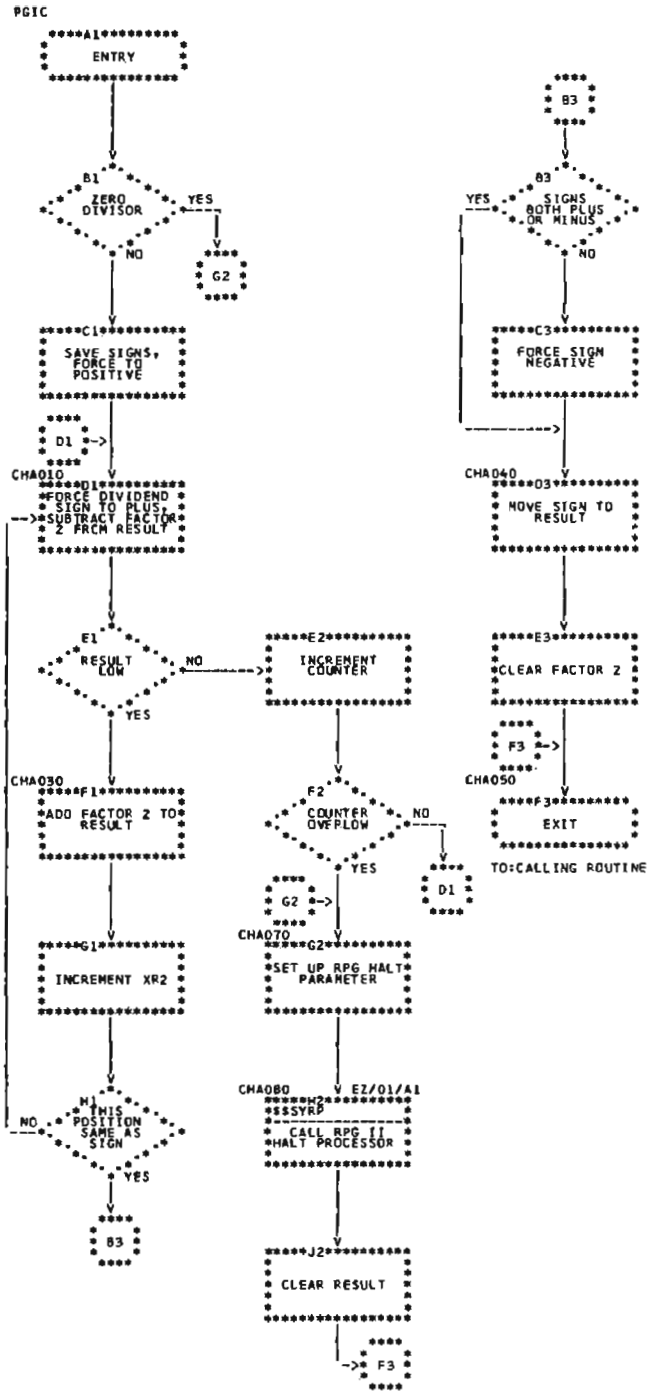


Chart ER. Divide Subroutine

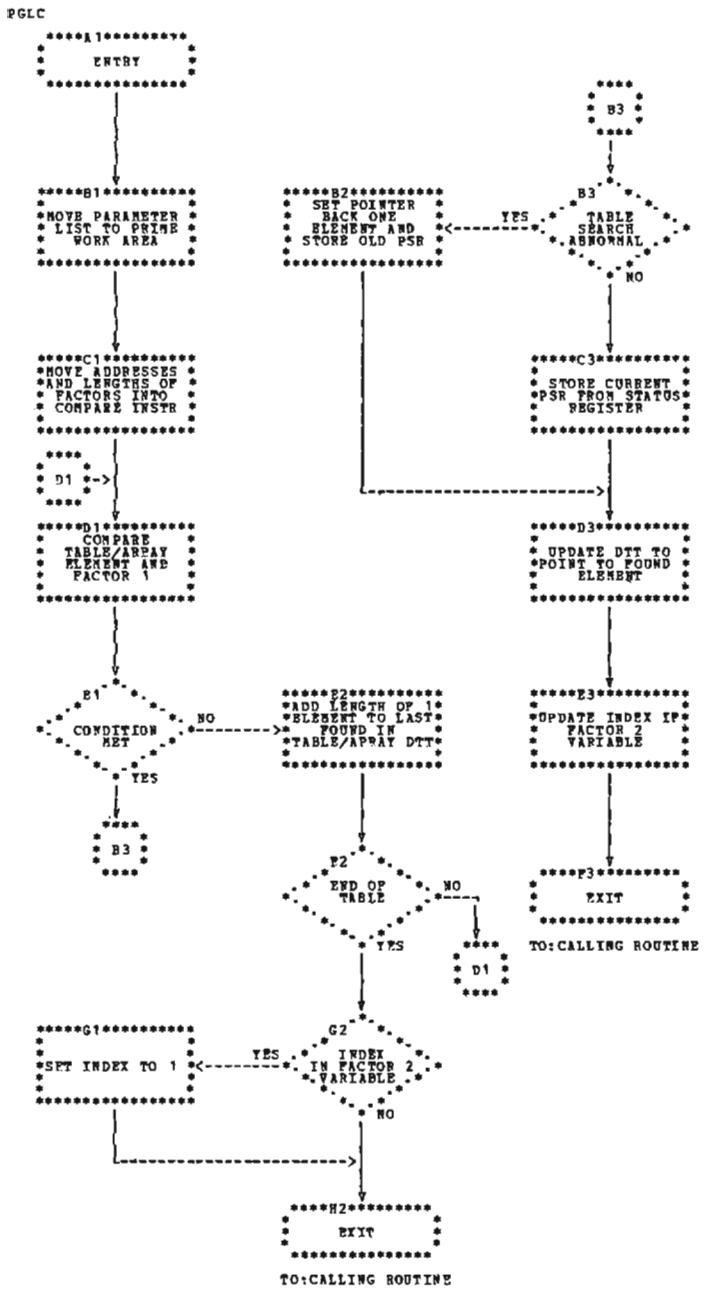


Chart ES. LOKUP Subroutine

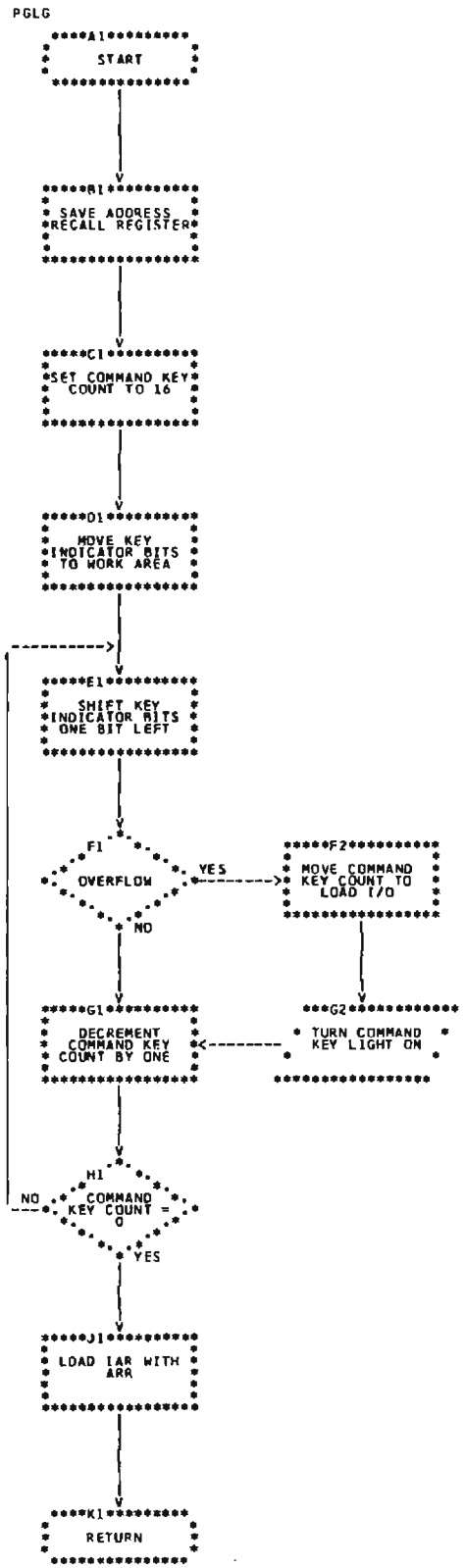


Chart ET. Command Key Indicator Light Restorer Routine (Model 6)

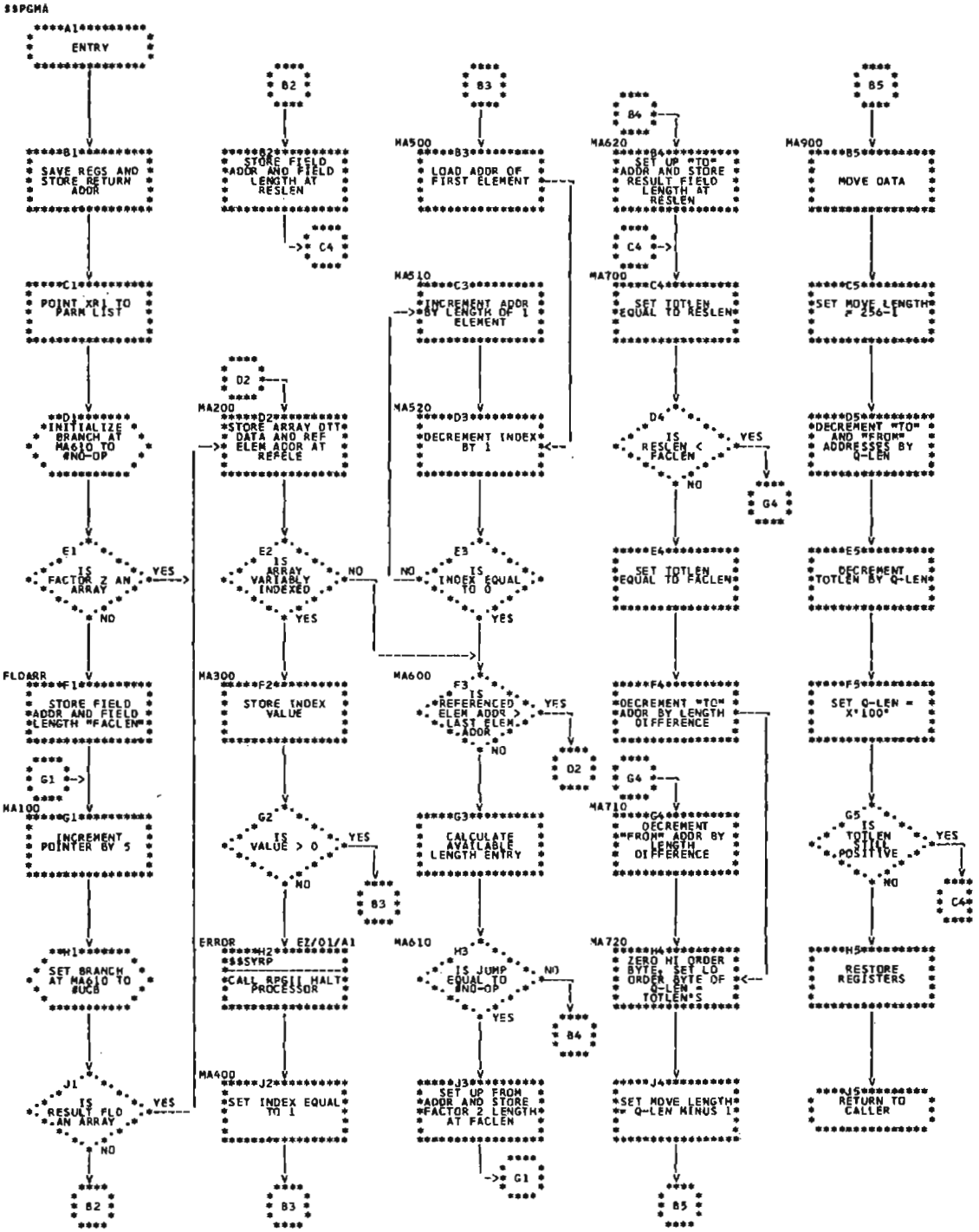


Chart EU. Move Array

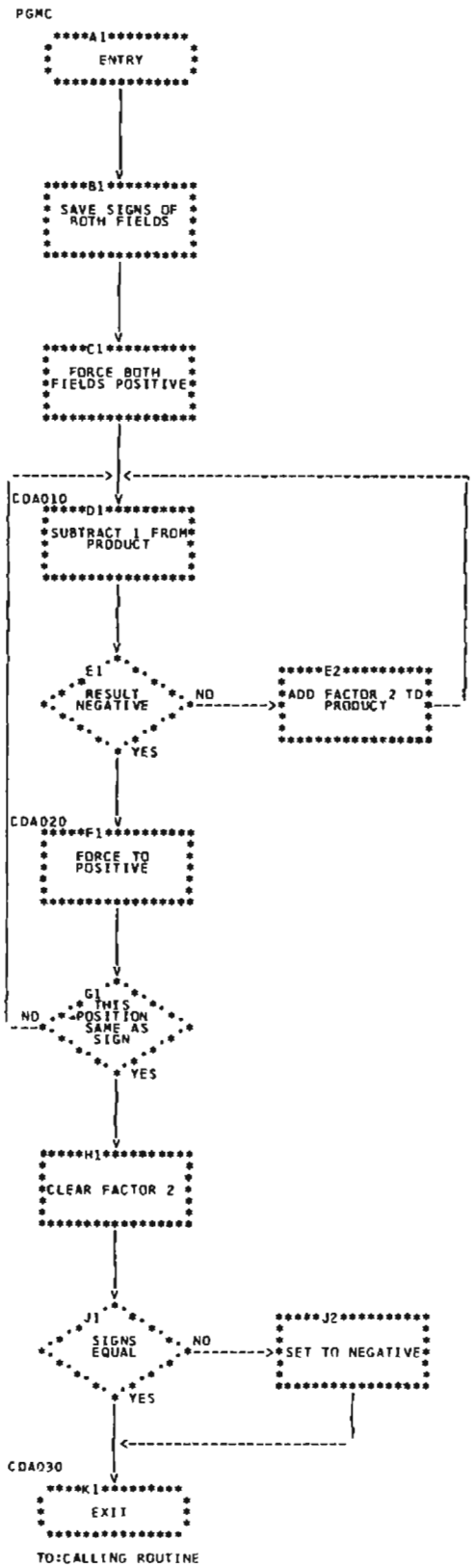


Chart EV. Multiply Subroutine (Models 6, 10, and 12)

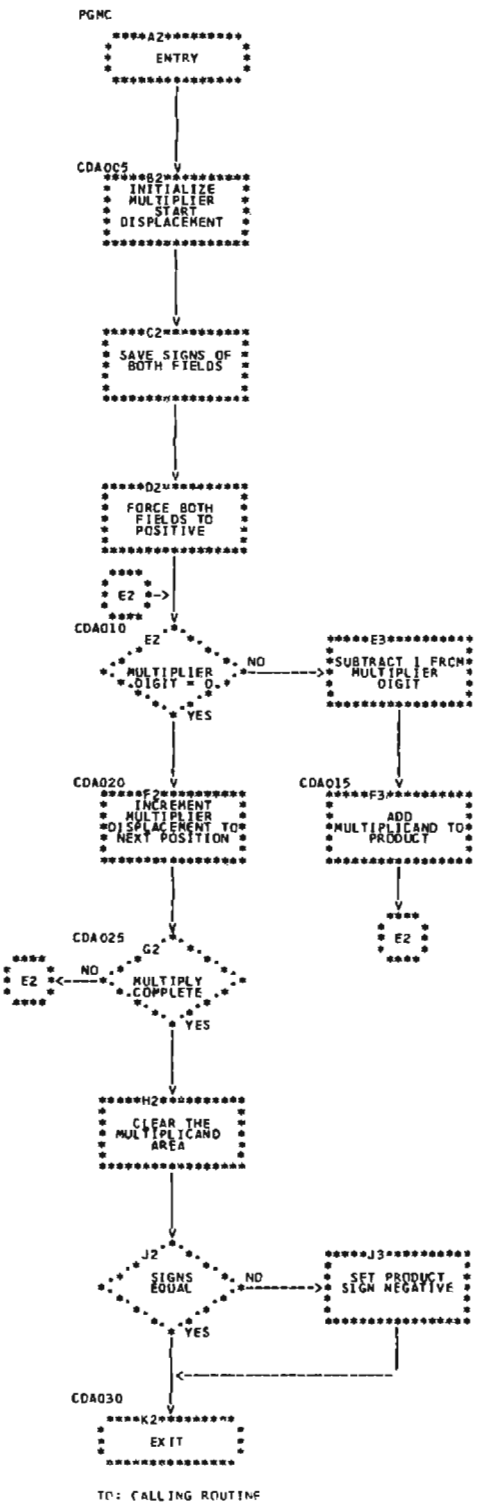


Chart EW. Multiply Subroutine (Model 15)

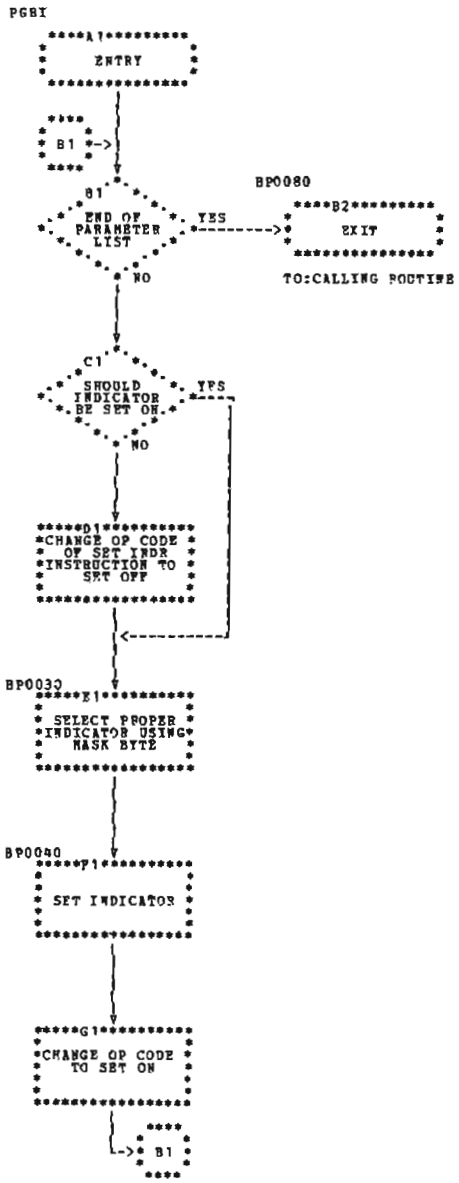


Chart EX. Set Resulting Indicators Subroutine

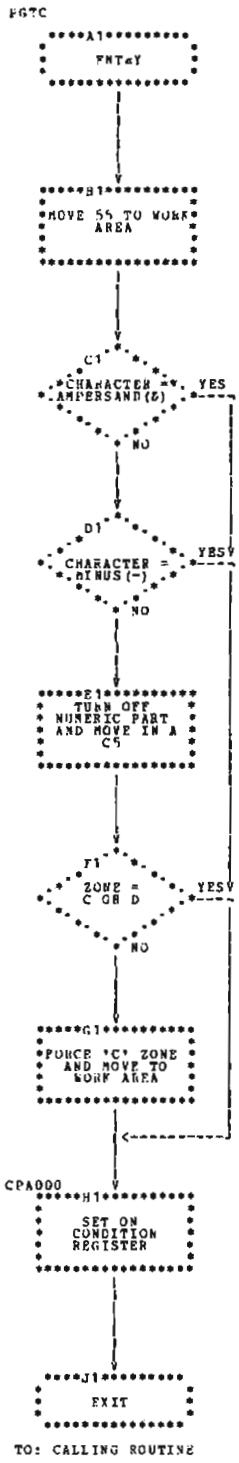


Chart EY. Test Zone Subroutine

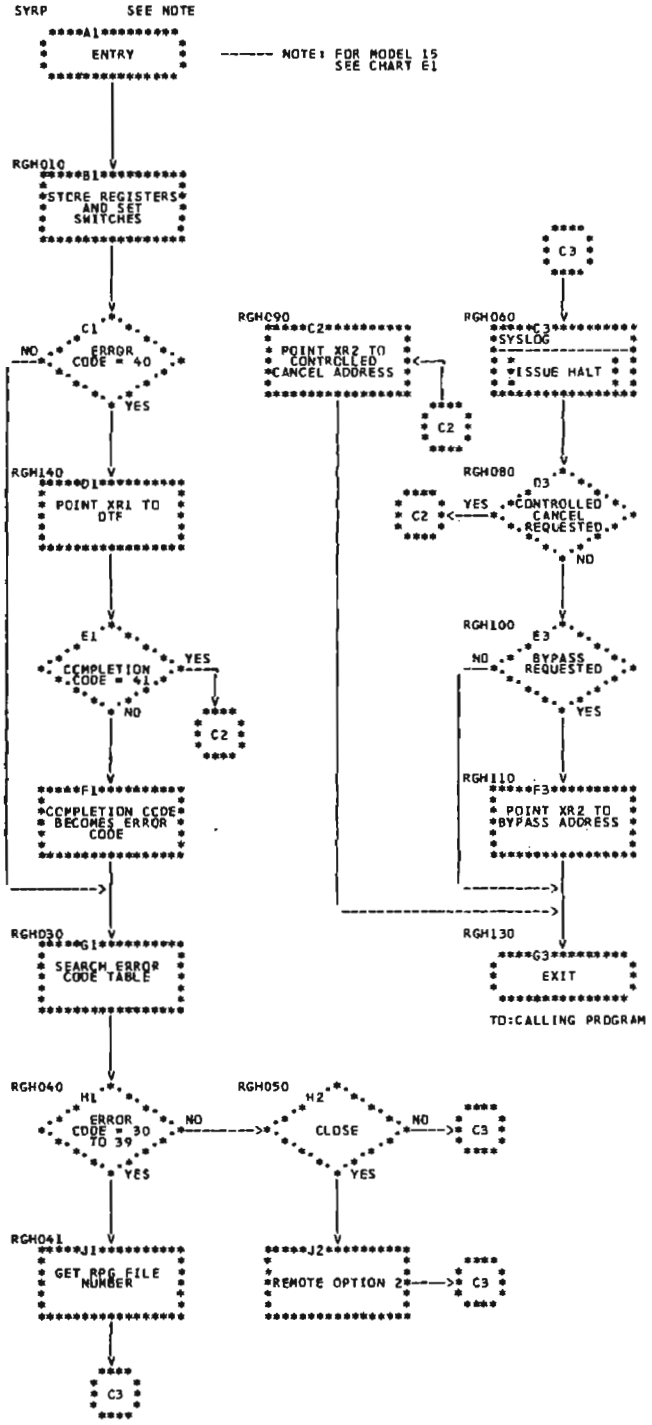


Chart EZ. RPG Halt Transient (Models 6, 10, and 12)

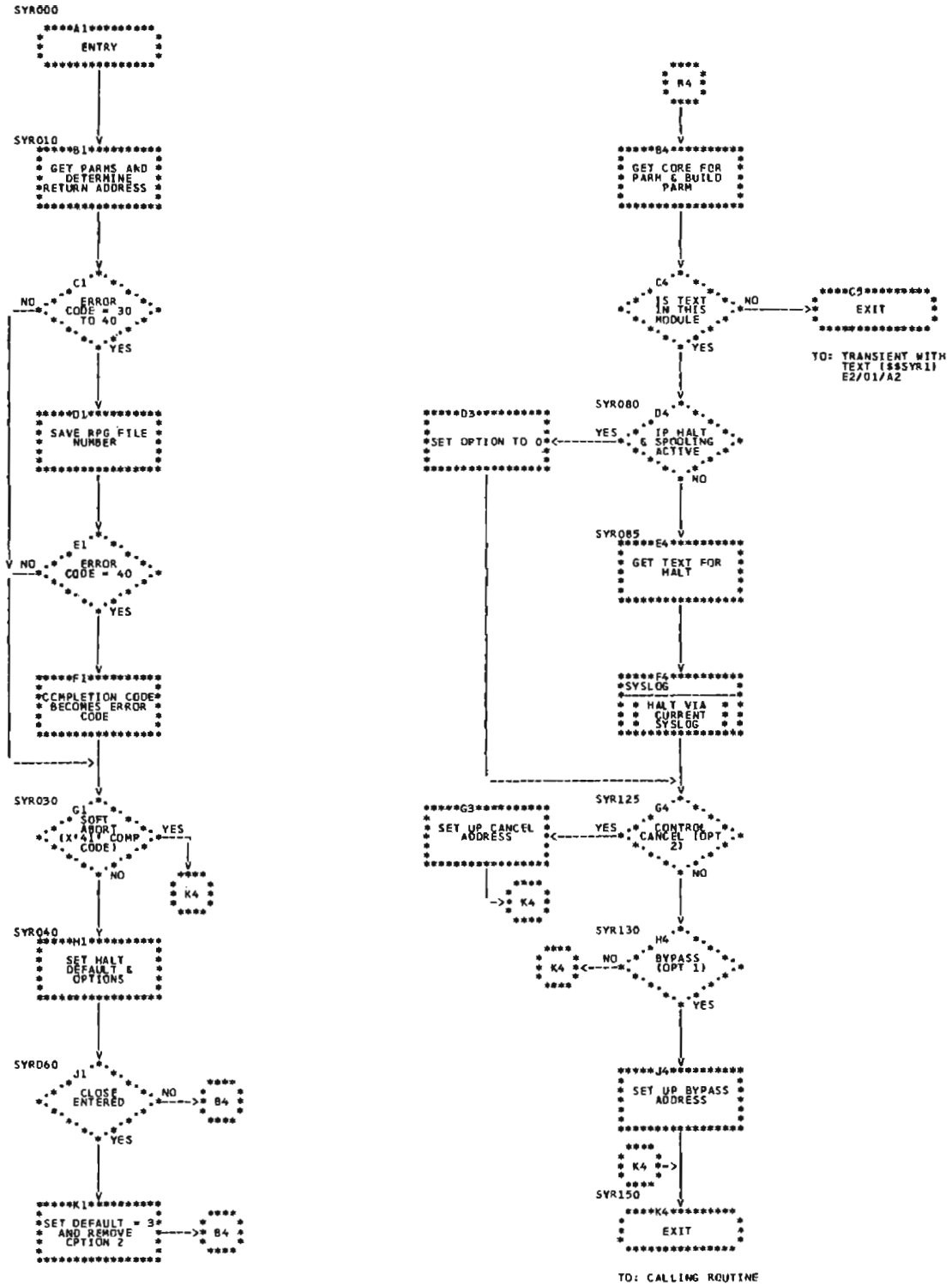


Chart E1. RPG II Halt Transient (Model 15)

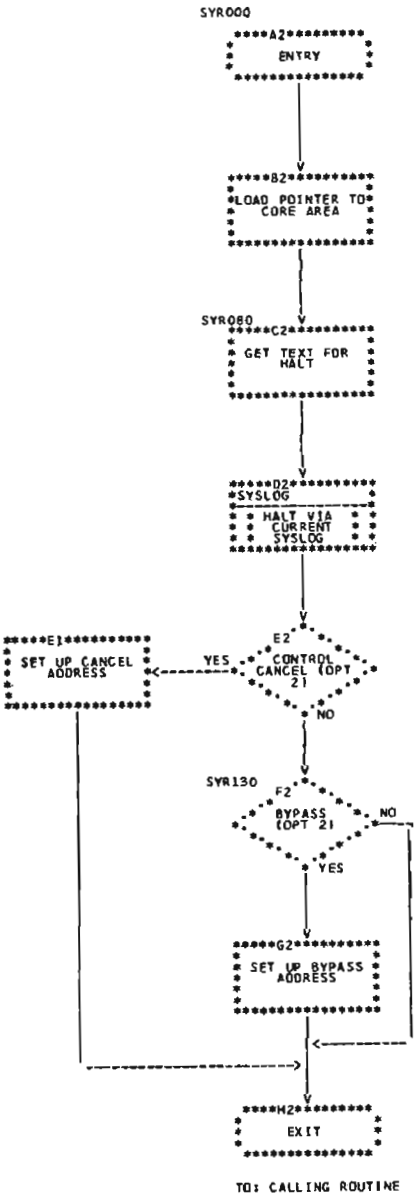


Chart E2. RPG II Halt Transient Library 1 (Model 15)

DATA AREAS

This section describes the layout and contents of data areas used by the object program. Although lengths and locations of data areas vary with the source program, absolute addresses and fixed lengths of data areas are given where appropriate.

Reserved Object Communications Area (ROCA)

The reserved object communications area is the first 256 bytes of the Root Segment for every RPG II program. Only part of ROCA is actually generated since most of it is composed of work areas.

Prime Work Area

This 144-byte work area is located in the beginning of ROCA from X'00' to X'8F'. The following blocks of object code use the prime work area:

Object Code	*Bytes Used*
TESTZ	X'00'-X03'
Z-ADD	X'00'-X'0F'
ADD	X'00'-X'19'
Z-SUB	X'00'-X'19'
SUB	X'00'-X'19'
Multiply	X'00'-X'32' (Models 6, 10, and 12)
	X'00'-X'3A' (Model 15)
Divide	X'00-X'3D'
SQRT	X'00'-X'3D'
LOKUP	X'00'-X'3D'
Control fields	X'00'-X'8F'
Matching fields	X'00'-X'8F'
Chain packed keys	X'07'-X'0F'
MVR	X'1B'-X'2A'
Table load	X'60'-X'83'
*PRINT	X'00'-X'8F'
TIME	X'00'-X'0C' (Model 15 only)

Constant Area

The 50-byte constant area follows the prime work area in ROCA. This area is located from X'90' to X'C1'. Compiler phase $RPGH builds the constant area and phases $RPPC and $RPPO make additions to it. The constant area contains:

Byte	*Contents*
X'90'-X'97'	Constants used by many routines: X'40FFFF0000010002'
X'98'-X'99'	Address of first IOCB in the IOCB chain
X'9A'-X'9B'	Address of IOCB of file currently being processed
X'9C'-X'9D'	Address of IOCB of a forced file
X'9E'-X'9F'	Address of first table load DTF
X'A0'-X'A1'	1P save area containing the error restart address for the First Page Output routine
X'A2'-X'A7'	UDATE in user-specified format
X'A2'-X'A3'	UMONTH (UDAY if using European method of notation)
X'A4'-X'A5'	UDAY (UMONTH if using European method of notation)
X'A6'-X'A7'	UYEAR
X'A8'-X'A9'	Month and day of compile
X'AA'-X'AB'	Time of compile (HHMM Model 15 only)
X'AC'-X'AF'	Branch to controlled cancel
X'B0'-X'B3'	Branch to Input Mainline
X'B4'-X'B8'	Halt parameters used by the RPG II Halt Processor X'B4' = X'43' X'B5' = X'40' X'B6' - X'B7' = Console stick-light mask (see Halt/Syslog, *IBM System/3 Disk Systems System Control Program Logic Manual*, SY21-0502) X'B8' = Acceptable restart options
X'B9'-X'BA'	Address of alternate collating sequence table
X'BB'-X'BD'	Constants used by output routines 'CR*'
X'BE'-X'BF'	Address of ROCA
X'C0'-X'C1'	Reserved

Note: The Dump Object Tables subroutine may use bytes X'9E'-X'AB' to receive inline code for use by the subroutine.

Indicator Table

The indicator table is located in ROCA at bytes X'C2'-X'D9' immediately following the constant area. Each possible RPG II indicator is assigned a location in the 26-byte table (Figure 4-4).

Command key indicators (Model 6 only)

Displacement from XR1	Hex Byte Mask 80	40	20	10	08	04	02	01
C2	H4	H3	H2	H1		MR (Int.)	MR (Ex.)	1P
C3	L1	L0	LR	H9	H8	H7	H6	H5
C4	L9	L8	L7	L6	L5	L4	L3	L2
C5	U1	U2	U3	U4	U5	U6	U7	U8
C6	KH	KG	KF	KE	KD	KC	KB	KA
C7	KQ	KP	KN	KM	KL	KK	KJ	KI
C8								
C9	07	06	05	04	03	02	01	
CA	15	14	13	12	11	10	09	08
CB	23	22	21	20	19	18	17	16
CC	31	30	29	28	27	26	25	24
CD	39	38	37	36	35	34	33	32
CE	47	46	45	44	43	42	41	40
CF	55	54	53	52	51	50	49	48
D0	63	62	61	60	59	58	57	56
D1	71	70	69	68	67	66	65	64
D2	79	78	77	76	75	74	73	72
D3	87	86	85	84	83	82	81	80
D4	95	94	93	92	91	90	89	88
D5					99	98	97	96
D6	OV Ex.	OG Ex.	OF Ex.	OE Ex.	OD Ex.	OC Ex.	OB Ex.	OA Ex.
D7	OV 1st Int.	OG 1st Int.	OF 1st Int.	OE 1st Int.	OD 1st Int.	OC 1st Int.	OB 1st Int.	OA 1st Int.
D8	OV 2nd Int.	OG 2nd Int.	OF 2nd Int.	OE 2nd Int.	OD 2nd Int.	OC 2nd Int.	OB 2nd Int.	OA 2nd Int.
D9	Total cycle switch	Control fields processed	Overflow being processed	EOF on look-ahead	Close has been entered	* * RESERVED		

Note: For each overflow indicator there are two internal indicators. The first internal indicator indicates that overflow has occurred; the second indicator indicates that the overflow output code has been fetched.

Ex. = External
Int. = Internal

Figure 4-4. Indicator Table

Secondary Work Area

Byte X'DA' is not used. Byte X'DB' contains the modification level of the compiler that generated the program. Byte X'DC' contains the release (version) number of the compiler that generated the program. The remaining 34 bytes in ROCA are used by the subroutines as a work area. The bytes from X'DD' to X'FF' are used as follows:

Subroutine	*Bytes Used*
Load Object Tables	X'DD'-X'EC'
Dump Object Tables	X'B9'-X'D0'
	X'DA'-X'EA'
DTF Parameter Save Area For Fetch	X'DD'-X'E2'
Unpack	X'E7'-X'FF'
Convert to Binary	X'E8'-X'FF'
Command Key Indicator Set Routine	X'EA'-X'ED'
Alternate Collating Sequence	X'EA'-X'FF'
Convert to Decimal	X'EC'-X'FF'
Pack	X'ED'-X'FF'
File Translate	X'F1'-X'FF'
Array Index	X'F3'-X'FF'
Set Resulting Indicators	X'FA'-X'FF'

Trailer Table

This 8-byte table is generated by phase $RPJS for each valid trailer record (TR) specification group found in the input specifications. The table format is:

Byte	*Contents*
0-1	Displacement in record to the low order end of the first trailer
2-3	Displacement to the low order end of the last trailer in group
4-5	Displacement to current trailer (initially first)
6-7	Actual length of one trailer

Define the Table (DTT)

One 8-byte DTT is associated with each array or table in the object program. The DTT address may be found in the source listing. Each DTT is in the following format:

Byte	*Contents*
0-1	Address of rightmost byte of the first array/table element
2-3	Address of rightmost byte of the last array/table element
4-5	Address of rightmost byte of last looked-up element if a table (used for work area if an array) *Note:* These bytes are initially the same as bytes 0-1.
6-7	Length of one element

Define the File (DTF)

The DTF is the primary external interface to a program calling an access method. Compile-time phase $RPGN builds a pre-open DTF for each device specified. The pre-open DTF is passed to data management. During program open, data management modifies the pre-open DTF and returns it to RPG II as a post-open DTF. The size of the pre-open DTF built by phase $RPGN for disk files depends on the access method being used to process the file. For a detailed description of pre-open and post-open DTFs, see *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual*, SY21-0512, and/or *IBM System/3 Model 15 System Data Areas and Diagnostic Aids*, SY21-0032.

Alternating Collating Sequence and Translate Tables

The alternate collating sequence and translate tables immediately precede the match field save area at X'BA' in ROCA. These two tables have the same format and similar uses.

The first entry in each table is one byte containing the number of entries in the table. The remaining entries are two bytes each. The first byte is the external value; the second byte is the internal value. Each 2-byte entry corresponds to one user-specified entry (see *Library of Subroutines).*

Match Field Save Areas

Match field save areas are allocated in the Root Segment. The match field save areas can be found by locating the multifile logic code in the source listings. The first compare instruction contains the address of the first match field save area.

If one file is specified as a match file, then one save area is set aside. If two or more files are designated as match files, two match field save areas are allocated in the Root Segment. Each area is only as long as the number of match levels specified. For example, if match field levels M1, M3, and M4 are specified, then the save area would only be large enough to hold 'M1M3M4'.

Phase $RPHS generates and diagnoses the use of the match field save areas. Phase $RPHT generates the initializing object code. The first save area contains the matching values of the last selected file and is used for sequence checking. The second save area contains the match values of the last selected primary file and is used to control the setting of the MR indicator. See *Detailed Object Program Flow, Multifile and Matching Records Logic* for further discussion.

Control Field Save Area

The control field save area is located in the Root Segment immediately following the match field save areas. Control fields are defined on the input specifications. The control field save area can be found by locating the control field logic in the source listing. The address in the second MVC instruction is the address of the save area.

The control field save area is only as long as the sum of all the control levels defined. If there are three control levels, then the control field save area will be only large enough to contain the three levels. The levels are generated in descending order from L9 to L1. Phase $RPHS generates and diagnoses the use of the control fields. Phase $RPHT generates the initializing object code.

Constants, Edit Words, and Edit Codes

Constants and edit words are specified on the output-format specifications and generated as subsegments at compile time by phases $RPLN and $RPLR. When placed in the Root Segment, constants and edit words are optimized. For example, if constants '1 2 3' appear twice in output-format specifications, they are combined and entered in the Root Segment as '1 2 3'. Edit words can be optimized in a similar way.

Edit codes are specified on the output-format specifications. Compiler phase $RPHT generates the edit patterns for the edit codes. Edit code patterns are placed in the Root Segment.

Error Recovery Procedure (ERP) Area

For every device used within the program, a 5-byte error recovery procedure area is set aside in the Root Segment. Data management returns addresses in the ERP area of where to go when an error occurs for each device. Each ERP area immediately follows the corresponding device DTF.

Input and Output Buffers

Main storage contains two buffer areas – an input buffer area and an output buffer area. There is one input buffer for each device type. The input buffer address can be found in the IOCB at post-open time (see *IOCB* in this section).

An output buffer is assigned for each device specified. If the output buffer length is less than 144 bytes and if *PRINT is not used (Model 10 or 12), the output buffer is located in the prime work area in ROCA. At post-open time, the address of the output buffer area may be found in the IOCB.

Completion Codes from Data Management

Data management passes a series of completion codes to RPG II in the post-open DTF at X'0E'. For a READ or GET the following codes are passed:

Code	*Meaning*
40	Normal completion
41	Controlled cancel requested
42	End of file
44*	No record found, out of extent, DU, DG, D0
46*	Duplicate conversation reply requested (BSC)
4B*	Invalid ASCII code
4D*	Invalid call by user
4E*	Programmer lost communication

For a PUT, WRITE, or ADD the following codes are passed:

Code	*Meaning*
40	Normal completion
41	Controlled cancel requested
44	Updating a record not found
46	Conversational reply requested (BSC)
48	Overflow
4B*	Invalid ASCII code
4D*	Invalid call by user
4E*	Programmer lost communication
50*	Key changed
60*	Duplicate add
62*	Add or load out of sequence
64*	Key too low or too high for indexed random multivolume online file.
68*	Sequence ADD to multivolume indexed file because high key missing from previous volume.
70*	End of extent
72*	Key too high for last volume of multivolume indexed file.

* = These completion codes are sent to the RPG Halt Processor Transient ($$SYRP).

For information about the RPG II Halt Processor Transient, see *Library of Subroutines.*

RPG II passes a parameter list for each file device to data management in the DTF starting at X'0F'. The Input Processing Control routine (IPCR) moves the parameters

from the IOCB to the DTF location. See Figures 4-5 and 4-6 for parameter contents. For output, the parameters are moved by the output code to the DTF. For SET/KEY operations, the parameters are moved by the SET/KEY code to the DTF (Model 6 only).

Input/Output Control Block (IOCB)

The input/output control block (IOCB) contains information about files. Compile-time phase $RPGN builds a 17-byte IOCB for each output file and a 38-byte IOCB for each input file. The address of the first IOCB may be found at X'99' of ROCA. IOCBs are chained together with the address of the next IOCB location at bytes 2-3 of each IOCB. The chain and read record parameters are moved into bytes 24-30 by the Chain and Read routine. Bytes 21-37 are entered into the IOCB at object time by the Record ID routine. Each IOCB contains:

Byte	*Bit*	*Contents*
0	0	1 = End of file has occurred
	1	1 = File not open
	2	1 = Identify look-ahead file
	3	1 = Non-input control file (not primary or secondary)
	4	1 = Translate file
	6	1 = End-of-file specified on file description specifications
	7	1 = Buffer full (does not need to be read from this cycle)
1	0	1 = BSCA 'last' file
	1	1 = Limits file
	2	1 = Combined file
	3	1 = Update file
	4-6	Record address type 000 = Data base 010 = Key 100 = Record ID 110 = Record number
	7	1 = Record address file
2-3		IOCB chain address
4-5		DTF address
6-7		Translate table address
8-9		File relation address ('from' or 'to' IOCB addresses for record address files or tables)
10		Overflow indicator mask
11-12		Record length
13-14		Address of output work area
15		Sequence number (in binary)
16		External indicator

If the file is an output file only, the following entries will not be present:

Byte	*Bit*	*Contents*
17-18		Input buffer address
19-20		Alphabetic sequence input record processing address
21-23		Address of last numeric input record processed and sequence information (byte 23 identical to byte 37)
24		Communication byte X'01' = Stacker select request (Models 10 and 12) X'02' = Data fields present in record X'04' = Control fields present in record X'08' = Matching fields present in records X'10' = Numeric sequence in record X'20' = Console file X'40' = Numeric sequence in this file X'80' = Recycle check bit (If all numeric sequence checking is optional, this bit is used to determine if a record does not fit any of the numeric sequences, indicating an error.)
25-26		Resulting indicator mask and displacement
27		Operation code for IOCS
28	0-4	If MFCU stacker, not used; if console, entire byte contains input buffer length (Model 10 or 12)
	5-7	000 = No select 010 = Print 4 tiers 100 = Stacker 4 101 = Stacker 1 110 = Stacker 2 111 = Stacker 3 (Model 10 or 12)
	0-7	X'00' = No select X'01' = Stacker 1 (MFCU and MFCM) X'02' = Stacker 2 (MFCU and MFCM) X'03' = Stacker 3 (MFCU and MFCM) X'04' = Stacker 4 (MFCU and MFCM) X'05' = Stacker 5 (MFCM only) (Model 15 only)
29-30		Address of Move Input Fields, code for this record type
31-32		Address of Control Fields Move code for this record type
33-34		Address of Matching Records moves code for this record type
35-36		Address of next numeric sequence checking code for this file
37		Numeric sequence information X'01' 1 = Numerous 0 = One X'02' 1 = Mandatory record found X'04' 1 = Mandatory 0 = Optional

OVERLAYS (MODELS 6, 10, and 12)

The RPG II Compiler uses a unique overlay system. This discussion concerns three areas of the overlay structure. First, the basic overlay concepts are discussed and defined. Then, the technique employed in RPG II is discussed including the Overlay Fetch routine and the overlay fetch table. Finally there is a section to help determine how to find an overlay and its contents in a source listing.

Overlay Concept

When the size of the program to be generated exceeds the size of main storage, an overlay program is required. An overlay program uses the same areas in main storage during different stages of a problem. When one segment of code is no longer needed in main storage, another segment can replace all or part of it. The RPG II Compiler uses an automatic overlay editor which determines during compile time the overlay structure of the object program.

Segments

The first step required to generate overlays is to divide the object program into segments. A segment is part of a program that is a logical unit of code and can be identified separately from other object code. There are three segment types in RPG II: the Root Segment, mainline segments, and subsegments.

Root Segment

The Root Segment is unique since it is the only segment that remains in main storage throughout execution of the program. It cannot be overlaid. The Root Segment has no fixed size. It contains the RPG II work area including indicators, DTFs, IOBs, IOCBs, buffers, data fields, and tables. In short, all data used in more than one cycle during execution must be stored in the Root Segment. The Overlay Fetch routine (see *Overlay Fetch Routine* in this section) is the only routine that is required in the Root Segment.

A segment may access the Root Segment at anytime with any instruction. The Root Segment, however, can access other segments only by using a branch instruction.

Mainline Segments

The mainline segments contain cycle control for the RPG II object program. Mainline segments may be called only by other mainlines with only simple branches (no branch and return) allowed. The mainlines in RPG II are:

1. Open
2. Detail Output
3. Input Records
4. Total Calculations
5. Total Output
6. LR and Overflow Control
7. Input Fields
8. Detail Calculations
9. Program Close

Subsegments

Subsegments are routines or subroutines that can be called by mainline segments or other subsegments. Control is always returned to the calling segments or subsegments. The RPG II subsegments are:

All library subroutines

Record ID Processing

Multifile Logic Processing

Control Fields Processing

LR Output

LR Calculations

OA-OV Overflow

Input Processing Control routine

Output Processing Control routine

Exception Output

Fetch Overflow

Chain Code Blocks

I/O Interface Blocks

Output Field Moves, not inline

Overlay Priority

Each program contains segments of code which are frequently used and others which are seldom required during execution. The overlay editor (phases $RPSB and $RPSE) calculates an effective overlay structure.

Each segment or subsegment gets a priority number assigned specifying the frequency with which it is called. For practical reasons, the numbers are assigned in reverse order, which means that the highest priority is represented by the lowest number. The highest priority is reserved for the Root Segment because it must be in storage at all times. The priorities are:

Priority (hex)	*Description*
00	Root Segment
00	Overlay Fetch routine
00	Transfer vector table (to call main overlays)
00	Overlay area
00	File Translate subroutine
01	Detail Output Mainline
01	Input Mainline
01	Total Calculations Mainline
01	Total Output Mainline
01	LR and Overflow Control Mainline
01	Input Processing Control routine (IPCR) Subsegment

Byte	Bit	Disk	Data Recorder	Printer	Ledger	CRT	Console	Keyboard (Manual Mode)
0	0	Get	Read	Print	Operation code	Operation code	Input*	Tractor 1
	1	Put/Add					Output*	Tractor 2
	2	Update	Punch					
	3							Turn off command key lights
	4							
	5							
	6						Accept characters only	
	7	Do not clear the output buffer						Manual mode
1				Skip before	Space after; X'80'= eject	Blank screen	Number of bytes to be printed	Element return
2				Space before		Blank lines before	Number of bytes to be read on WTOR*	Field light mask
3				Skip after		Blank lines after	Zone: Space before	Allowed command key mask Byte 1,Bit 0=Com. Key 16 ≀ Bit 7=Com. Key 9 Byte 2,Bit 0=Com. Key 8 ≀ Bit 7=Com. Key 1
							Digit: Space after	
4				Space after				
5				Start print position				Address of tab table
6				Number of bytes to be printed				
7				Position after print				

Byte	Bit	Keyboard** (Display or Store Mode)
0		Address of field
1		
2		
3	0	Tractor 1
	1	Tractor 2
	2	
	3	
	4	Numeric field
	5	Blind key mode
	6	Display mode
	7	
4		Field length
5		Field light mask
6		Allowed command key mask Byte 1,Bit 0=Com. Key 16 ≀ Bit 7=Com. Key 9 Byte 2,Bit 0=Com. Key 8 ≀ Bit 7=Com. Key 1
7		

*WTOR = Write to operator with reply specified.

**For Display or Store mode, the start of the keyboard parameter passed by RPG II will start at X'0C'.

Figure 4-5. Model 6 Data Management Parameters

Byte	Bit	MFCU	Disk	Printer	Console	Special	BSCA
0	0	Read	Get		Input*	Input	Input
	1	Print	Put/Add	Print	Output*	Output	Output
	2	Punch	Update				
	3	Move			"I"-type inquiry program		
	4						
	5						
	6				Accept characters only		
	7	Do not clear output buffer					
1	2	Print 4 lines		Skip before	Number of bytes to be printed		
	5-7	Stacker select					
2				Space before	Number of bytes to be read on WTOR*		
3				Skip	Zone: Space before Digit: Space after		
4				Space after			

*WTOR = Write to operator with reply specified

Figure 4-6. Models 10 and 12 Data Management Parameters

Priority (hex)	*Description*
01	Input fields
01	Output Processing Control routine (OPCR) Subsegment
01	Literals, constants, edit patterns, and parameters
01	Detail Calculations Mainline
01	Record Identification processing
01	Multi-file Logic processing
01	Control Fields processing
03	Alternate Collating Sequence subroutine
04	Set resulting Indicators subroutine
08	Array Index subroutine
15	Multiply subroutine
15	I/O Hook
15-1A*	Calculation subroutines (SR in cols 7 and 8)
16	Exception Output Subsegment
20	LOKUP subroutine
22	Pack subroutine
22	Unpack subroutine
25	Divide subroutine
28	TESTZ subroutine
30	Convert to Binary subroutine
30	Convert to Decimal Subroutine
32	RA File Process subroutine
32	Square Root subroutine
32	Chain code block
36	DEBUG subroutine
40	Fetch Overflow Subsegment
40	OA Overflow Subsegment
40	OB Overflow Subsegment
40	OC Overflow Subsegment
40	OD Overflow Subsegment
40	OE Overflow Subsegment
40	OF Overflow Subsegment
40	OG Overflow Subsegment
40	OV Overflow Subsegment
56	Open Mainline
56	Program Close Mainline
64	LR Output Subsegment
64	LR Calculations Subsegment
64	Load Object Tables
64	Dump Object Tables

*The first Calculation subroutine in the source code will receive a priority of 15, the second 16, and so forth until 1A is assigned. After that, all other calculation subroutines will receive 1A priorities.

Suboverlays

RPG II has a 2-level overlay structure. Mainline segments may become overlays. They are loaded in the main overlay area. The main overlay area has a minimum size of 256 bytes (one sector). Subsegments may become suboverlays which are stored in the suboverlay area. The minimum size of the suboverlay area is also 256 bytes (one sector). Suboverlays can be called only by mainlines. If a subsegment calls another subsegment, only one suboverlay containing both subsegments is generated. With this method, overlays that are too large for storage space can be reduced by breaking out some subsegments and generating suboverlays. The size of the overlay areas is increased to the size of the largest overlay if this proves necessary. Multiple suboverlays may be generated for one mainline. However, only one of these may be in main storage at any given time.

Overlay Technique

If the available storage size specified on the control statement is smaller than the object program size, overlays are required. The Overlay phases have two different functions. First, they format the object code generated by the Assign and Assemble phases for use by the *linkage editor.* Second, they create overlays by using an automatic *overlay editor* (phases $RPSB and $RPSF). Chart FA gives a general description of the overlay phases.

Overlay Editor

If it is determined that overlays are necessary, phase $RPSB acts as the overlay editor. Using the segment list built by the previous overlay phases, the overlay editor begins the following 3-step cycle for each segment and subsegment in the list:

1. Flags the segment or subsegment that has the lowest priority and has not previously been flagged as a main overlay or suboverlay.

2. Determines the length of the main overlay or suboverlay areas and adds the length to the total program.

3. Subtracts the length of the main overlay or suboverlay from the total program.

These three steps are repeated until one of two things happen:

1. The total program is reduced to a size that will fit in main storage.

2. All of the segments and subsegments are flagged as main overlays or suboverlays and the total program is still too big. In this case, an error message is written.

If the 3-step cycle is completed successfully, the overlay editor completes the task of flagging the remaining segments and subsegments as overlays and suboverlays.

Next the overlay editor (phase $RPSF) checks to see if, during the process of taking segments and subsegments out of the total program (overlaying), the total program was reduced so that it has some unused storage space. The unused storage space is then refilled with overlays previously taken out.

In addition to the overlay cycle described, the overlay editor does some optimizing of the overlays. Two or more smaller overlays are often combined to make one larger one if their combined size does not exceed the main overlay and/or suboverlay areas. In addition, small suboverlays can be combined with a small main overlay if they will be used by the small main overlay and the created overlay does not exceed the combined size of the overlay areas.

Overlay Fetch Routine

The Overlay Fetch routine, built by phase $RPSG, is 128 bytes long. The main function of this routine is to fetch overlays from access devices into the overlay areas in main storage. In addition, bits are set in the overlay fetch table telling where the overlays are.

The Overlay Fetch routine requires three parameters as input. They are:

1. Overlay number (1 byte)
2. Entry address of the overlay (2 bytes)
3. Return address from the overlay (2 bytes)

Phase $RPSG builds a transfer vector containing the input to the Overlay Fetch routine. The transfer vector contains these instructions:

```
ST   OVFRS1,ARR     Save the return address
B    OVFR           Call the Overlay Fetch routine
DC   1,X'NN'        One byte containing the
                    overlay number minus one
DC   2,A'ENTRY'     Two-byte entry address
```

The Overlay Fetch routine checks to see if the overlay passed is in main storage. If it is, the routine branches to the overlay. If it is not in main storage, the overlay fetch table entries are checked to see if they use the same main storage. If they do, the overlay is flagged as not being in main storage.

After checking all entries in the overlay fetch table, the entry of the overlay to be called in is flagged as in main storage and the overlay is loaded into main storage. The Overlay Fetch routine exits to the overlay. Chart FB gives a description of the Overlay Fetch routine.

Overlay Fetch Table

The overlay fetch table built by compiler phase $RPSG contains one 7-byte entry for each overlay or suboverlay in this format:

Byte	*Bit*	*Contents*
0-1		Cylinder/sector address of overlay relative to the Root Segment
2		Number of text sectors
3-4		Storage address where overlays are loaded
5		Displacement of relocation dictionary in last text sector
6	0	1 = Overlay in storage now
	1	1 = Overlay using overlay area (bit 2 can also be on)
	2	1 = Overlay using suboverlay area (bit 1 can also be on)
	3-6	Unused
	7	1 = Special OPEN/CLOSE mainline

How to Find an Overlay

The following steps may be used to determine what overlays are in main storage when a process check occurs and where to find them.

1. Locate the address of the Overlay Fetch routine on the core usage map of the source listing (Figure 4-7, part 1).
2. Locate the overlay fetch table in the dump. The overlay fetch table is 115 bytes past the start address of the Overlay Fetch routine. It can be obtained by this hex formula: Address of Overlay Fetch routine +X'73' = Overlay fetch table (Figure 4-7, part 2).
3. Mark off every 7-byte entry in the overlay fetch table until the last entry is reached. The last entry is X'FF' (Figure 4-7, part 2).

4. Number each entry, left to right, starting with number 1. Each entry refers to an overlay.

5. Look at the seventh byte in each entry. This is the flag byte. The first bit will be on for every overlay in storage at the time of the dump.

6. Compare the numbers you have given the overlays in storage at the time of the dump with the number of the overlays in the core usage map (Figure 4-7, part 1). This gives the names and addresses of the segments within the overlays which were in storage at the time of the dump.

For further understanding of the areas within the overlays, see the detail flowcharts in this section.

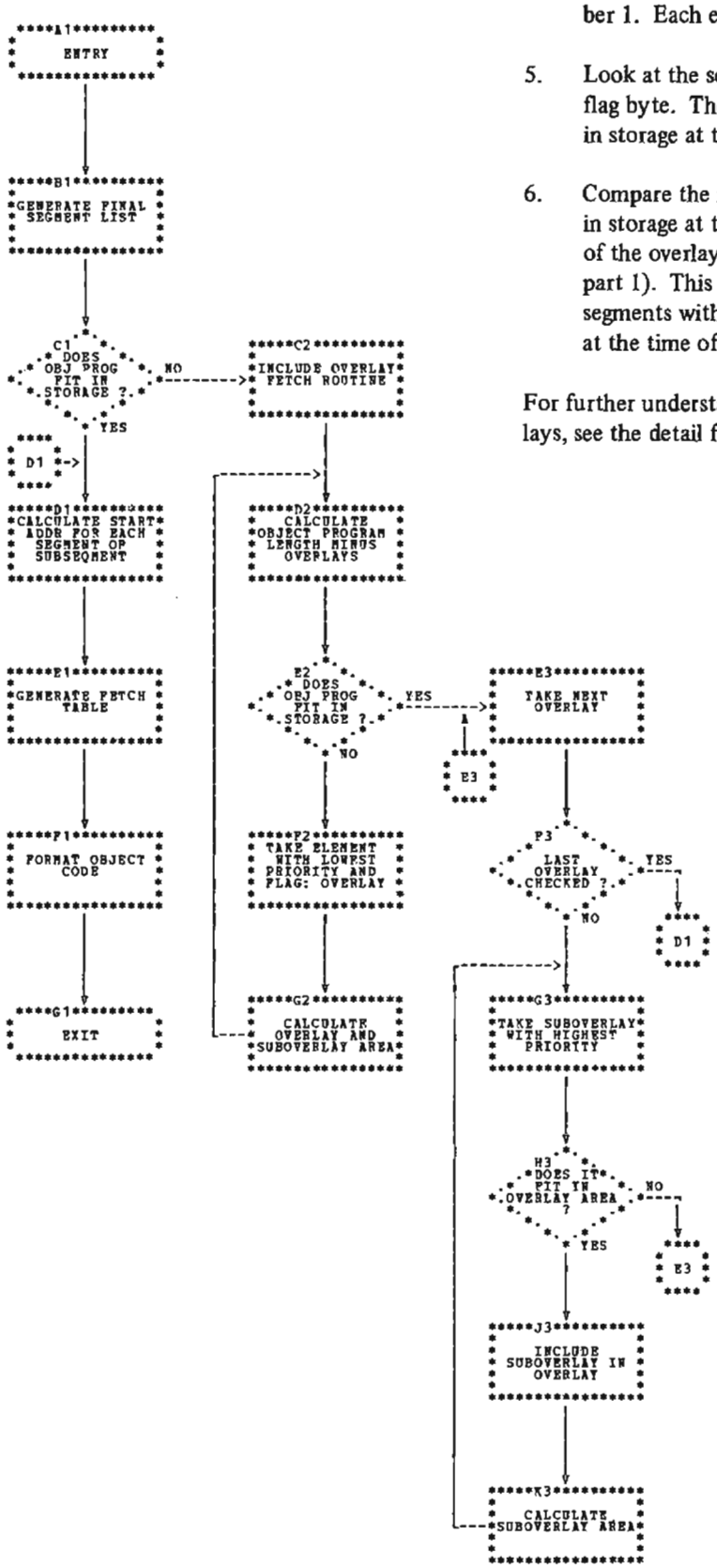


Chart FA. General Overlay Flow

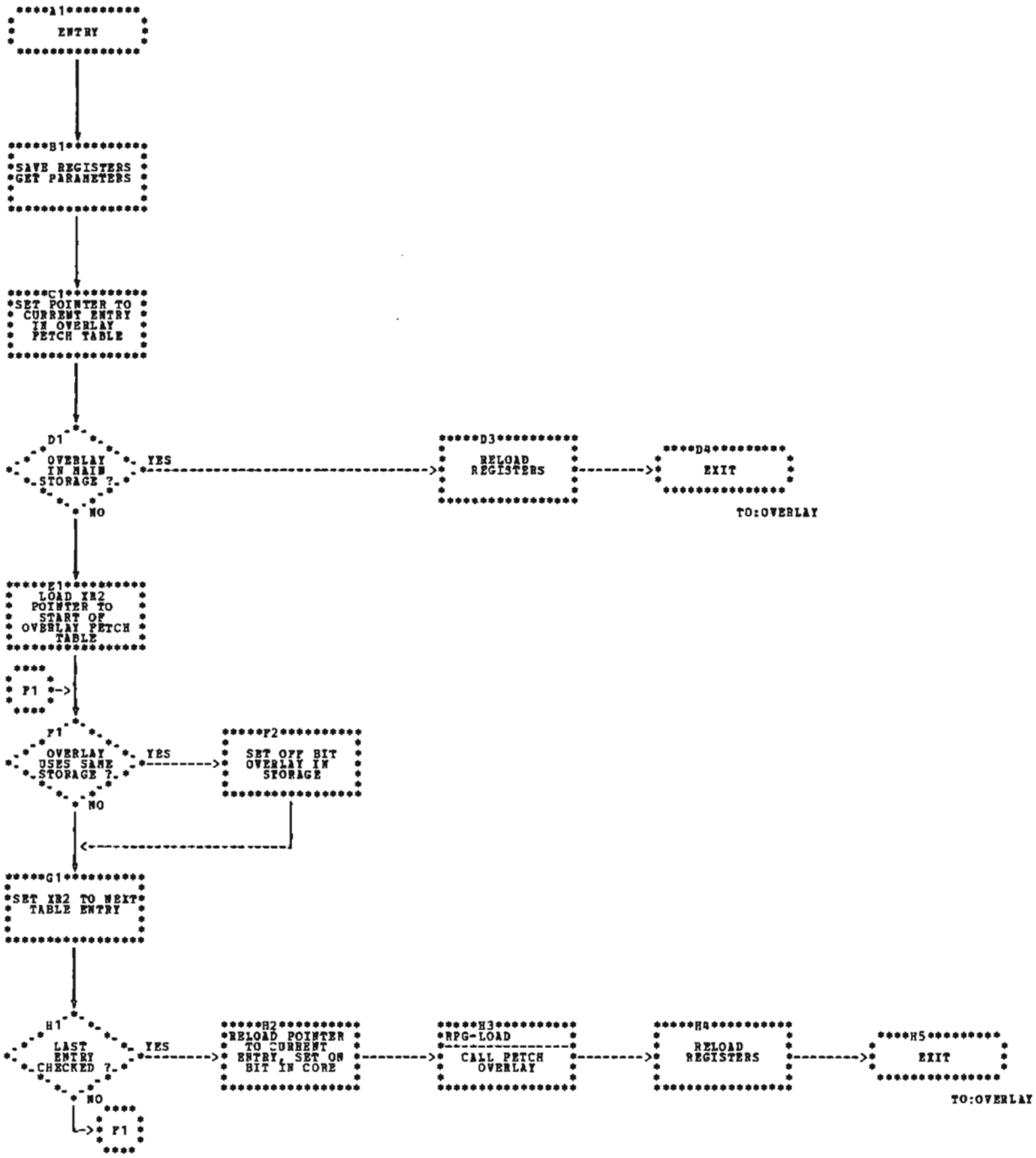


Chart FB. Overlay Fetch Routine

```
 INDICATORS USED
       LR 01 02

RG 314 UNREFERENCED FIELD NAMES
 STMT#  NAME  DEC  LGTH  DISP
  0005 NODATA       001   0100

  FIELD NAMES USED
 STMT#  NAME  DEC  LGTH  DISP
  0005 NODATA       001   0100
  0006 COUNT    0   006   0106
  0007 RECNBR   0   003   0109

  ERROR SEVERITY                                          TEXT
RG  314 W       FIELD, TABLE OR ARRAY NAME DEFINED BUT NEVER USED.
```

ADDRESS OF OVERLAY FETCH ROUTINE

```
                                        CORE USAGE OF RPGII CODE
   START  NAME IF  CODE     NAME     TITLE
   ADDR   OVERLAY  LENGTH
   0C00            04F6     RGROOT   ROOT
   10F6            017E     RGSUBS   OVERLAY FETCH ROUTINE
   1274            0600     RGSUBS   OVERLAY FETCH AREA
   12E6   #$$001   008E     RGMAIN   INPUT MAINLINE
   127C   #$$001   006A     RGSUBS   INPUT CTRL RTN
   1374   #$$001   003A     RGSUBS   RECORD ID
   13AE   #$$001   0026     RGSUBS   CONTROL FIELDS
   1274   #$$001   0008     RGSUBS   SUBSEG
          #$$001   0027     $$BDMC   SYSTEM SUBR
   1282   #$$002   000B     RGMAIN   TOTAL CALCS
   1274   #$$002   000E     RGSUBS   CONSTANTS
   1274   #$$003   0022     RGMAIN   INPUT FIELDS
   133D   #$$004   0048     RGMAIN   DETAIL CALCS
   132A   #$$004   0005     RGSUBS   CONSTANTS
   132F   #$$004   000E     RGSUBS   CONSTANTS
   1280   #$$004   00AA     RGSUBS   OUTPUT CTRL RTN
          #$$004   0043     $$PGRI   RESET RESULTING INDR
   1385   #$$004   0031     RGSUBS   EXCEPTION
   1274   #$$004   000C     RGSUBS   SUBSEG
          #$$004   005F     $$IOUT   DISK INDEXED OUTPUT
          #$$004   0094     $$SRBI   SYSTEM SUBR
          #$$004   0026     $$SRUA   SYSTEM SUBR
          #$$004   007B     $$SRBR   SYSTEM SUBR
          #$$004   001C     $$SRDF   SYSTEM SUBR
          #$$004   001C     $$SRTC   SYSTEM SUBR
          #$$004   002F     $$SRBP   SYSTEM SUBR
          #$$004   0019     $$SRCR   SYSTEM SUBR
          #$$004   0015     $$SRPD   SYSTEM SUBR
          #$$004   0081     $$SRMO   SYSTEM SUBR
          #$$004   0043     $$SRSB   SYSTEM SUBR
          #$$004   0015     $$SRRD   SYSTEM SUBR
   1274   #$$005   000B     RGMAIN   TOTAL OUTPUT
   1347   #$$006   0024     RGMAIN   LR & OVERFLOW PROCESSING
   1280   #$$006   00AA     RGSUBS   OUTPUT CTRL RTN
   132A   #$$006   001D     RGSUBS   OVERFLOW SUBSEGMENT
   1274   #$$006   000C     RGSUBS   SUBSEG
          #$$006   0027     $$BDMC   SYSTEM SUBR
   1438   #$$007   0024     RGMAIN   CLOSE
   1280   #$$007   00AA     RGSUBS   OUTPUT CTRL RTN
```

OVERLAY IN STORAGE

```
   1416   #$$007   000E     RGSUBS   CONSTANTS
   132A   #$$007   00EC     RGSUBS   CONSTANTS
   145C   #$$007   0076     RGSUBS   LR PROCESSING
   1274   #$$007   000C     RGSUBS   SUBSEG
          #$$007   0027     $$BDMC   SYSTEM SUBR
   1424   #$$007   0014     RGSUBS   LR CALCS
   1274   #$$008   0072     RGMAIN   OPEN

                  03144     SAMPL1   TOTAL CORE USAGE
RG  999 W       PROGRAM EXCEEDS CORE IN COLUMNS 12 THRU 14 OF HEADER CARD.
RRABC 12      0  3
```

Figure 4-7 (Part 1 of 2). Overlay Sample

Figure 4-7 (Part 2 of 2). Overlay Sample

OVERLAYS (MODEL 15)

An overlay program is required if the size of the program to be generated is greater than the amount of main storage available for execution of the program. The Overlay Linkage Editor calculates an effective overlay structure. See *IBM System/3 Overlay Linkage Editor and Checkpoint/Restart Program Logic Manual* (SY21-0530), for detailed information about the Overlay Linkage Editor.

Overlay Category

Each program is made up of segments of code. Some segments are frequently used and others are seldom required during execution of the program. Each segment or subsegment is assigned a category number specifying the frequency with which it is called. The lower category numbers are assigned to the most frequently used segments, the higher category numbers to the less frequently used segments. There is one exception to this, category 126, which contains RPG mainlines and other frequently called segments and receives special handling by the Overlay Linkage Editor. Category 00 is non overlayable and must remain in main storage at all times during execution of the program.

The categories are as follows:

Category (decimal)	Mnemonic ID	Description
00	GLOBAL	Buffers and IOBs
00	COMMON	Fields, pre-execution and execution time table/array
00	Object Program Name	ROCA, pre-execution and execution time table/array DTTs (from col. 75-80 of the RPG Header Card)
00	$#RT02	IOCBs, DTFs, DTTs and compile time table/array
00	$#MISC	Edit codes, SETLL code, etc
00	$#IPCR	Input Processing Control routine
00	$#OPCR	Output Processing Control routine
00	$#CON0-F	Literals, constants, edit patterns and parameters
00	$$PGAB	File Translate routine
00	$$PGBB	BSCA logic for transmit/receive
00	$$PGBC	BSCA logic for conversational
00	$$PGBG	BSCA logic for receive
00	$$PGBP	BSCA logic for transmit
00	$$PGRB	BSCA logic for transmit/receive (2770/2780)
00	$$PGRG	BSCA logic for recieve (2770/2780)
00	$$PGRP	BSCA logic for transmit (2770/2780)
00	$$PGDC	DEBUG Routine
10	$$PGDI	Alternate Collating Sequence routine
11	$$PGRI	Set Resulting Indicator routine
15	$$PGAA	Array Index routine
28	$#IH01-19	Input hook (number = file number)
28	$#OH01-14	Output hook (number = file number)
28-33*	$#SR00-FF	Calculation subroutines
28	$$PGMC	Multiply subroutine
29	$#EXPT	Exception Output Segment
35	$#MF01-FF	Move Output Field for OR lines
39	$$PGLC	LOKUP routine
41	$$PGCO	Pack routine
41	$$PGCI	Unpack routine
44	$$PGIC	Divide routine
45	$$PGDP	Display routine
47	$$PGTC	TESTZ routine
48	$$PGMA	MOVEA routine
55	$$PGBI	Convert to Binary routine
55	$$PGBO	Convert to Decimal routine
57	$#RAFL	RA File Processing routine
57	$#CHN0-F	Chain Code Subsegment
57	$$PGAC	Square Root routine
71	$#OAOF	OA Overflow routine
71	$#OBOF	OB Overflow routine
71	$#OCOF	OC Overflow routine
71	$#ODOF	OD Overflow routine
71	$#OEOF	OE Overflow routine
71	$#OFOF	OF Overflow routine
71	$#OGOF	OG Overflow routine
71	$#OVOF	OV Overflow routine
71	$#FOVF	Fetch Overflow routine
93	$#OPEN	Open Mainline
93	$#CLOS	Close Mainline
107	$$PGFI	Load Object Tables
107	$$PGFO	Dump Object Tables
107	$#LROT	LR Output
107	$#LRC	LR Calculations
126	$#DOUT	Detail Output Mainline
126	$#INPT	Input Mainline
126	$#TCAL	Total Calculations Mainline
126	$#TOUT	Total Output Mainline
126	$#LROF	LR and Overflow Processing Mainline
126	$#IFLD	Input Fields Mainline
126	$#DCAL	Detail Calculations Mainline
126	$#RCID	Record ID routine
126	$#MFLG	Multi-File Logic routine
126	$#CFLD	Control Fields Processing routine

*The first five calculation subroutines in the source code are assigned categories 28 through 32 respectively. All other subroutines are assigned category 33.

DUMP ANALYSIS

This section presents an aid for examining the areas of a storage dump of an RPG II Compiler program.

Figures 4-8 through 4-11 show the source listings of the specification statements for two Model 6 sample programs and the sample storage dumps of these two programs.

Figures 4-12 through 4-15 show the source listing of the specification statements for two Model 10 sample programs and the sample storage dumps of these two programs. Figures 4-12 through 4-15 can also be used as an aid for examining storage dump areas of an RPG II Compiler program for the Model 12.

Figures 4-16 through 4-19 show the source listing of the specification statements for two Model 15 sample programs and the sample storage dumps of these two programs.

```
                0101 H     008                                                                  SAMPL1


                0102 F*********************************************                              SAMPL1
                0103 F*                                           *                              SAMPL1
                0104 F* THIS PROGRAM -                            *                              SAMPL1
                0105 F*                                           *                              SAMPL1
                0106 F*   1. LOADS 100 RECORDS TO AN INDEXED FILE.*                              SAMPL1
                0107 F*                                           *                              SAMPL1
                0108 F*   2. READS ONLY A BLANK RECORD AND A /* RECORD*                          SAMPL1
                0108 F*      AS INPUT DATA.                       *                              SAMPL1
                0109 F*                                           *                              SAMPL1
                0110 F*   3. CREATES THE OUTPUT DATA USING A      *                              SAMPL1
                0111 F*      LOOP IN THE CALCULATION SPECIFICATIONS.*                            SAMPL1
                0112 F*                                           *                              SAMPL1
                0113 F*   4. USES KEYS FROM 000005 THROUGH 000500 *                              SAMPL1
                0114 F*      IN INCREMENTS OF 5.                  *                              SAMPL1
                0115 F*                                           *                              SAMPL1
                0116 F*   5. SHOULD BE FOLLOWED BY SAMPLE PROGRAM 2*                             SAMPL1
                0117 F*      TO VERIFY THAT THE FILE WAS PROPERLY *                              SAMPL1
                0118 F*      LOADED.                              *                              SAMPL1
                0119 F*                                           *                              SAMPL1
                0120 F*********************************************                              SAMPL1
0001            0121 FINPUT   IPE F  96  96            CONSOLE                                  SAMPL1
0002            0122 FDISKOUT O   F 256 128 06AI     1 DISK                           01        SAMPL1
0003            0123 FOUTPUT  O   F   1 132            TRACTR1


0004            0201 IINPUT   NS  01   1 C                                                     SAMPL1
0005            0202 I                                    1   1 NODATA                        SAMPL1


0006            0301 C   01               Z-ADD0          COUNT   60                           SAMPL1
0007            0302 C   01               Z-ADD0          RECNBR  30                           SAMPL1
0008            0303 C           REPEAT   TAG                                                  SAMPL1
0009            0304 C   01      COUNT    ADD  5          COUNT                                SAMPL1
0010            0305 C   01      RECNBR   ADD  1          RECNBR                               SAMPL1
0011            0306 C   01      COUNT    COMP 505                        02                  SAMPL1
0012            0307 C   01N02            EXCPT                                                SAMPL1
0013            0308 C   01N02            GOTO REPEAT                                          SAMPL1
0014            0309 CLR         RECNBR   SUB  1          RECNBR                               SAMPL1


0015            0401 OOUTPUT  T  201   LR                                                     SAMPL1
0016            0402 O                                        20 'SAMPLE PROGRAM 1 HAS'        SAMPL1
0017            0403 O                                        27 'LOADED'                      SAMPL1
0018            0404 O                          RECNBR2       31                               SAMPL1
0019            0405 O                                        39 'RECORDS'                     SAMPL1
0020            0406 O                                        61 'INTO AN INDEXED FILE.'       SAMPL1
0021            0408 O         T  2      LR                                                    SAMPL1
0022            0409 O                                        21 'KEYS ARE IN ASCENDING'       SAMPL1


0023            0410 O                                        42 'SEQUENCE STARTING AT'        SAMPL1
0024            0411 O                                        64 '000005 AND INCREASING'       SAMPL1
0025            0412 O                                        84 'IN INCREMENTS OF 5.'         SAMPL1
0026            0413 O         T      01 LR                                                    SAMPL1
0027            0414 O                                        21 'SAMPLE PROGRAM 2 WILL'       SAMPL1
0028            0415 O                                        44 'PRINT FROM THE INDEXED'      SAMPL1
0029            0416 O                                        65 'FILE TO SHOW THAT IT'        SAMPL1
0030            0417 O                                        86 'WAS PROPERLY LOADED.'        SAMPL1
0031            0501 ODISKOUT E          01N02                                                SAMPL1
0032            0502 O                          COUNT          6                             SAMPL1
0033            0503 O                                        94 'RECORD NUMBER'               SAMPL1
0034            0504 O                          RECNBR       128                             SAMPL1
```

Figure 4-8 (Part 1 of 2). Source Listing of Sample Program SAMPL1 (Model 6)

```
 INDICATORS USED
       LR 01 02

RG 314 UNREFERENCED FIELD NAMES
 STMT#  NAME   DEC  LGTH  DISP
  0005 NODATA       001   0100

  FIELD NAMES USED
 STMT#  NAME   DEC  LGTH  DISP
  0005 NODATA       001   0100
  0006 COUNT    0   006   0106
  0007 RECNBR   0   003   0109

  ERROR SEVERITY                                          TEXT
RG  314 W       FIELD, TABLE OR ARRAY NAME DEFINED BUT NEVER USED.


                                        CORE USAGE OF RPGII CODE
   START  NAME IF  CODE     NAME     TITLE
   ADDR   OVERLAY  LENGTH
   0C00            050C     RGROOT   ROOT
   117B            008E     RGMAIN   INPUT MAINLINE
   1114            0067     RGSUBS   INPUT CTRL RTN
   1209            003A     RGSUBS   RECORD ID
   1243            0026     RGSUBS   CONTROL FIELDS
   110C            0008     RGSUBS   SUBSEG
   1269            000B     RGMAIN   TOTAL CALCS
   1274            0022     RGMAIN   INPUT FIELDS
   12A7            0048     RGMAIN   DETAIL CALCS
   12A2            0005     RGSUBS   CONSTANTS
   1320            0043     $$PGRI   RESET RESULTING INDR
   12EF            0031     RGSUBS   EXCEPTION
   1296            000C     RGSUBS   SUBSEG
   1363            005F     $$IOUT   DISK INDEXED OUTPUT
   13C2            0093     $$SRBI   SYSTEM SUBR
   1455            0026     $$SRUA   SYSTEM SUBR
   147B            007B     $$SRBR   SYSTEM SUBR
   14F6            001C     $$SRDF   SYSTEM SUBR
   1512            001C     $$SRTC   SYSTEM SUBR
   152E            0015     $$SRPD   SYSTEM SUBR
   1543            0019     $$SRCR   SYSTEM SUBR
   155C            002F     $$SRBP   SYSTEM SUBR
   158B            0081     $$SRMO   SYSTEM SUBR
   160C            0043     $$SRSB   SYSTEM SUBR
   164F            0015     $$SRRD   SYSTEM SUBR
   1664            000B     RGMAIN   TOTAL OUTPUT
   168C            0024     RGMAIN   LR & OVERFLOW PROCESSING
   166F            001D     RGSUBS   OVERFLOW SUBSEGMENT
   16B0            0027     $$BIMC   SYSTEM SUBR
   189B            0024     RGMAIN   CLOSE
   16E3            00AA     RGSUBS   OUTPUT CTRL RTN
   1879            000E     RGSUBS   CONSTANTS
   178D            00EC     RGSUBS   CONSTANTS
   18BF            0076     RGSUBS   LR PROCESSING
   16D7            000C     RGSUBS   SUBSEG
   1887            0014     RGSUBS   LR CALCS
   1935            0072     RGMAIN   OPEN

                  03495     SAMPL1   TOTAL CORE USAGE
```

DISK DATA MANAGEMENT ROUTINES

NON-DISK DATA MANAGEMENT LINKAGE ROUTINE

Figure 4-8 (Part 2 of 2). Source Listing of Sample Program SAMPL1 (Model 6)

Figure 4-9 (Part 1 of 4). Storage Dump of Sample Program SAMPL1 (Model 6)

```
11C0 1243C087 1269C087 168C3C32 11EDF287 0D3C3011 E0880118 F29003BC 001CBB01  *KC GKZ GG<@2J_2G<@OJ_8AQ2&C@ *#A*
11E0 00E20100 C2020CB4 C0870004 8F007408 3CC08711 0CB88000 F21004C0 871209BA  * SA BB<4 G D! 4H@ GJ<8  2&D GKI:*
                                RECORD ID
1200 01007510 3C120811 0CB50112 B5101434 081216C2 0100009C 0A220AF2 8708C087  *A 5&@KHJ<5AK5&M4HKOBA  *¢S¢2GH G*
1220 11CAC0B7 11D1C201 0C00C087 11FF7D40 00F2010B C0B7120F A202C980 60128DC0  *J¢ GJJBA<  GJ≠'  2A.  GK!SBI -K(  *
                     CONTROL FIELDS LOGIC AND MOVES
1240 871222B8 20C3F210 1535020C 9BBB0100 2C011257 1A7A0000 7B80D9F2 9004C0B7  *GKS8-C2&N5B<$#A  X&KP!:  8 R2&D G*
                                TOTAL CALCS                INPUT FIELDS MOVE
1260 11C27AB0 D9C08711 C6B20C3 C0101BA2 C0871664 75029BB9 0218F210 0C2C0112  *JB: R GJFB-C &QS GDU5B$9BQ2&<%AK*
                                                           LINK TO DISK OUTPUT
1280 881EB502 12C08700 00C08712 A72C000D 0000C087 1289C202 1363C201 0D303510  *H!5BK G   GKX% (   GKIBBLTBA(05&*
                     CONSTANTS    DETAIL CALCS
12A0 16E4F0F5 F5F0F5B8 02C9F290 0C045000 0612A204 200D0912 A27802C9 F2902C06  *&(FKTF-(IG9DU&(FGK&KW GL-ADI 9DI*
12C0 500D0612 A3062000 09187944 A5100006 47D21012 A6C08713 208104C9 007904C9  *&(FKTF-(IG9DU&(FGK&KW GL-ADI 9DI*
                                EXCEPTION OUTPUT
12E0 F2900BC0 8712EFC0 8712B9C0 B7117B84 08131F5C 8F8F90C2 020C003C 400D7F78  *2&H GK? GK9 GJ#4HL-*!!&BB< @ (≠8*
1300 02C97904 C9C09013 1C8C0505 0D068C0C 5D10B68C 027F0D09 C0871296 C0B7117B  *BI9DI &L*(EE(F<()QF<B≠(I GKD GJ#*
     $$PGRI
1320 740BFF74 02FD7502 FF7404FB F21000B9 FF00F210 227504FB 2C00133E 00F20004  *4H≠4B'5B≠4D#2& 9≠ 2&5SD#% L= 2 D*
1340 3C7B1349 2C01134B 027A0000 3C7A1349 E20203C0 B7132CE2 02017402 FF7502FB  *@#LI%AL.B:  @:LI5BC GL%SBA4B≠5B'*
     $$IOUT
1360 7510FFB4 0809B401 0BB84003 F29032B8 0127BB01 27F21029 3CFF138D 2E00138D  *5&≠4HI4A.8 C2&2BAX$AX2&Z@≠L() L(*
1380 382C0113 8F32B501 0DB6013C 1D000000 00F2820D BC600EF2 811CBC62 0EC08715  *8%AL!25A(6A@)    2B(@-+2A*@S+ GN*
13A0 26C08714 55C08714 F6BD700E F2B107C0 B713C2BD 700EC0B1 1526C087 147BC087  *W GMN GM6'0+2AG GLB'O+ ANW GM# G*
     $$SRBI
13C0 15234408 1450AD01 2D3AF202 7DB50136 C0871543 3CFF1403 2E001403 38750109  *NT4HM&_A_:2B'5A6 GNC@≠MC) MC85AI*
13E0 AC012238 AE00222E B6012236 0114524C 002E228E 002E1454 2C011406 0D2E0114  *%A8B) S)6AS6AMK% )S+ )MM%AMF()AM*
1400 063C4C00 00000088 4003F290 03B40132 AC002526 AF012418 6C020325 BCFF22AE  *F@(    8 C2&C4A2% VW?AU$%BCV@≠S)*
1420 00222EAE 002238F2 A0088E00 221454F2 201BB501 366C0106 2DC08715 2EBC002E  * S)) S82-H+ SMM2-$5A6%AF_ GN)@ )*
     $$SRUA
1440 E20127C0 B7155CF2 8703BC70 0EC0B700 00FFFF00 03340814 79AC0326 2BAE012B  *SAX GN*2GC@O+ G  ≠≠ C4HM9%CW,)A,*
     $$SRBR
1460 17E20123 B9FF2AF2 100C8F00 2A147AC0 87155CBC 002AC087 00000134 0814F185  *PSAT9≠=2&(! =M: GN*@ = G  A4HM15*
1480 0113C087 1543B0B0 0FB81027 C015160C AE011917 AC012000 AC012417 AC012215  *AL GNC' !8&X NO(>AKP%A-(%AUP%ASN*
14A0 8F012214 F5AD0119 22F2823A AC012619 AF012622 AF012426 AC011922 C0871588  *!ASM5_ARS2B:%AWR?AWS?AUW%ARS GN.*
14C0 C087160C C0871543 8C011914 F3AE0119 26F22016 BDB00FF2 B104C087 160CAC01  * GG( GNC<ARM3)AKW2-D' !2AD GG<%A*
     $$SRDF
14E0 2426C087 149CC087 158BAC01 0D20C087 0000FFFF 00013408 1511AB01 2B1DF282  *UW GM* GN.%A(- G  ≠≠ A4HNJ_AZ)2B*
     $$SRTC
1500 0DB9FF2B F21407BC 700EAC03 2B26C087 00009C01 0D09AE01 0D198E01 0D152DAE  *(9≠,2MG@O+%C,W G  *A(I)A(R+A(N_)*
     $$SRPD
1520 011917BC 400EB501 0BB51009 00013408 15427B07 037A0203 7C0004C0 B7000BC0  *ARP@ +5A.5&I A4HNB$GC:BC@ D G H *
     $$SRCR
1540 87000034 0B1554C0 87000C7C 000C7B01 02C09000 00BC410E C0871526 34081586  *G  4HNM G (@ (BAB &  @A+ GNW4HNF*
```

Figure 4-9 (Part 2 of 4). Storage Dump of Sample Program SAMPL1 (Model 6)

```
1560 9C002207 4E000615 88786006 F290054E 0106158A BD0022F2 8109BF00 221587C0   ** 8G+ FNHB-F2&E+AFNC' S2AI| SNG *
                            $$SRMD
1580 B71564C0 87000001 04002034 08160534 0215FA2C 01160924 3C6C15CC B9800FF2   *GNU G  AD -4HGE4BN:ZAQIU@ZN(9 12*
15A0 10043C9C 15CC7501 09AF0119 24B60119 AE011924 B5020D36 02160B0F 00160816   *&D@*N(5AI?ARU6AR)ARU5B(6BD.| DHD*
15C0 06F28210 36011607 36021607 6CFF0000 C08715BB 3D001609 F2811C3C 00160836   *F2B&6A0G6BDG%≠   GN$' OI2A*@ DH6*
15E0 01160936 0216090C 0315F615 CF0E0015 F416096C 000000C2 020000AE 010D24B5   *ADI6BDI(CN6N|+ N4DI%   BB  )A(U5*
                                     $$SRSB
1600 0113C087 00000100 0000FFFF 3408164E BD800FF2 011E7502 136C0106 06B50215   *AL G  A    ≠≠4HD+' |2A;5RL%AFF5BN*
1620 C087155C B81027BB 1027F210 1EC08716 4FF2870E B81027C0 101616C0 87152EBA   * GN*8&X≠&X2&; GD|2G+B&X &DD GN):*
                                              $$SRRD
1640 1027B401 11750113 B40113C0 87000034 0816637B 07037A01 037C0004 C0870008   *&X4AJ5AL4AL G  4HOT≠GC:AC@ D G H*
              TOTAL OUTPUT                    OVERFLOW SUBSEGMENT
1660 C0870000 7820C3C0 1018AAC0 87168C34 0816867A 20D90C04 0DC0168B C0871607   * G  8-C &Q= GD(4HDF:-R(D( D. GDP*
                                     LR AND OVERFLOW PROCESSING
1680 7B20D9C0 87000000 06000000 7820C3C0 1018AE78 01D7C010 166F7BFF D77BFFD8   *≠-R G   F   8-C &Q)8AP &07≠≠P≠≠Q*
                                              $$BDMC
16A0 7A02C278 04C2F210 037B02C2 C0871274 F2870F5B 5BC2C4D4 C340F0F3 61F1F861   *:BBDB2&C≠BB GK42G|$$BDMC 03/18/*
                                                                 LINK TO PRINTER OUTPUT
16C0 F7F03408 16D62C01 16D11BC0 87000480 223802C0 871142C2 0216B0C2 010D4135   *704HDD%ADJ$ G D SBB GJBBBDDBA(A5*
              OPCR
16E0 1016E416 E5340817 8C340217 01750205 9C010D0E 2C001739 242C0017 4524C0FF   *&DUDV4HP(4BPA5RE*A(+% P9U% PEU ≠*
1700 0000B801 0FF21006 0C8F0CBF 0C90B8F0 00782001 F290046C 001C107D 000AF281   *  8A|2&F(|(|(&8D 8-A2&D% *&' C2A*
1720 4838200C D9F21041 1C011762 0A1C0117 660AB9FF 12F21006 BD0012F2 0415B9FF   *H8-(R2&A*AFS¢*AFW¢9≠K2&F' K2DN9≠*
1740 10F21006 BD0010F2 0409BD48 0EF20119 F2B70EBD 480EF281 083C3B17 613C3B17   *&2&F' &2DI'H+2AR2G+'H+2AH@$F/@$P*
1760 6500000C D600000C D7BD4D0E F20209B9 010EBD44 0EF2110A C2020CB4 C0870004   *V  (D  (P'(+2BI9A+'D+2J¢BB(4 G D*
                                          CONSTANTS
1780 8F407502 0EC2010C 00C08700 00400100 00020140 00000002 01400000 010001E2   *| 5B+BA(  G   A  BA    BA   A AS*
17A0 C1D4D7D3 C540D7D9 D6C7D9C1 D440F140 C8C1E2D3 D6C1C4C5 C4D9C5C3 D6D9C4E2   *AMPLE PROGRAM 1 HASLOADEDRECORDS*
17C0 C9D5E3D6 40C1D540 C9D5C4C5 E7C5C440 C6C9D3C5 4BD2C5E8 E240C1D9 C540C9D5   *INTO AN INDEXED FILE.KEYS ARE IN*
17E0 40C1E2C3 C5D5C4C9 D5C7E2C5 D8E4C5D5 C3C540E2 E3C1D9E3 C9D5C740 C1E3F0F0   * ASCENDINGSEQUENCE STARTING AT00*
1800 F0F0F0F5 40C1D5C4 40C9D5C3 D9C5C1E2 C9D5C7C9 D540C9D5 C3D9C5D4 C5D5E3E2   *0005 AND INCREASINGIN INCREMENTS*
1820 40D6C640 F54BE2C1 D4D7D3C5 40D7D9D6 C7D9C1D4 40F240E6 C9D3D3D7 D9C9D5E3   * OF 5.SAMPLE PROGRAM 2 WILLPRINT*
1840 40C6D9D6 D440E3C8 C540C9D5 C4C5E7C5 C4C6C9D3 C540E3D6 40E2C8D6 E640E3C8   * FROM THE INDEXEDFILE TO SHOW TH*
                                                                      CONSTANTS
1860 C1E340C9 E3E6C1E2 40D7D9D6 D7C5D9D3 E840D3D6 C1C4C5C4 4BF1D9C5 C3D6D9C4   *AT ITWAS PROPERLY LOADED.|RECORD*
                   LR CALCS                                                CLOSE
1880 40D5E4D4 C2C5D934 0B189A78 20C3F290 0607200D 091B79C0 87117B7A 20C33C80   * NUMBER4H@+8-C2&FG-(IQ? GJ$:-C@ *
     MAINLINE
18A0 18A7C087 1887C087 1664C087 18BF3501 0C997502 05C08700 0483C087 00048434   *QX GQG GDU GD≠5A(R5BE G DC G DD4*
     LR PROCESSING
18C0 0819345C 8F8F90C2 020C000C 050DC117 928C1313 17B28C05 1A17B88C 021E0D09   *HR4*||&BB( (E(AFK(LLF2(E+FB(B$(I*
18E0 BAF01E9B 021C908C 062617BF 8C143C17 D4C08716 D70C050D C117988C 141417E9   *:0;$B*&(FWP≠(M@PM GDP(E(APQ(MMPZ*
```

Figure 4-9 (Part 3 of 4). Storage Dump of Sample Program SAMPL1 (Model 6)

```
1900 8C132917 FD8C143F 1812BC12 53182SC0 8716D70C 050DC117 9EBC1414 183A8C15 *<LZF'(M≠GK(KLQV GQF(EAF}(MMG:(N*
                                                  OPEN MAINLINE
1920 2B18508C 13401864 8C135518 78C08716 D7C08711 7BC08700 04000201 0C005F1A *,QA(L QU(LNG8 GQP GJ≠ G D EA( ¬†*
1940 DCDC6C00 C5377A01 C27A40C3 75019975 0205C087 00048B75 0205C087 00048275 ***2 E7:AE: C5AR5BE G D.5BE G DE5*
1960 0205B801 03F2900D 7BC00079 1000F290 043C8719 854D0103 0C99F281 07750103 *ΕΕBAC2A(≠  9& 2AD@GRE(AC(R2AG5AC*
1980 C087195F F2870AC2 020CB4C0 87000A6F 11C2010C 003CF00D 090C070B * GR¬2GCEE(4 G DIJBA( @ ( @0(I(G(*
                  END OF SAMPL1
19A0 080D09C0 87117800 00000000 00000000 00000000 00000000 00000000 00000000 *H(I GJ≠                *
19C0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *                      *
1A00 21AC0326 2BAE012B 17E20123 B9FF2AF2 100C8F00 2A1A22C0 871A238C 002AC087 */%CW,>A,FSAT9≠=2&(I =†S GTT@ = G*
1A20 00000134 081A4B9C 0022074E 00061A4F 7860006F2 90054E01 061A51BD 0022F281 *  A4H†(* SG+ F†18-F2&E†AF†J' S2A*
1A40 098F0022 1A4EC087 1A2BC087 00000104 00203408 1A667B07 037A0103 7C0004C0 *II S†+ GT, G  AD -4H†W$GC:AC@ D *
1A60 87000BC0 87000034 081A7B7B 07037A02 037C0004 C0870008 C0870000 340814BD *G H G  4H†#$GC:BC@ D G H G  4H†(*
1A80 C0870000 7C000C7B 0102C090 0000BC41 0EC0871B 46340818 0F34021B 042C011B * G (@ (8AB &  @A+ G9F4HS14RSDZAS*
1AA0 13243C6C 1A1B6B80 0FF21004 3C9C1A16 7501094F 01192AB6 0119AE01 1924B502 *LU@2†09 I2&D@*†05AI?ARU6AR)ARU5B*
1AC0 0D360218 150F001B 121B10F2 82103601 1B113602 1B116CFF 0000C087 1AC53D00 *(6R$NI $K$&2R&6A$J6B$JZ≠  GTE' *
1AE0 1B13F281 1C3C001B 12360118 13360218 13C031B 001A0190E 001AFE1B 136C0000 *$L2A*@ $K&A$L6B$L(C$ †R+ †=$LZ  *
1B00 00C20200 00AE010D 24B50113 C0870000 0100E2E2 FFFF3408 1B31A001 291DF282 *  BB  )A(U5AL G  A SS≠F4H$1_AZ)2B*
1B20 01B9FF2B F21407BC 700EAC03 2B26C087 00009C01 0D09AE01 0D19BE01 0D1B4DAE *(9≠,2MG@O+%C,W G  *A(I)A(R+A(S()*
1B40 011917BC 400EB501 0BB51009 00012104A 04070409 09070409 04040204 02040404 *ARP@ +5A.5&I A/CDGDIIGDIIDDDEEDDD*
1B60 09020402 04260204 07070404 04FE2409 0C040704 041B0404 04080707 04040704 *IEDDEDWBDGGDDD=UI(DGDD$DDDMGGIDGD*
1B80 04020404 04070409 0F041310 0AFE0704 05050C11 02040704 040D0913 04260604 *DEDDDDGDI!DL&C=GDEE(JBDGID(ILDWFD*
1BA0 47050505 050B0302 02208080 80FE8080 80808080 80808080 80808080 80808016 *GEEEE.CBB-   =              G*
1BC0 04070404 04070A15 1B040405 11FE0804 10140D0A 04141504 0C090B0B 040C1515 *DGDDDGCN$DDDEJ=HDAM(CDMOD(I..D(NN*
1BE0 15040404 050A1504 02070408 04FE0704 04040204 021D25FF 00000000 00000000 *NDDDECNDBGDHD=GDDDKDB)V≠      *
1C00 2EC0871C 70740278 E2020774 0232E202 01740270 7C0733C2 021CC39C 07078FC0 *) G*04B$SBG4B2SBA4B'@G3BB*C*GGI *
1C20 87000469 D2027AC0 871CF2F2 8703F287 29C0871B FE34081C 535E0034 377D6034 *G DZKB: G*22GC2GZ G$=4H*L} 47'-4*
1C40 F2811758 003537E2 02017D08 35F2840A C0870000 7C042EF2 87167C02 2EF28710 *2AP} 57SBA'MS2Dc G  @D)2GD0B)2G&*
1C60 7C012EF2 870A7BF0 2E740232 5C003335 C2010D04 C2021705 8C06061C 87C08714 *@A)2G<$0)4B2* 35BA(DBBPE(FF*G GM*
1C80 E140040E 09090900 60000001 03C103B3 40C4C9D9 40E2E8E2 E3C5D440 C6F1D9F1 */ D+III -  ACALL DIR SYSTEM F1R1*
1CA0 C6F2D9F2 F0F0F0F0 F3F9F8F0 F9F9F9F0 F0F0F4F0 F5000040 40404040 40404040 *F2R2000039809999990000405      *
1CC0 40404040 40404040 40404040 F2000000 00001CEC D65E5BD9 C2C4E3E2 D65E5BD9 *          *2     *@D$^RBDTSD$^R*
```

Figure 4-9 (Part 4 of 4). Storage Dump of Sample Program SAMPL1 (Model 6)

```
                0101 H     008                                                                  SAMPL2


                0102 F*************************************************                  SAMPL2
                0103 F*                                                *                  SAMPL2
                0104 F* THIS PROGRAM -                                 *                  SAMPL2
                0105 F*                                                *                  SAMPL2
                0106 F*   1. MUST BE PRECEDED BY SAMPLE PROGRAM 1      *                  SAMPL2
                0107 F*      WHICH LOADS AN INDEXED FILE.              *                  SAMPL2
                0108 F*                                                *                  SAMPL2
                0109 F*   2. READS AN INDEXED FILE SEQUENTIALLY.       *                  SAMPL2
                0110 F*                                                *                  SAMPL2
                0111 F*   3. USES A BLOCK LENGTH FOR DISK WHICH        *                  SAMPL2
                0112 F*      IS DIFFERENT FROM THAT USED FOR           *                  SAMPL2
                0113 F*      LOADING THE FILE IN SAMPLE PROGRAM 1.     *                  SAMPL2
                0114 F*                                                *                  SAMPL2
                0115 F*   4. COUNTS THE NUMBER OF RECORDS READ SO      *                  SAMPL2
                0116 F*      THAT THE USER CAN QUICKLY VERIFY THAT     *                  SAMPL2
                0117 F*      100 RECORDS WERE LOADED.                  *                  SAMPL2
                0118 F*                                                *                  SAMPL2
                0119 F*************************************************                  SAMPL2
0001            0120 FDISKOUT IPE F 512 128 06AI     1 DISK                        01      SAMPL2
0002            0121 FOUTPUT  O   F   1 132     OF     TRACTR1                             SAMPL2


0003            0201 IDISKOUT NS  01   1 C0                                               SAMPL2
0004            0202 I                                    1   6 KEY                      SAMPL2
0005            0203 I                                   82  94 DESC                     SAMPL2
0006            0204 I                                  126 128ORECNBR                   SAMPL2


0007            0301 C   01      COUNT     ADD  1         COUNT   30                      SAMPL2


0008            0401 OOUTPUT  H  204   1P                                                SAMPL2
0009            0402 O        OR       OF                                                SAMPL2
0010            0403 O                                     5 'KEY'                       SAMPL2
0011            0404 O                                    22 'DESCRIPTION'               SAMPL2
0012            0405 O                                    30 'PAGE'                      SAMPL2
0013            0406 O                           PAGE  Z  35                             SAMPL2
0014            0407 O         D  1     01                                               SAMPL2
0015            0408 O                           KEY       6                            SAMPL2
0016            0409 O                           DESC     21                            SAMPL2
0017            0410 O                           RECNBRZ  25                            SAMPL2
0018            0411 O         T 3   01 LR                                              SAMPL2
0019            0412 O                           COUNT Z   3                            SAMPL2
0020            0413 O                                    26 'RECORDS WERE READ FROM'    SAMPL2
0021            0414 O                                    44 'THE INDEXED FILE.'         SAMPL2
```

Figure 4-10 (Part 1 of 2). Source Listing of Sample Program SAMPL2 (Model 6)

```
INDICATORS USED
       LR OF 1P 01

 FIELD NAMES USED
STMT#  NAME   DEC  LGTH  DISP
 0013 PAGE     0   004   011C
 0004 KEY          006   0105
 0005 DESC         013   0112
 0006 RECNBR   0   003   0115
 0007 COUNT    0   003   0118


                                        CORE USAGE OF RPGII CODE
  START   NAME IF   CODE     NAME     TITLE
  ADDR    OVERLAY  LENGTH
  0C00             068C     RGROOT   ROOT
  12F8             009D     RGMAIN   INPUT MAINLINE
  1294             0067     RGSUBS   INPUT CTRL RTN
  1398             003A     RGSUBS   RECORD ID
  13D2             0026     RGSUBS   CONTROL FIELDS
  128C             0008     RGSUBS   SUBSEG
  13F8             003B     $$ISIP   DISK IDX SEQ INPUT
  1433             0019     $$SRCR   SYSTEM SUBR
  144C             007B     $$SRBR   SYSTEM SUBR
  14C7             002C     $$SRIF   SYSTEM SUBR
  14F3             0081     $$SRRC   SYSTEM SUBR
  1574             0029     $$SRRI   SYSTEM SUBR
  159D             0043     $$SRIC   SYSTEM SUBR
  15E0             001C     $$SRTC   SYSTEM SUBR
  15FC             0081     $$SRMO   SYSTEM SUBR
  167D             0043     $$SRSB   SYSTEM SUBR
  16C0             0015     $$SRRD   SYSTEM SUBR
  16D5             002F     $$SRBP   SYSTEM SUBR
  1704             0015     $$SRPD   SYSTEM SUBR
  1719             002C     RGMAIN   INPUT FIELDS
  1746             0010     RGMAIN   DETAIL CALCS
  1745             0001     RGSUBS   CONSTANTS
  175C             0033     RGMAIN   DETAIL OUTPUT
  1756             0006     RGSUBS   CONSTANTS
  178F             000B     RGMAIN   TOTAL OUTPUT
  17BE             0024     RGMAIN   LR & OVERFLOW PROCESSING
  179A             0024     RGSUBS   OVERFLOW SUBSEGMENT
  17E2             0028     RGSUBS   SUBSEG
  182E             008E     RGMAIN   OPEN
  1816             0018     RGSUBS   CONSTANTS
  180A             000C     RGSUBS   SUBSEG
  1993             0018     RGMAIN   CLOSE
  18BC             00AA     RGSUBS   OUTPUT CTRL RTN
  1966             002D     RGSUBS   CONSTANTS
  19AB             0030     RGSUBS   LR PROCESSING
  19DB             0027     $$BDMC   SYSTEM SUBR

                 03586      SAMPL2   TOTAL CORE USAGE
```

DISK DATA MANAGEMENT ROUTINES

NON-DISK DATA MANAGEMENT LINKAGE ROUTINE

Figure 4-10 (Part 2 of 2). Source Listing of Sample Program SAMPL2 (Model 6)

Figure 4-11 (Part 1 of 4). Storage Dump of Sample Program SAMPL2 (Model 6)

Figure 4-11 (Part 2 of 4). Storage Dump of Sample Program SAMPL2 (Model 6)

Figure 4-11 (Part 3 of 4). Storage Dump of Sample Program SAMPL2 (Model 6)

```
                CONSTANTS
1960 0C00C087 17884000 03010001 D9C5C3D6 D9C4E240 E6C5D9C5 4089C5C1 C440C6D9  *<  GP.  CA ARECORDS WERE READ FR*
                                                  CLOSE
1980 D6D4E3C8 C5404C9D5 C4C5E7C5 C440C6C9 D3C54BFA 20C3C087 19AB3501 0C997502  *OMTHE INDEXED FILE.:-C GR>5A<R5R*
                          LR PROCESSING
19A0 05C08700 0483C087 00048404 0819DA5C 8F8F90C2 020C000C 050DB019 6E8C0202  *E G DC G DD4HR+*!!&BB< (E(OR,<BB*
                                                                  $$BDMC
19C0 0D1B6AF0 029B0200 908C1519 19818C10 2E1992C0 87180AC0 B712FBF2 870F5B5B  *(G:OB$B &(NRRA<&,RK GGC GK$2G!$$*
1DE0 C2C4D4C3 40F0F361 F1F861F7 F0340818 012C0119 FC1BC087 00048052 D402C087  *BDMC 03/18/704H+AZAR@$ G D KMB G*
                END OF SAMPL2
1A00 18B00000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  *0$                              *
1A20 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  *                                *
1A40 00C26200 00AE0160 24B50113 C0870000 0100E2E2 FFFF3408 1B31A0DJ 291DF282  * B8  )A(UEAL G  A $S≠≠4H$1..AZ)2B*
1A60 00C26200 00AE0160 24B50113 C0870000 0100E2E2 FFFF3408 1B31A0DJ 291DF282  *(9≠,2MB@0+ZC,W G  *A(I)A(R+A(S()*
1A80 D1B917BC 406EB501 0BB51009 00012104 04070409 09070409 04040204 02040404  *ARF@ +5A.5AI A/CBGUIIGDIIDEBDDD*
1AA0 D1B917BC 406EB501 0BB51009 00012104 04070409 09070409 04040204 02040404  *IBBBUWBAGGDDD=UI(DGDUSDDDHGGDDGD*
1AC0 09020402 04260204 07070404 04FE2409 0C040704 641B0404 04080707 04040704  *DBDDDGDIIG.&C=GBEE(.JBDGIDI(LDWFD*
1AE0 04020404 04070407 0F041310 06FE0704 05050C11 02040704 04080913 04266604  *GEEEE.CBB-  =                    G*
1B00 47050505 050B0362 02208080 50FE8080 80808080 80808080 80808080 80808016  *GGDDBGGCNGIDEJ=HDAM(CDMDD(T...DCNN*
1B20 04070404 04070A15 1B040405 11FE0804 10140D0A 04141604 0C090B0B 040C1515  *NDDDEC.NBBGDHD=GDDDEBS)V≠       *
1B40 15040404 050A1504 02070408 04FE0704 04040204 021D25FF 00000000 06000000  *) G*04E*SRG04R25BA4B'@G3RB*C*GGI *
1B60 2EC08710 70740278 E2020774 0232E202 01740270 7C0733C2 021CC39C 07078FC0  *G DZKB= G*22GC2GZ G$=4H*L} 47'-4*
1B80 87000469 D2027AC0 871CF2F2 8703F287 29C08718 FE340818 535E0034 377D6034  *2AP} 57SBA'H52DC G  @D)2GG@R)2G&*
1BA0 F281175E 003537E2 02017D00 35F2840A C0870000 7C042EF2 87167C02 2EF28710  *@A)25C$0?4B2* 3SBA(DBBPE(FF*G GM*
1BC0 7C012EF2 B7067BF0 2E740232 5C003335 C2010B04 C202175C 8C06061C 87C08714  */ D+III -  ACALL DIR SYSTEM F1R1*
1BE0 E140040E 09090900 60000001 03C1D3D3 40C4C9D9 40E2E8E2 E3C5D440 C6F1D9F1  *F2R2000039809999990000405       *
1C00 C6F2D9F2 F0F0F0F0 F3F9F8F0 F9F9F9F9 F9F9F0F0 F0F0F4F0 F5000040 40404040  *                *@    *@@$$RBDTSQ$$R*
1C20 40404040 40404040 40404040 4040401C F2000000 00001CBC D65B5B09 C2C4E3E2  *BSWS000000800000010DDDDU+GGLG'HHC*
1C40 C2E2E5E2 F0F0F0F0 F0F0F0F0 F0F0F0F0 F0F11604 0404240E 07071307 7D080B0A  *2G+$$RBLF07/07/704H:%4A;-4B1TBA;*
1C60 F2870E5B 5B09C2D3 D7F0F761 F0F761F7 F0340818 2E340118 1F34021E 23C2011E  *5%C*CBB¬ * CL$IK9-K2&W9 K2&C*ARN*
1C80 B56C0A1C 0AC2021F 005C0003 13780F12 796012F2 10E67940 12F2100A 5C011915  *@ †† K*AFR2 H@G). G:I* )W†@ /BB¬*
1CA0 7C001A7B 40125C01 0619F280 083C871D 4EC0871E 891C001D 661A7C00 21C2021F  * 5A*SR *   GJ_+ $D;K'-;02A() A *
1CC0 00750116 E2020690 00000CC0 871E2B0E 001E061E D23D601E D6F2B140 5D000100  *2AGKAA G)X@A KAA+ $D;K'-;02AY) A*
1CE0 F2810702 0101C087 1D67BC01 00E20101 0E00JED6 1ED23D60 1EB6F281 285D0001  * 2AR+  $K'≠  A)(+ $D;K'-;02AH G;*
1D00 90F20112 8E00001E D2BD3F00 C0011DBD 0E001ED6 1ED23D60 1ED6F281 08C0871E  *_ G)C G;..@  4R$G( $!G'=;02AV+ $*
1D20 2BC06710 B5C0871E 2BBC0000 34021E08 0C001ECF JED63DFE JED6F201 250E001E  *+!JL8-$+2&F+A;+$N(A$+$(2A((AA$+@ *
1D40 DE1EE338 601ECEF2 90060E01 1EDE1ED5 0D011ECE 1ECCF281 0C8C0101 1ECE3C00
```

Figure 4-11 (Part 4 of 4). Storage Dump of Sample Program SAMPL2 (Model 6)

```
     0101 H      008                                                                  SAMPL1


     0102 F**************************************************                         SAMPL1
     0103 F*                                                 *                         SAMPL1
     0104 F* THIS PROGRAM -                                  *                         SAMPL1
     0105 F*                                                 *                         SAMPL1
     0106 F*   1. LOADS 100 RECORDS TO AN INDEXED FILE.      *                         SAMPL1
     0107 F*                                                 *                         SAMPL1
     0108 F*   2. READS ONE RECORD FROM FILE $SOURCE FOR     *                         SAMPL1
     0109 F*      INPUT. THE FILE $SOURCE IS BUILT WHEN      *                         SAMPL1
     01091F*      SAMPLE PROGRAM SAMPL2 IS COMPILED BY       *                         SAMPL1
     01092F*      GIVING A RETAIN-T PARAMETER TO THE         *                         SAMPL1
     01093F*      FILE $SOURCE.                              *                         SAMPL1
     01094F*                                                 *                         SAMPL1
     0110 F*   3. CREATES THE OUTPUT DATA USING A            *                         SAMPL1
     0111 F*      LOOP IN THE CALCULATION SPECIFICATIONS.    *                         SAMPL1
     0112 F*                                                 *                         SAMPL1
     0113 F*   4. USES KEYS FROM 000005 THROUGH 000500       *                         SAMPL1
     0114 F*      IN INCREMENTS OF 5.                        *                         SAMPL1
     0115 F*                                                 *                         SAMPL1
     0116 F*   5. SHOULD BE FOLLOWED BY SAMPLE PROGRAM 2     *                         SAMPL1
     0117 F*      TO VERIFY THAT THE FILE WAS PROPERLY       *                         SAMPL1
     0118 F*      LOADED.                                    *                         SAMPL1
     0119 F*                                                 *                         SAMPL1
     0120 F**************************************************                         SAMPL1
0001 0121 F$SOURCE IP  F   96  96            DISK                                      SAMPL1
0002 0122 FDISKOUT O   F  256 128 06AI     1 DISK                      01              SAMPL1
0003 0123 FPRINTER O   F   96  96            PRINTER                                   SAMPL1


0004 0201 I$SOURCE NS  01                                                              SAMPL1
0005 0202 I                                        1    1 NODATA                       SAMPL1


0006 0301 C    01                Z-ADDO      COUNT   60                                SAMPL1
0007 0302 C    01                Z-ADDO      RECNBR  30                                SAMPL1
0008 0303 C           REPEAT     TAG                                                   SAMPL1
0009 0304 C    01     COUNT      ADD  5      COUNT                                     SAMPL1
0010 0305 C    01     RECNBR     ADD  1      RECNBR                                    SAMPL1
0011 0306 C    01     COUNT      COMP 505                    02                        SAMPL1
0012 0307 C    01N02             EXCPT                                                 SAMPL1
0013 0308 C    01N02             GOTO REPEAT                                           SAMPL1
0014 03081C                      SETON                  LR                             SAMPL1
0015 0309 CLR         RECNBR     SUB  1      RECNBR                                    SAMPL1


0016 0401 OPRINTER T  204    LR                                                        SAMPL1
0017 0402 O                                20 'SAMPLE PROGRAM 1 HAS'                   SAMPL1
0018 0403 O                                27 'LOADED'                                 SAMPL1
0019 0404 O                      RECNBRZ   31                                          SAMPL1
0020 0405 O                                39 'RECORDS'                                SAMPL1
0021 0406 O                                61 'INTO AN INDEXED FILE.'                  SAMPL1
0022 0408 O           T  2       LR                                                    SAMPL1
0023 0409 O                                21 'KEYS ARE IN ASCENDING'                  SAMPL1
0024 0410 O                                42 'SEQUENCE STARTING AT'                   SAMPL1
0025 0411 O                                64 '000005 AND INCREASING'                  SAMPL1
0026 0412 O                                84 'IN INCREMENTS OF 5.'                    SAMPL1
0027 0413 O           T     01 LR                                                      SAMPL1
0028 0414 O                                21 'SAMPLE PROGRAM 2 WILL'                  SAMPL1
0029 0415 O                                44 'PRINT FROM THE INDEXED'                 SAMPL1
0030 0416 O                                65 'FILE TO SHOW THAT IT'                   SAMPL1
0031 0417 O                                86 'WAS PROPERLY LOADED.'                   SAMPL1
0032 0501 ODISKOUT E         01N02                                                     SAMPL1
0033 0502 O                      COUNT      6                                          SAMPL1
0034 0503 O                                94 'RECORD NUMBER'                          SAMPL1
0035 0504 O                      RECNBR   128                                          SAMPL1
```

Figure 4-12 (Part 1 of 2). Source Listing of Sample Program SAMPL1 (Models 10 and 12)

```
 INDICATORS USED
       LR 01 02

RG 314 UNREFERENCED FIELD NAMES
 STMT#  NAME  DEC  LGTH  DISP
  0005 NODATA       001   0100

  FIELD NAMES USED
 STMT#  NAME  DEC  LGTH  DISP
  0005 NODATA       001   0100
  0006 COUNT    0   006   0106
  0007 RECNBR   0   003   0109

  ERROR SEVERITY                                        TEXT
RG  314 W       FIELD, TABLE OR ARRAY NAME DEFINED BUT NEVER USED.


                                        CORE USAGE OF RPGII CODE
    START  NAME IF  CODE     NAME     TITLE
    ADDR   OVERLAY  LENGTH
    1000            0642     RGROOT   ROOT
    1694            0091     RGMAIN   INPUT MAINLINE
    1725            0034     RGSUBS   RECORD ID
    1759            0026     RGSUBS   CONTROL FIELDS
    1642            004A     RGSUBS   INPUT CTRL RTN
    168C            0008     RGSUBS   SUBSEG
    177F            0027     $$CSIP   5444 CONSEC INPUT
    17A6            0079     $$SRBR   SYSTEM SUBR
    181F            0026     $$SRUA   SYSTEM SUBR
    1845            001C     $$SRTC   SYSTEM SUBR
    1861            0081     $$SRMO   SYSTEM SUBR
    18E2            0043     $$SRSB   SYSTEM SUBR
    1925            0038     $$SRDI   SYSTEM SUBR
    195D            002F     $$SRBP   SYSTEM SUBR
    198C            000B     RGMAIN   TOTAL CALCS
    1997            0022     RGMAIN   INPUT FIELDS
    19CA            004B     RGMAIN   DETAIL CALCS
    19C5            0005     RGSUBS   CONSTANTS
    1A45            0043     $$PGRI   RESET RESULTING INDR
    1A15            0030     RGSUBS   EXCEPTION
    19B9            000C     RGSUBS   SUBSEG
    1A88            0059     $$IOUT   5444 INDEXED OUTPUT
    1875            001C     $$SROF   SYSTEM SUBR
    1AE1            0094     $$SRBI   SYSTEM SUBR
    1B91            000B     RGMAIN   TOTAL OUTPUT
    1BC5            0024     RGMAIN   LR & OVERFLOW PROCESSING
    18A8            001D     RGSUBS   OVERFLOW SUBSEGMENT
    189C            G00C     RGSUBS   SUBSEG
    1BE9            00F8     $$LPRT   5203 PRINT
    1CE4            0072     RGMAIN   OPEN
    1EFE            002D     RGMAIN   CLOSE
    1FDC            000E     RGSUBS   CONSTANTS
    1D56            009D     RGSUBS   OUTPUT CTRL RTN
    1DF3            00E9     RGSUBS   CONSTANTS
    1EEA            0014     RGSUBS   LR CALCS
    1F2B            0076     RGSUBS   LR PROCESSING

                    04001    SAMPL1   TOTAL CORE USAGE REQUIRED TO EXECUTE
EL6EDP 0     3


TOTAL NUMBER OF LIBRARY SECTORS REQUIRED   18
```

Figure 4-12 (Part 2 of 2). Source Listing of Sample Program SAMPL1 (Models 10 and 12)

Figure 4-13 (Part 1 of 5). Storage Dump of Sample Program SAMPL1 (Models 10 and 12)

```
13A0 00000000 000040FF FF000001 0002111D 111D0000 00000000 00000000 00000000   *...... .........................*
13C0 0000C087 1E6EC087 16FE4340 00000000 00C3D95C 10000200 00000000 00000000   *.......... ......CR*............*
13E0 00000000 00000000 00000000 00000000 00000600 00000000 00000000 00000000   *................................*
1400 00000000 00000000 00000000 00000000 00000000 00004408 40A0008A 8000142C   *........................ .......*
1420 00000000 00C00000 14161181 00000003 00000000 00000000 00000000 00000000   *................................*
1440 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00030000   *................................*
1460 00000000 00000000 00050000 00000004 00000000 00000000 00000000 00000000   *................................*
1480 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00040000   *................................*
14A0 00000000 00000000 00060000 00000000 00000000 00000000 00000000 00000000   *................................*
14C0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   *................................*

1520 0C000000 00000000 00000000 BA5C40A1 00BA0000 15420020 004001DA 1181152C   *.............* .......... ......*
1540 0000FFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   *................................*
1560 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   *................................*

              INPUT CONTROL ROUTINE
1640 FFFF1644 3408168B 3401165D E2010075 02059C00 0F1B9C01 0D12C0FF 00C06C01   *............S.................%.*
1660 1200BD41 0EF28219 F281098D 440EF282 0DF2810D C20210B4 C0870004 8F407A80   *.....2..2.....2..2..B........ ..*
                                LINK TO  DISK INPUT INPUT MAINLINE
1680 00D20200 C2011000 C0870000 C201177F 35101643 7B02C97B F1C27B9F C37BFFC4   *.K..B.......B.......#.I#1B#.C#.D*
16A0 7A40C378 20C3C010 16D40C01 16BE1722 750299B8 1000F210 150C0117 1000001E   *. C..C...M............2.........*
16C0 0116BE97 B9C100B9 8001C010 170AB880 00F29006 7AFFC47A E0C3C087 1759C087   *.....A...........2....D..C......*
16E0 198CC087 1BC53C32 1709F287 0D3C3017 09B80118 F29003BC 001CBB01 00E20100   *.....E....2.........2........S..*
1700 C2021084 C0870004 8F007408 3CC08716 42B88000 F21004C0 871725BA 01007510   *B...................2...........*
                RECORD ID
1720 3C172416 8CB50112 85101434 081732C2 0100009C 0A220AF2 8708C087 16E6C087   *...............B.......2.....W..*
                                                                 CONTROL FIELDS
1740 16EDC201 1000C087 171BC087 172B8202 C9800019 B0C08717 3E7820C3 F2101535   *..B.............I..........C2...*
```

Figure 4-13 (Part 2 of 5). Storage Dump of Sample Program SAMPL1 (Models 10 and 12)

```
TBN  78 02 C9
JC   F2 90 0C
ZAZ  04 50 11 06  19C5
ZAZ  04 20 11 09  19C5
→TBN  78 02 C9
 JC   F2 90 2C
 AZ   06 50 11 06  19C6
 AZ   06 20 11 09  1EDC
 ZAZ  44 A5 10 11  06
 SZ   47 D2 10     19C9
 BC   C0 87 1A 45
 DATA  81 04 C9 00
 TBF  79 04 C9
 JC   F2 90 08
 BC   C0 87 1A 15
 BC   C0 87 19 0C
 SBN  7A 20 C3
 BC   C0 87 16 94
```

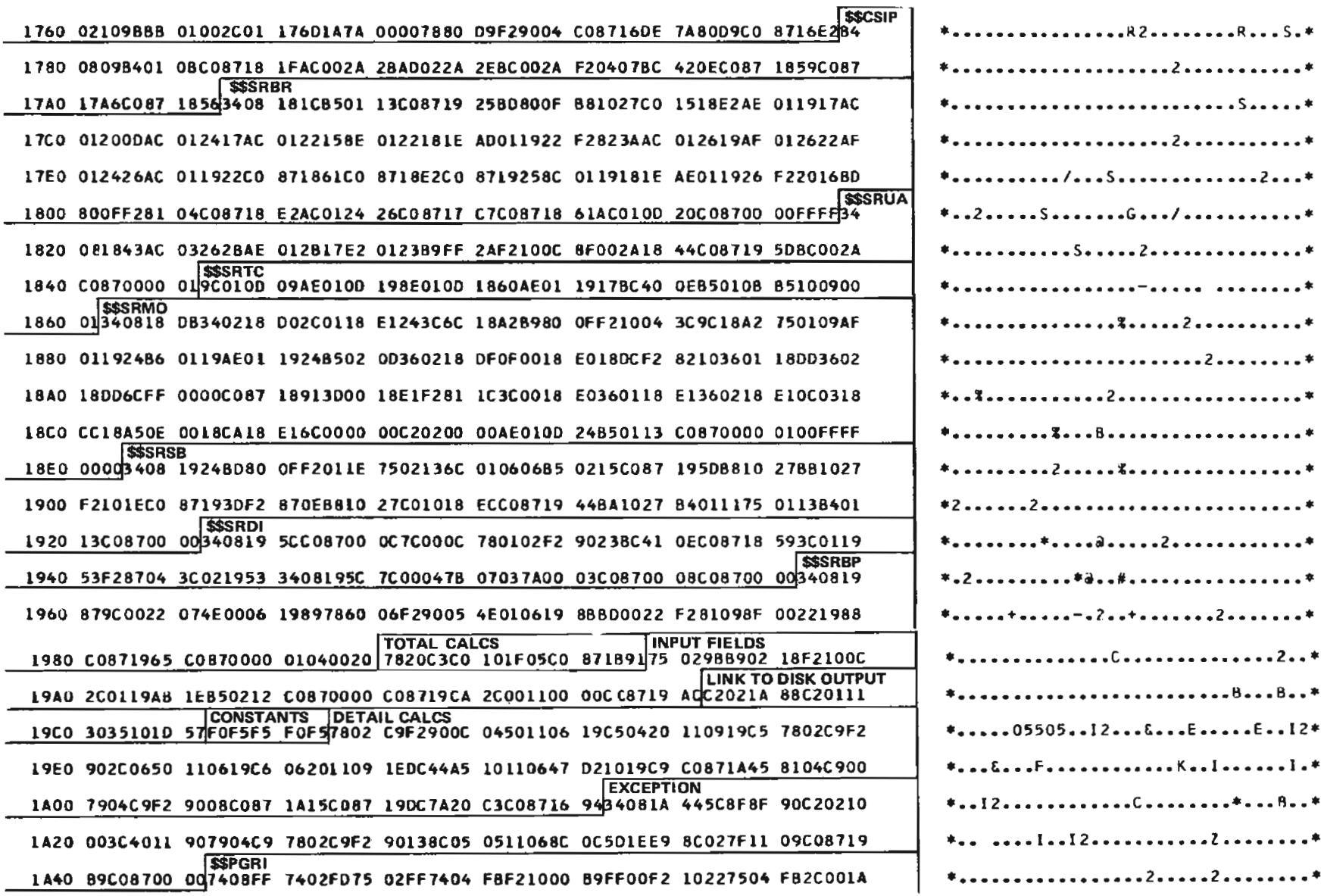
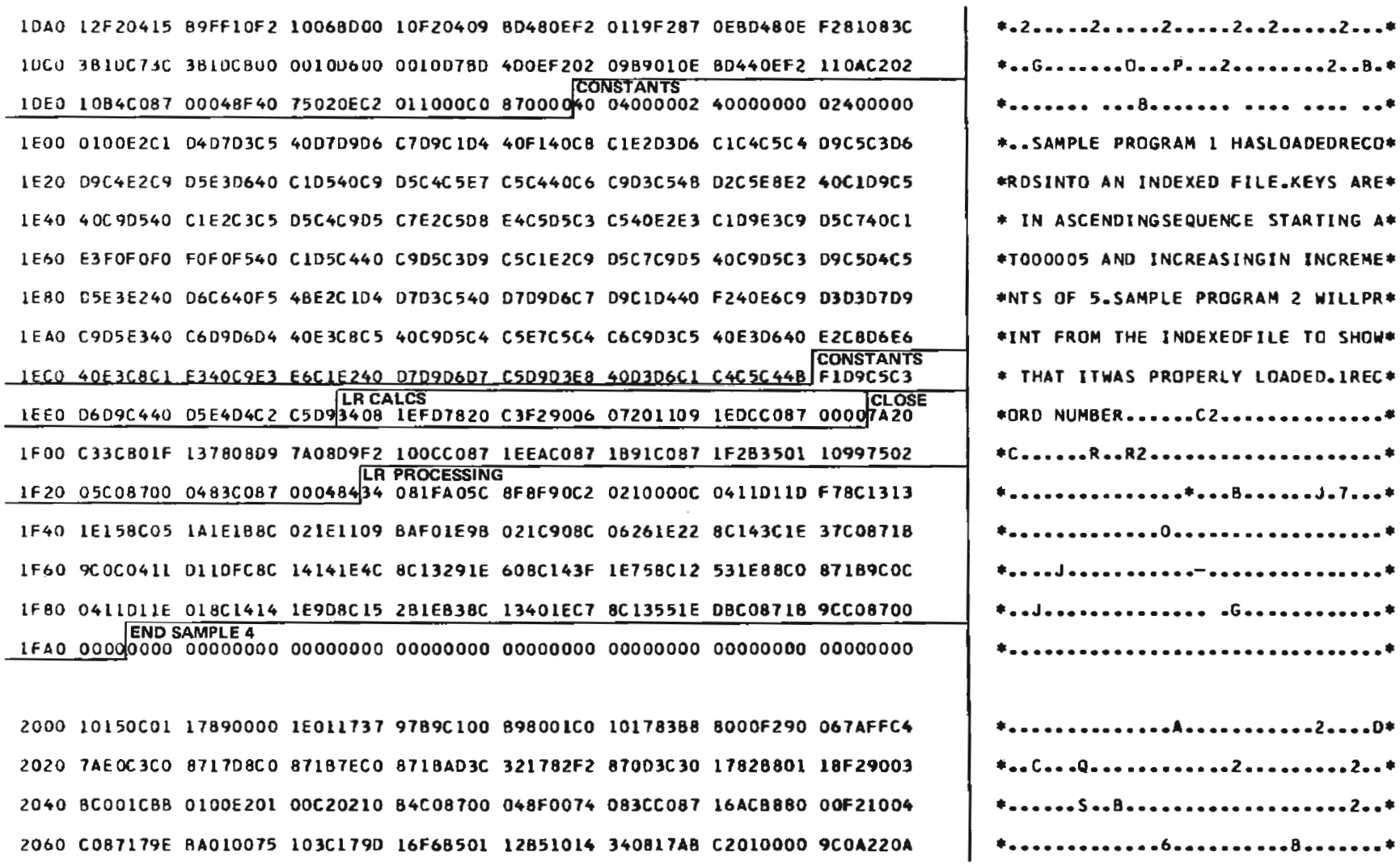


Figure 4-13 (Part 3 of 5). Storage Dump of Sample Program SAMPL1 (Models 10 and 12)


```
1A60 6300F200 043C7B1A 6E2C011A 70027A00 003C7A1A 6EE20203 C0871A51 E2020174  *..2...#.............S......S...*
1A80 02FF7502 FD7510FF B4080984 01088840 03F29032 B8012788 0127F210 293CFF1A  *.............. .2........2.....*
1AA0 822E001A B2382C01 1AB432B5 010DB601 3C1D0000 0000F282 0D8C600E F28103BC  *.....................2...-.2...*
1AC0 620EC087 1859C087 181FC087 1B758D70 0EC08118 59C0871A E1C08717 A6C08718  *................................*
1AE0 56340818 69AD012D 3AF2027E B50136C0 8719253C FF1B222E 00182238 750109AC  *.........2.....................*
1B00 012238AE 00222EB6 01223601 1B72AC00 2E228E00 2E1B742C 011B250D 2E011B25  *................................*
1B20 3C4C0C00 00008840 03F29003 B40132AC 002526AF 0124186C 020325BC FF22AE00  *....... .2.............%........*
1B40 222EAE00 2238F2A0 088E0022 1B74F220 15B50136 6C01062D C0871944 BC002EE2  *......2.......2.....%.........S*
1B60 0127C087 195DC087 0000BC70 0EC08718 59FFFF00 03340818 90AD0129 1DF2820D  *.............................2..*
1B80 B9FF28F2 1407BC70 0EAC032B 26C08700 007820C3 C0101F16 C08718C5 C2021BE9  *...2.............C........EB..Z*
1BA0 C2011141 35101D57 34081B8F 7A20D90C 0411D11B C4C08718 9C7B20D9 C0870000  *B.............R...J.D....#.R....*
1BC0 00060000 007820C3 C0101F1A 7801D7C0 101BA87B FFD778FF D87A02C2 7804C2F2  *.......C.......P....#.P#.Q..B..B2*
1BE0 10037B02 C2C08719 97340809 84010BBC 400EE201 10BA0414 C0871C4E E20111C0  *..#.B.......... .S........+S...*
1C00 871C4E88 400FC090 1CDEF1E2 00C0871C B72C001C 38220C02 1C371C38 3C001C36  *..+. .....1S.....................*
1C20 B5011F1E 011C3701 71E6015B 011870E6 0185010D 1C000000 00E20112 BA0614C0  *.........W.....W.........S......*
1C40 871C4E8C 01211C09 E20113F2 8709B408 217D0000 F2815A9C 001500BB 5B162C00  *..+.....S..2.....'..2.......$...*
1C60 1C68143A 041C68F1 0000C087 1CB7B0E0 18B50118 AC00241A B80814F2 9004AC00  *.......1.....................2....*
1C80 241B2C01 1C8915F3 00006C01 0515C1E0 1CB75C01 0305BB02 14AE0024 15B80414  *.......3..%...A...*.............*
1CA0 F29004AC 002415AD 002423F2 04038C48 0EBB0514 B5102134 081CDDB0 E31BB9E2  *2..........2.................T..S*
1CC0 1B89F71A F210102C 011CD51D BA1002C0 87000480 0000028D 410EC001 00008501  *..7.2.....N.....................*
1CE0 08851009 C0870004 00C20110 005F18DA DA6C00C5 377A01C2 7A40C375 01997502  *.........B........%.E...B. C.....*
1D00 05C08700 04887502 05C08700 04827502 05880103 F2900D7B C0007910 00F29004  *.....................2..#.....2..*
1D20 3C871D34 4D010310 99F28107 750103C0 871D0EF2 800AC202 1084C087 00048F11  *.........2.........2..B.........*
1D40 C2011000 3C401100 3CFC1109 0C071108 1109C087 16941D58 34081DF2 34021D74  *B.... ...0.................2....*
1D60 14F29004 AC002415 AD002423 F20403BC 480E8805 14851021 34081D9E B0E31BB9  *.2..........2.................T..*
1D80 E21BB9F7 1AF21010 2C011D96 1DBA1002 C0870004 80000002 8D410EC0 011D2FB5  *S..7.2.........................*
```

Figure 4-13 (Part 4 of 5). Storage Dump of Sample Program SAMPL1 (Models 10 and 12)




```
1DA0 12F20415 89FF10F2 10068D00 10F20409 8D480EF2 0119F287 0E8D480E F281083C  *.2.....2.....2.....2..2.....2...*
1DC0 3B1DC73C 3B1DC800 0010D600 0010D78D 4D0EF202 09B9010E 8D440EF2 110AC202  *..G.......O...P...2........2..B.*
1DE0 10B4C087 00048F40 75020EC2 011000C0 87000040 04000002 40000000 02400000  *....... ...8....... .... .... ..*
1E00 0100E2C1 D4D7D3C5 40D7D9D6 C7D9C1D4 40F140C8 C1E2D3D6 C1C4C5C4 D9C5C3D6  *..SAMPLE PROGRAM 1 HASLOADEDRECO*
1E20 D9C4E2C9 D5E3D640 C1D540C9 D5C4C5E7 C5C440C6 C9D3C54B D2C5E8E2 40C1D9C5  *RDSINTO AN INDEXED FILE.KEYS ARE*
1E40 40C9D540 C1E2C3C5 D5C4C9D5 C7E2C5D8 E4C5D5C3 C540E2E3 C1D9E3C9 D5C740C1  * IN ASCENDINGSEQUENCE STARTING A*
1E60 E3F0F0F0 F0F0F540 C1D5C440 C9D5C3D9 C5C1E2C9 D5C7C9D5 40C9D5C3 D9C5D4C5  *T000005 AND INCREASINGIN INCREME*
1E80 D5E3E240 D6C640F5 4BE2C1D4 D7D3C540 D7D9D6C7 D9C1D440 F240E6C9 D3D3D7D9  *NTS OF 5.SAMPLE PROGRAM 2 WILLPR*
1EA0 C9D5E340 C6D9D6D4 40E3C8C5 40C9D5C4 C5E7C5C4 C6C9D3C5 40E3D640 E2C8D6E6  *INT FROM THE INDEXEDFILE TO SHOW*
1EC0 40E3C8C1 E340C9E3 E6C1E240 D7D9D6D7 C5D9D3E8 40D3D6C1 C4C5C44B F1D9C5C3  * THAT ITWAS PROPERLY LOADED.1REC*
1EE0 D6D9C440 D5E4D4C2 C5D93408 1EFD7820 C3F29006 07201109 1EDCC087 00007A20  *ORD NUMBER......C2.............*
1F00 C33C801F 13780809 7A08D9F2 100CC087 1EEAC087 1B91C087 1F2B3501 10997502  *C.......R..R2.....................*
1F20 05C08700 0483C087 00048434 081FA05C 8F8F90C2 0210000C 0411D11D F78C1313  *.................*...B.......J.7...*
1F40 1E158C05 1A1E188C 021E1109 8AF01E98 021C908C 06261E22 8C143C1E 37C08718  *.............0.....................*
1F60 9C0C0411 D11DFC8C 14141E4C 8C13291E 608C143F 1E758C12 531E88C0 8718 9C0C  *....J...........-.................*
1F80 0411D11E 018C1414 1E9D8C15 2B1EB38C 13401EC7 8C13551E DBC08718 9CC08700  *..J.............. .G.............*
1FA0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  *................................*

2000 10150C01 17890000 1E011737 9789C100 898001C0 10178388 8000F290 067AFFC4  *.............A.....2....D*
2020 7AE0C3C0 8717D8C0 8718 7EC0 8718AD3C 321782F2 870D3C30 17828801 18F29003  *..C...Q.....2.....2..*
2040 8C001C8B 0100E201 00C20210 84C08700 048F0074 083CC087 16ACB880 00F21004  *.......S..B.....................2..*
2060 C087179E 8A010075 103C179D 16F68501 12851014 340817A8 C2010000 9C0A220A  *.............6.........B.......*
```

Figure 4-13 (Part 5 of 5). Storage Dump of Sample Program SAMPL1 (Models 10 and 12)

```
      0101 H      008                                                                    SAMPL2


      0102 F************************************************                             SAMPL2
      0103 F*                                               *                            SAMPL2
      0104 F* THIS PROGRAM -                                *                            SAMPL2
      0105 F*                                               *                            SAMPL2
      0106 F*   1. MUST BE PRECEDED BY SAMPLE PROGRAM 1     *                            SAMPL2
      0107 F*      WHICH LOADS AN INDEXED FILE.             *                            SAMPL2
      0108 F*                                               *                            SAMPL2
      0109 F*   2. READS AN INDEXED FILE SEQUENTIALLY.      *                            SAMPL2
      0110 F*                                               *                            SAMPL2
      0111 F*   3. USES A BLOCK LENGTH FOR DISK WHICH       *                            SAMPL2
      0112 F*      IS DIFFERENT FROM THAT USED FOR          *                            SAMPL2
      0113 F*      LOADING THE FILE IN SAMPLE PROGRAM 1.    *                            SAMPL2
      0114 F*                                               *                            SAMPL2
      0115 F*   4. COUNTS THE NUMBER OF RECORDS READ SO     *                            SAMPL2
      0116 F*      THAT THE USER CAN QUICKLY VERIFY THAT    *                            SAMPL2
      0117 F*      100 RECORDS WERE LOADED.                 *                            SAMPL2
      0118 F*                                               *                            SAMPL2
      0119 F************************************************                             SAMPL2
0001  0120 FDISKIN  IPE F 512 128 06AI      1 DISK                        01        SAMPL2
0002  0121 FPRINTER O   F  96  96      OF      PRINTER                               SAMPL2


0003  0201 IDISKIN  NS  01   1 CO                                                    SAMPL2
0004  0202 I                                      1   6 KEY                          SAMPL2
0005  0203 I                                     82  94 DESC                         SAMPL2
0006  0204 I                                    126 1280RECNBR                       SAMPL2


0007  0301 C   01       COUNT     ADD  1         COUNT   30                          SAMPL2


0008  0401 OPRINTER H  204   1P                                                      SAMPL2
0009  0402 O        OR       OF                                                      SAMPL2
0010  0403 O                                   5 'KEY'                               SAMPL2
0011  0404 O                                  22 'DESCRIPTION'                       SAMPL2
0012  0405 O                                  30 'PAGE'                              SAMPL2
0013  0406 O                        PAGE  Z   35                                     SAMPL2
0014  0407 O         D  1    01                                                      SAMPL2
0015  0408 O                        KEY        6                                     SAMPL2
0016  0409 O                        DESC      21                                     SAMPL2
0017  0410 O                        RECNBRZ   25                                     SAMPL2
0018  0411 O         T 3   01 LR                                                     SAMPL2
0019  0412 O                        COUNT Z    3                                     SAMPL2
0020  0413 O                                  26 'RECORDS WERE READ FROM'            SAMPL2
0021  0414 O                                  44 'THE INDEXED FILE.'                 SAMPL2
```

Figure 4-14 (Part 1 of 2). Source Listing of Sample Program SAMPL2 (Models 10 and 12)

```
INDICATORS USED
       LR OF 1P 01

 FIELD NAMES USED
STMT#  NAME  DEC  LGTH  DISP
 0013 PAGE    0   004   011C
 0004 KEY         006   0105
 0005 DESC        013   0112
 0006 RECNBR  0   003   0115
 0007 COUNT   0   003   0118


                                    CORE USAGE OF RPGII CODE
   START  NAME IF  CODE     NAME    TITLE
   ADDR   OVERLAY  LENGTH
   1000            06AC     RGROOT  ROOT
   16FE            00A0     RGMAIN  INPUT MAINLINE
   179E            003A     RGSUBS  RECORD ID
   17D8            0026     RGSUBS  CONTROL FIELDS
   16AC            004A     RGSUBS  INPUT CTRL RTN
   16F6            0008     RGSUBS  SUBSEG
   17FE            003B     $$ISIP  5444 IDX SEQ INPUT
   1839            0079     $$SRBR  SYSTEM SUBR
   18B2            0038     $$SRDI  SYSTEM SUBR
   18EA            006D     $$SRIC  SYSTEM SUBR
   1957            007B     $$SRRC  SYSTEM SUBR
   19D2            0029     $$SRRI  SYSTEM SUBR
   19FB            001C     $$SRTC  SYSTEM SUBR
   1A17            0081     $$SRMO  SYSTEM SUBR
   1A98            0043     $$SRSB  SYSTEM SUBR
   1ADB            002F     $$SRBP  SYSTEM SUBR
   1B0A            002C     RGMAIN  INPUT FIELDS
   1B37            0010     RGMAIN  DETAIL CALCS
   1B36            0001     RGSUBS  CONSTANTS
   1B4C            0032     RGMAIN  DETAIL OUTPUT
   1B47            0005     RGSUBS  CONSTANTS
   1B7E            0008     RGMAIN  TOTAL OUTPUT
   1BAD            0024     RGMAIN  LR & OVERFLOW PROCESSING
   1B89            0024     RGSUBS  OVERFLOW SUBSEGMENT
   1BF4            008E     RGMAIN  OPEN
   1BDD            0017     RGSUBS  CONSTANTS
   1BD1            000C     RGSUBS  SUBSEG
   1CAA            00FB     $$LPRT  5203 PRINT
   1C82            0028     RGSUBS  SUBSEG
   1E6E            0021     RGMAIN  CLOSE
   1E42            002C     RGSUBS  CONSTANTS
   1DA5            009D     RGSUBS  OUTPUT CTRL RTN
   1E8F            0030     RGSUBS  LR PROCESSING

                  03775     SAMPL2  TOTAL CORE USAGE REQUIRED TO EXECUTE
EL6EDP 0     3


TOTAL NUMBER OF LIBRARY SECTORS REQUIRED   16
```

Figure 4-14 (Part 2 of 2). Source Listing of Sample Program SAMPL2 (Models 10 and 12)

```
PRIME WORK AREA
1000 F0F0F0F0 F6F54040 D9C5C3D6 D9C440D5 E4D4C2C5 D94040F1 F3404040 40404040   *000065  RECORD NUMBER  13       *
1020 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *                                 *
                                                  CONSTANTS
1080 40404040 40404040 40404040 40404040 40FFFF00 00010002 111D111D 00000000   *                ..............*
10A0 00000000 00000000 00000000 C0871E6E C08716FE 43400000 000000C3 D95C1000   *..................... .....CR*..*
     INDICATORS                                                        SEC WORK
10C0 02000040 00000000 00020000 00000000 00000000 00000000 00800000 06000000   *... ............................*
AREA
10E0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   *................................*
     FIELDS                                                            IOCB1
1100 F0F0F0F0 F6F5D9C5 C3D6D9C4 40D5E4D4 C2C5D9F0 F1F3F0F1 F3F0F0F0 F1020411   *000065RECORD NUMBER0130130001...*
1120 43115400 00003A00 00801000 01001300 17C31000 008202C9 80001823 C0871787   *..................C.....I........*
     IOCB2                                        DISK DTF
1140 10000010 00111D11 9C000000 3A200060 10000200 A0E38801 FFFF119C 16C8111D   *................-.....T.......H..*
1160 13004080 13801380 02000080 007FC480 C6000216 AB0100C4 98008001 C4800000   *.. ...........D.F.....D....D...*
                                                                  PRINTER DTF
1180 C40C9016 320000C4 00159600 06C48000 05000000 00000000 00000000 E00040C3   *D......D.....D................. C*
11A0 1154FFFF 1DC41143 10004040 00000001 E2010412 60001200 11C01266 1D9F5F3C   *.....D....  ....S...-...........*
     NOT USED
11C0 12000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   *................................*
11E0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   *................................*

1260 127CE201 E201127C 00126600 00000000 00000000 00000000 00000000 F0F0F0F0   *.aS.S..a.....................0000*
1280 F6F54040 D9C5C3D6 D9C440D5 E4D4C2C5 D94040F1 F3404040 40404040 40404040   *65  RECORD NUMBER  13           *
12A0 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *                                 *

1300 F0F0F0F0 F6F54040 40404040 40404040 40404040 40404040 40404040 40404040   *000065                          *
1320 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *                                 *
1340 40404040 40404040 40404040 40404040 40D9C5C3 D6D9C440 D5E4D4C2 C5D94040   *                 RECORD NUMBER   *
1360 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40F0F1F3   *                                013*
     DISK IOB                                                  DISK PHYSICAL BUFFER
1380 C49C40A1 00C49801 13960020 000018BA 11541380 1154F0F0 F0F0F6F5 40404040   *D. ..D.............000065     *
13A0 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *                                 *
```

Figure 4-15 (Part 1 of 5). Storage Dump of Sample Program SAMPL2 (Models 10 and 12)

```
13E0 40404040 40404040 C5C3D6D9 C440D5E4 D4C2C5D9 40404040 40404040 40404040  *     RECORD NUMBER          *
1400 40404040 40404040 40404040 40404040 404040F0 F1F3F0F0 F0F0F7F0 40404040  *                 013000070  *
1420 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040  *                            *
1460 40404040 40404040 C5C3D6D9 C440D5E4 D4C2C5D9 40404040 40404040 40404040  *     RECORD NUMBER          *
1480 40404040 40404040 40404040 40404040 404040F0 F1F4F0F0 F0F0F7F5 40404040  *                 014000075  *
14A0 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040  *                            *
14E0 40404040 40404040 C5C3D6D9 C440D5E4 D4C2C5D9 40404040 40404040 40404040  *     RECORD NUMBER          *
1500 40404040 40404040 40404040 40404040 404040F0 F1F5F0F0 F0F0F8F0 40404040  *                 015000080  *
1520 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040  *                            *
1560 40404040 40404040 C5C3D6D9 C440D5E4 D4C2C5D9 40404040 40404040 40404040  *     RECORD NUMBER          *
1580 40404040 40404040 40404040 40404040 404040F0 F1F6C400 40A100C4 000015AC  *                 016D. ..D....*
15A0 00200000 188A1154 15961300 F0F0F0F0 F0F50000 00F0F0F0 F0F1F000 0080F0F0  *.....000005...000010...00*
15C0 F0F0F1F5 000400F0 F0F0F0F2 F0000480 F0F0F0F0 F2F50008 00F0F0F0 F0F3F000  *0015...000020...000025...000030.*
15E0 0880F0F0 F0F0F3F5 000C00F0 F0F0F0F4 F0000C80 F0F0F0F0 F4F50010 00F0F0F0  *..000035...000040...000045...000*
1600 F0F5F000 1080F0F0 F0F0F5F5 001400F0 F0F0F0F6 F0001480 F0F0F0F0 F6F50018  *050...000055...000060...000065..*
1620 00F0F0F0 F0F7F000 188F0F0 F0F0F7F5 001C00F0 F0F0F0F8 F0001C80 F0F0F0F0  *.000070...000075...000080...0000*
1640 F8F50020 00F0F0F0 F0F9F000 2080F0F0 F0F0F9F5 002400F0 F0F0F1F0 F0002480  *85...000090...000095...000100...*
1660 F0F0F0F1 F0F50028 00F0F0F0 F1F1F000 2880F0F0 F0F1F1F5 002C00F0 F0F0F1F2  *00105...000110...000115...00012*
1680 F0002C80 F0F0F0F1 F2F50030 00F0F0F0 F1F3F000 3080F0F0 F0F1F3F5 003400F0  *0...000125...000130...000135...0*
16A0 F0F0F1F4 F0003480 FFFFFFFF 16AE3408 16F53401 16C7E201 00750205 9C000F18  *00140.....5...GS.........*
16C0 9C010D12 C0FF17FE 6C01120D BD410EF2 8219F281 098D440E F2820DF2 810DC202  *.........3.....2...2..2..B.*
16E0 10B4C087 00048F40 7A8000D2 0200C201 1000C087 178AC201 17FE3510 16AD7920  *.....  ...K..B.....B.........*
1700 D7792008 F2900378 20D67820 D87802C9 78F1C278 9FC378FF C47A40C3 7820C3C0  *P..Q2..#.O#.Q#.I#1B#.C#.D. C..C.*
```

INPUT CONTROL ROUTINE

LINK TO DISK INPUT INPUT

MAINLINE

Figure 4-15 (Part 2 of 5). Storage Dump of Sample Program SAMPL2 (Models 10 and 12)

```
1720 10174D0C 01173717 98750299 881000F2 10150C01 1789179F 1E011737 9789C100  *.............2.............A.*
1740 898001C0 10178388 8000F290 067AFFC4 7AE0C3C0 8717D8C0 87187EC0 8718AD3C  *.........2....D..C...Q.........*
1760 321782F2 870D3C30 17828801 18F29003 BC001C88 0100E201 00C20210 84C08700  *...2.........2........S..8......*
1780 048F0074 083CC087 16F68880 00F21004 C087179E BA010075 103C179D 16F68501  *.........6...2.............6..*   RECORD ID
17A0 12851014 340817AB C20117CD 9C0A220A F28708C0 87175FC0 871766C2 011000C0  *........B.......2.........B....*
17C0 87179470 F000F201 08C08717 A48202C9 80001823 C0871787 7820C3F2 10153502  *...*0.2........I.........C2....*   CONTROL FIELDS LOGIC
17E0 109B8B01 002C0117 EC1A7A02 C9788009 F29004C0 8717577A 80D9C087 17583408  *............I..R2........R...$..*   AND MOVE   $$ISIP
1800 09B40108 B84027F2 9007BC42 0EC0871A 0FC08718 39C08719 27884027 F2101685  *..... .2.................. .2...*
1820 0136C087 18B2C087 1957BDFF 22C08119 D2C08718 EAC0871A 0C340818 AF850113  *.................K.............*   $$SRBR
1840 C0871882 BD800F88 1027C015 1A98AE01 1917AC01 200DAC01 2417AC01 22158E01  *................................*
1860 2218B1AD 011922F2 823AAC01 2619AF01 2622AF01 2426AC01 1922C087 1A17C087  *.......2.......................*
1880 1A98C087 18828C01 1918B1AE 011926F2 2016BD80 0FF28104 C0871A98 AC012426  *..............2.....2.........*
18A0 C087185A C0871A17 AC010D20 C0871815 FFFF3408 18E9C087 000C7C00 0C780102  *.....................Z....@.....*   $$SRDI
18C0 F29023BC 410EC087 1A0F3C01 18E0F287 043C0218 E0340818 E97C0004 7B07037A  *2.............2.........Z@..#...*
18E0 0103C087 0008C087 18263408 19518501 36AE0130 388E0130 19568C01 2019539E  *.............................*   $$SRIC
1900 012009AD 012030F2 02448C01 3019549E 013009AE 0130388E 01301956 C0871ADB  *.......2.....................*
1920 C08718CA F2872734 08195185 0136AC01 26309F01 26099C01 2506B801 02AD0226  *....2.........................*
1940 2EF29004 AD022641 F282038A 4027C087 183500FF FF000334 0819CAB5 01309C02  *.2.....2... .....................*   $$SRRC
1960 2500AE01 2418AC00 26258C00 25B50113 9D012406 F28241F2 81208E00 2519CD9D  *.....................2..2........*
1980 002507F2 84328F01 2419CF88 7C24F290 058F0124 19D1C087 19708C01 1919CCAE  *...2.........@.2......J..........*
19A0 011926AE 011917AD 011915F2 020AAF01 1917BC00 22F2870F BCFF22BC 00258C01  *.............2.........2.........*
19C0 1919CCAE 011926C0 87182AFF FF010004 00203408 19F68501 30360119 FA1C0119  *.....................6...........*   $$SRRI
19E0 F8002E01 19F81B85 01134C01 0619F8C0 8718CAC0 87000000 00FFFF9C 010D09AE  *8....8.........8.................*   $$SRTC
1A00 010D198E 010D1A16 AE011917 BC400E85 0108B510 09000134 081A9134 021A862C  *.............. .....................*   $$SRMO
1A20 011A9724 3C6C1A58 B9800FF2 10043C9C 1A587501 09AF0119 24860119 AE011924  *.....%.....2.....................*
1A40 85020D36 021A950F 001A961A 92F28210 36011A93 36021A93 9CFF0000 C0871A47  *.............2...................*
1A60 3D001A97 F2811C3C 001A9636 011A9736 021A970C 031A821A 580E001A 801A979C  *....2.....................$.......*
```

Figure 4-15 (Part 3 of 5). Storage Dump of Sample Program SAMPL2 (Models 10 and 12)

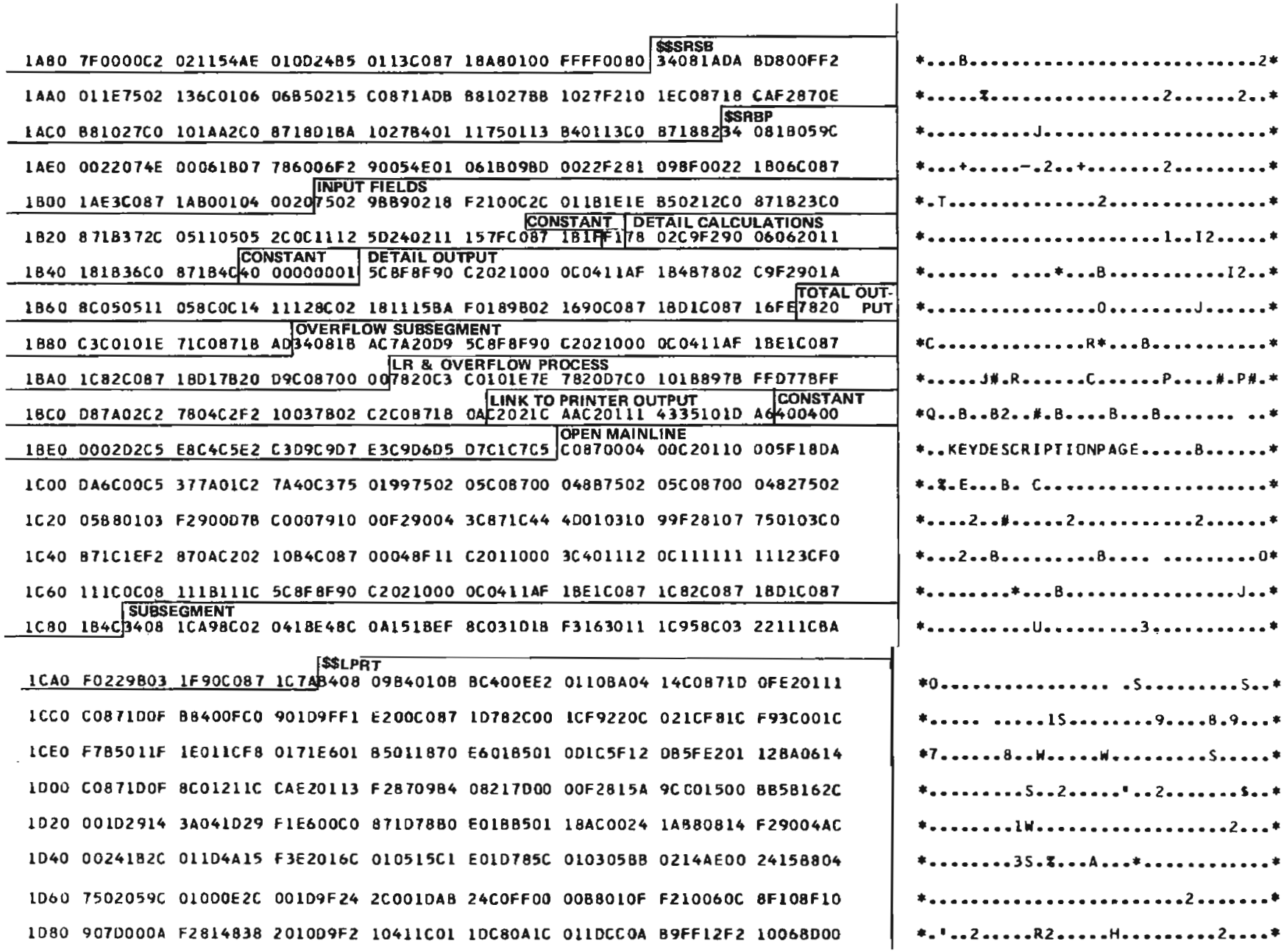


Figure 4-15 (Part 4 of 5). Storage Dump of Sample Program SAMPL2 (Models 10 and 12)

```
                    OUTPUT CONTROL ROUTINE
1DA0 010BB510 091DA734 081E4134 021DC375 02059C01 0D0E2C00 1DEE242C 001DFA24  *.............C.................*
1DC0 C0FF1CAA B8010FF2 10060C8F 108F1090 7D000AF2 81483820 10D9F210 411C011E  *.......2........'..2.....R2.....*
1DE0 170A1C01 1E180AB9 FF12F210 06BD1212 F20415B9 FF10F210 06BD1210 F20409BD  *..........2.....2.....2.....2...*
1E00 480EF201 19F2870E BD480EF2 81083C3B 1E163C3B 1E1A3A20 10D63A20 10D7BD4D  *..2..2.....2.............O...P..*
1E20 0EF20209 B9010EBD 440EF211 0AC20210 B4C08700 048F4075 020EC201 1000C087  *.2........2..B........ ...B.....*
          CONSTANTS
1E40 1B7A4000 03010009 C5C3D6D9 C4E240E6 C5D9C540 D9C5C1C4 40C6D9D6 D4E3C8C5  *.. ....RECORDS WERE READ FROMTHE*
                    CLOSE MAINLINE
1E60 40C9D5C4 C5E7C5C4 40C6C9D3 C54B7A20 C37808D9 7A08D9F2 1004C087 1E8F3501  * INDEXED FILE...C..R..R2........*
                    LR PROCESSING
1E80 10997502 05C08700 0483C087 00048434 081EBE5C 8F8F90C2 0210000C 0411AF1E  *.....................*...B........*
                                                                    END
1EA0 468C0202 11188AF0 029B0200 908C1519 1E5C8C10 2B1E6DC0 8718D1C0 87000000  *.......0.........*........J......*
SAMPLE 2
1EC0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  *................................*


1F00 00C20200 00AE010D 24B50113 C0870000 0100E2E2 FFFF3408 1F31AD01 291DF282  *.B................SS..........2.*
1F20 0DB9FF28 F21407BC 700EAC03 2B26C087 00009C01 0D09AE01 0D198E01 0D1F4DAE  *....2.............................*
1F40 011917BC 400EB501 08B51009 0001210A 04070409 09070409 04040204 02040404  *.... .............................*
1F60 09020402 04260204 07070404 04FE2409 0C040704 041B0404 04080707 04040704  *.............................*
1F80 04020404 04070409 0F041310 0AFE0704 05050C11 0204070F 040D0913 04260604  *.............................*
1FA0 47050505 050B0302 02208080 80FE8080 80808080 80808080 80808080 80808016  *.............................*
1FC0 04070404 04070A15 1B040405 11FE0804 10140D0A 04141604 0C090B0B 040C1515  *.............................*
1FE0 15040404 050A1504 02070408 04FE0704 04040204 021D25FF 00000000 00000000  *.............................*
2000 10150C01 17890000 1E011737 9789C100 B98001C0 10178388 8000F290 067AFFC4  *.............A..........2....D*
2020 7AE0C3C0 8717D8C0 871B7EC0 8718AD3C 321782F2 870D3C30 1782B801 18F29003  *..C...Q..........2.........2..*
2040 BC001C8B 0100E201 00C20210 B4C08700 048F0074 083CC087 16ACB880 00F21004  *......S..B.................2..*
2060 C087179E BA010075 103C179D 16F68501 12B51014 340817AB C2010000 9C0A220A  *.............6.........B.......*
2080 F28708C0 87175FC0 871766C2 011000C0 8717947D F000F201 08C08717 A48202C9  *2.........B.......'0.2........I*
20A0 80001B23 C0871787 7820C3F2 10153502 1098BB01 002C0117 EC1A7A00 007880D9  *..........C2...................R*
```

Figure 4-15 (Part 5 of 5). Storage Dump of Sample Program SAMPL2 (Models 10 and 12)

```
                0101 H     008                                                                  SAMPL1


                0102 F*************************************************                      SAMPL1
                0103 F*                                                *                      SAMPL1
                0104 F* THIS PROGRAM -                                 *                      SAMPL1
                0105 F*                                                *                      SAMPL1
                0106 F*   1. LOADS 100 RECORDS TO AN INDEXED FILE.     *                      SAMPL1
                0107 F*                                                *                      SAMPL1
                0108 F*   2. READS ONE RECORD FROM FILE $SOURCE FOR    *                      SAMPL1
                0109 F*      INPUT. THE FILE $SOURCE IS BUILT WHEN     *                      SAMPL1
                01091F*      SAMPLE PROGRAM SAMPL2 IS COMPILED BY      *                      SAMPL1
                01092F*      GIVING A RETAIN-T PARAMETER TO THE        *                      SAMPL1
                01093F*      FILE $SOURCE.                             *                      SAMPL1
                01094F*                                                *                      SAMPL1
                0110 F*   3. CREATES THE OUTPUT DATA USING A           *                      SAMPL1
                0111 F*      LOOP IN THE CALCULATION SPECIFICATIONS.   *                      SAMPL1
                0112 F*                                                *                      SAMPL1
                0113 F*   4. USES KEYS FROM 000005 THROUGH 000500      *                      SAMPL1
                0114 F*      IN INCREMENTS OF 5.                       *                      SAMPL1
                0115 F*                                                *                      SAMPL1
                0116 F*   5. SHOULD BE FOLLOWED BY SAMPLE PROGRAM 2    *                      SAMPL1
                0117 F*      TO VERIFY THAT THE FILE WAS PROPERLY      *                      SAMPL1
                0118 F*      LOADED).                                  *                      SAMPL1
                0119 F*                                                *                      SAMPL1
                0120 F*************************************************                      SAMPL1
0001            0121 F$SOURCE IP  F 256 256            DISK                                  SAMPL1
0002            0122 FDISKOUT O   F 256 128 06AI     1 DISK                          01       SAMPL1
0003            0123 FPRINTER O   F  96  96            PRINTER                               SAMPL1


0004            0201 I$SOURCE NS  01                                                         SAMPL1
0005            0202 I                                    1   1 NODATA                       SAMPL1


0006            0301 CLO                    Z-ADD0        COUNT   60                       SAMPL1
0007            0302 CLO                    Z-ADD0        RECNBR  30                       SAMPL1
0008            0303 CLO          REPEAT    TAG                                            SAMPL1
0009            0304 CLO          COUNT     ADD  5        COUNT                            SAMPL1
0010            0305 CLO          RECNBR    ADD  1        RECNBR                           SAMPL1
0011            0306 CLO          COUNT     COMP 505                    02                 SAMPL1
0012            0307 CLO    N02               EXCPT                                          SAMPL1
0013            0308 CLO    N02               GOTO REPEAT                                    SAMPL1
0014            03081CLO                      SETON                     LR                 SAMPL1
0015            0309 CLR          RECNBR    SUB  1        RECNBR                           SAMPL1
0016            0401 OPRINTER T   204   LR                                                   SAMPL1
0017            0402 O                                    20 'SAMPLE PROGRAM 1 HAS'          SAMPL1
0018            0403 O                                    27 'LOADED'                        SAMPL1
0019            0404 O                          RECNBRZ   31                               SAMPL1
0020            0405 O                                    39 'RECORDS'                       SAMPL1
0021            0406 O                                    61 'INTO AN INDEXED FILE.'         SAMPL1
0022            0408 O         T  2     LR                                                  SAMPL1
0023            0409 O                                    21 'KEYS ARE IN ASCENDING'         SAMPL1
0024            0410 O                                    42 'SEQUENCE STARTING AT'          SAMPL1
0025            0411 O                                    64 '000005 AND INCREASING'         SAMPL1
0026            0412 O                                    84 'IN INCREMENTS OF 5.'           SAMPL1
0027            0413 O         T         LR                                                  SAMPL1
0028            0414 O                                    21 'SAMPLE PROGRAM 2 WILL'         SAMPL1
0029            0415 O                                    44 'PRINT FROM THE INDEXED'        SAMPL1
0030            0416 O                                    65 'FILE TO SHOW THAT IT'          SAMPL1
0031            0417 O                                    86 'WAS PROPERLY LOADED.'          SAMPL1
0032            0501 ODISKOUT E           N02                                              SAMPL1
0033            0502 O                          COUNT      6                               SAMPL1
0034            0503 O                                    94 'RECORD NUMBER'                 SAMPL1
0035            0504 O                          RECNBR   128                               SAMPL1
```

Figure 4-16 (Part 1 of 3). Source Listing of Sample Program SAMPL1 (Model 15)

```
 INDICATORS USED
        LR LO 01 02

RG 305  INDICATORS UNREFERENCED
        01

RG 314 UNREFERENCED FIELD NAMES
 STMT#  NAME
  0005 NODATA

  FIELD NAMES USED
 STMT#  NAME  DEC  LGTH  DISP
  0006 COUNT   0   006   0005
  0007 RECNBR  0   003   0008

  LABELS USED
 STMT#  NAME   TYPE
  0008 REPEAT  TAG

  ERROR NUMBER  STATEMENT NUMBER
    RG 273            0032

  ERROR SEVERITY                                        TEXT
RG  273 W       OUTPUT INDICATORS IN COL 23-31 MISSING OR ALL NEGATIVE.
RG  305 W       INDICATOR ASSIGNED BUT NOT USED TO CONDITION OPERATIONS.
RG  314 W       FIELD, TABLE OR ARRAY NAME DEFINED BUT NEVER USED.
```

Figure 4-16 (Part 2 of 3). Source Listing of Sample Program SAMPL1 (Model 15)

```
                OVERLAY LINKAGE EDITOR CORE USAGE MAP    11/19/73

 START    CATEGORY  NAME AND       CODE LENGTH
ADDRESS             ENTRY      HEXADECIMAL DECIMAL

 4000               GLOBAL           07C8     1992
 47C8               COMMON           0009        9
 4800          0    SAMPL1           0100      256
 4900          0    $#RT02           0123      291
 4A23          0    $#IPCR           004F       79
 4A24               $@0AC9
 4A72          0    $#OPCR           009C      156
 4A73               $@0B18
 480E          0    $#CON0           00E9      233
 48F7          0    $#CON1           0005        5
 48FC          0    $#CON2           000E       14
 4C0A          2    $$CSIP           0027       39
 4C31          2    $$IOUT           005F       95
 4C90          2    $$SRBR           0082      130
 4D12          2    $$SRUA           0026       38
 4D38          2    $$SRTC           001C       28
 4D38               DMSRLO
 4D49               DMSRTC
 4D4C               DMSRER
 4D54          2    $$SRBI           0105      261
 4E59          2    $$SRDF           001C       28
 4E75          2    $$SRMO           00A4      164
 4F19          2    $$SRSB           0046       70
 4F5F          2    $$SRDI           003E       62
 4F84               DMSRPD
 4F7D               DMSRRO
 4F9D          2    $$SRBP           002F       47
 4FCC          6    $$LPRT           0001      209
 509D         93    $#OPEN           006D      109
 510A        126    $#INPT           0090      144
 515C               $@0C25
 5163               $@0C2C
 5190               $@0C59
 5154               $@0C1D
 5158               $@0C21
 519A         28    $#IH01           0008        8
 51A2        126    $#TCAL           0052       82
 51F4         29    $#EXPT           0020       45
 5221         28    $#OH02           000C       12
 522D        126    $#IFLD           0010       29
 5246               $@0CE9
 524A         93    $#CLOS           002D       45
 5251               $@0EC2
 5262               $@0ED3
 5266               $@0ED7
 5275        107    $#LROT           0076      118
 52E8        126    $#TOUT           0008       11
 52F6        126    $#LRDF           0024       36
 531A         71    $#DAOF           0010       29
 5337         28    $#OH03           000C       12
 5343        126    $#RCID           0046       70
 537A               $@0C9A
 5389        126    $#CFLD           0026       38
 53AF         11    $$PGR1           0043       67


 START    CATEGORY  NAME AND       CODE LENGTH
ADDRESS             ENTRY      HEXADECIMAL DECIMAL

 53F2        107    $#LRC            0014       20

OL100 I  THE TOTAL CORE USED BY SAMPL1 IS  5126 DECIMAL.
OL101 I  THE START CONTROL ADDRESS OF THIS MODULE IS 4800.
OL104 I  TOTAL NUMBER OF LIBRARY SECTORS REQUIRED IS   13
         NAME-SAMPL1,PACK-SYSTEM,UNIT-R1,RETAIN-T,LIBRARY-O
```

Figure 4-16 (Part 3 of 3). Source Listing of Sample Program SAMPL1 (Model 15)

```
OPTION-PL1    ID-$ IAR-5021 PMR-0072 PSR-0004 XR2-4A02 XR1-4000 ARR-4FDF
      GLOBAL (BUFFERS AND IOBS)
4000  8012A500 000000E0 000E407C 08010000 129D0C14 AE420040 19000040 00000000  *..........@.. .................*
4020  E0000040 7C000000 00000004 00000000 40008820 90C09043 DB381053 FFF29014  *....@.....................2..*
4040  0C014555 46EEB880 82F29008 3C484550 3C87406D C20245A4 F4100088 3C444550  *.........2.....&...._B...4......&*
4060  C20245A4 F4100082 350246EA C08043AB B8408EF2 90960C05 47015457 40404040  *B...+.............2...........*
4080  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040  *................................*
      DUPLICATE LINES
4100  459F0C01 484C484E C0874092 C2024704 BC0100F4 1000863C 004111BD 5000C091  *.....<.+....B......4........&...*
4120  41DCBD60 00C08141 3C3B8048 5CF28745 B80200F2 100EB801 00F210A0 C2024750  *...-........*2.....2.....2..8..&*
4140  F4100085 C2024735 380846FA F2101EC2 02473EC2 0154C93D A846FAF2 8203D201  *4...8.......2..8...B..I....2..K.*
4160  12D20109 797006F2 90038A04 06F41000 85C08740 92B53104 C202454E 38024858  *.K.....2.....4.........B..+...$*
4180  F290128C 100FC087 4260BD41 0EC08145 463B0248 5B2C0141 A30D0E01 41A3484C  *2........-.........$.........<*
41A0  1C5F0000 5FJE0148 4C484A2D 01484C15 C004410C 3A024858 BC400FC0 874250BD  *.¬..¬...<.....<........$......-.*
41C0  700EC081 453EBD44 0EC08145 3E8E014F 459F0C01 484C484E C087410C 350246EA  *.............|.....<.+........*
41E0  88208FC2 02454E38 02485BF2 930EBC10 0FC08742 608D410E C0814546 3880485C  *...8..+...$2........-.........**
4200  40004000 000000A2 0000421E 00000000 00000000 00004200 49487C61           *..............................B/*
4220  007C5C01 0001484C 45A1F284 088C0115 45A1F287 0E000148 4C45A3F2 84058C01  *.@*....<..2.......2.....<..2....*
4240  1545A33A 02485BBC 400FC087 4260BD70 0EC08145 3EB0440E C081453E C08743AB  *......$......-.................*
4260  B40809B4 010BB501 11C08743 549C010D 08BC400E B9C00FF2 106EAC01 2E4FC087  *.....................2.>...|..*
4280  42EE6C02 1634F204 16BC002D AC002E14 8F002E43 746C0017 2EF20209 BC0114BC  *..%...2.............%...2......*
42A0  440EF287 43AE012E 4FC08742 EE=28418 B8400FF2 9006BC70 0EF2872C BC420E9F  *..2.....|....2.....2.....2.....*
42C0  00144374 C087427A B8800F7B 0707F290 067A0107 F287037A 02077C00 08F41000  *.......:...#..2..:..2..:..@..4..*
42E0  02B8100F C0104354 B5010BB5 10093408 435388C0 00F21018 AC01341C 886034F2  *.....................2........-.2*
4300  10088F01 2E4374F2 823B8E01 344376C0 8742FCAC 01331CBC 0134BD14 34F28411  *.......2.....................2..*
4320  8F012E43 74F28224 40904000 00C804A2 00004364 00200300 12904000 00C80400  *.....2.......H..............H..*
4340  00004346 4996C09D 400000C8 00A20300 44640020 00001290 400000C8 00000000  *...........H..............H....*
4360  43284996 F0F0F0F4 F9F54040 40404040 40404040 40404040 40404040 40404040  *....000495....................*
4380  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040  *................................*
```

Figure 4-17 (Part 1 of 6). Storage Dump of Sample Program SAMPL1 (Model 15)

```
43A0  40404040 40404040 40404040 40404040 40404040 40D9C5C3 D6D9C440 D5E4D4C2  *.....RECORD.NUMB*
43C0  C5D94040 40404040 40404040 40404040 40404040 40404040 40404040 40404040  *ER..............................*
43E0  40F0F9F9 F0F3F0F5 F0F04040 40404040 40404040 40404040 40404040 40404040  *.09900500......................*
4400  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040  *..............................*
4420  40404040 40404040 40404040 40404040 40404040 40D9C5C3 D6D9C440 D5E4D4C2  *.....................RECORD.NUMB*
4440  C5D94040 40404040 40404040 40404040 40404040 40404040 40404040 40404040  *ER..............................*
4460  40F1F0F0 F0F3F0F4 F8F54040 40404040 40404040 40404040 40404040 40404040  *.1000048S......................*
4480  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040  *..............................*
44A0  40404040 40404040 40404040 40404040 40404040 40D9C5C3 D6D9C440 D5E4D4C2  *.....................RECORD.NUMB*
44C0  C5D94040 40404040 40404040 40404040 40404040 40404040 40404040 40404040  *ER..............................*
44E0  40F0F9F7 F3F3F0F4 F3F04040 40404040 40404040 40404040 40404040 40404040  *.0970304.......................*
4500  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040  *..............................*
4520  40404040 40404040 40404040 40404040 40404040 40D9C5C3 D6D9C440 D5E4D4C2  *.....................RECORD.NUMB*
4540  C5D94040 40404040 40404040 40404040 40404040 40404040 40404040 40404040  *ER..............................*
4560  40F0F9F8 40904300 00C584A2 00200000 12904000 00C58500 00004582  *.098....F.....F.....F.....*
4580  49964090 40003000 38A20000 45A00020 00001290 40000006 88000000 45644996  *.....F.....F.....F.....*
45A0  F0F0F0F4 F2F500C8 00F0F0F0 F4F3F000 C880F0F0 F0F3F5 00CC00F0 F0F0F4F4  *000425.H.000430...00044*
45C0  F000CC80 F0F3F0F4 F4F50000 00F0F0F0 F4F50000 D090F0F3 F0F4F5F5 000400F0  *0...000445...000450...000455.M.0*
45E0  F0F0F4F6 F000D480 F0F0F0F4 F6F500D8 00F0F0F0 F4F7F000 D880F0F0 F0F4F7F5  *000460.M.000465.Q.000470.Q.000475*
4600  00DC00F0 F3F0F4F8 F3000C80 F0F0F0F4 F8F50100 30F0F0F0 F4F9F001 0080F0F0  *...000480...000485...000490...00*
4620  F0F4F9F5 01040300 F0F0F5F0 F3010480 F0F0F0F2 F2F50058 00F0F0F0 F2F3F000  *0495...000500...003225...000230.*
4640  5880F0F0 F3F2F3F5 005C00F0 F0F0F2F4 F0005C80 F0F0F0F2 F4F50080 00F0F0F0  *...000235.*.000240.*.000245...000*
4660  F2F5F000 8080F0F0 F0F2F5F5 008400F0 F0F0F2F6 F3008480 F0F0F0F2 F6F50088  *250...000255...000260...000265..*
4680  00F0F0F0 F2F7F000 8880F0F0 F0F2F7F5 008C00F3 F0F3F2F3 F3008C80 30518D00  *.000270...000275...000280.......*
46A0  F0F3F0F2 F8F53090 00F0F0F0 F2F9F000 9080F0F0 F0F2F9F5 009400F0 F0F0F3F0  *000285...000290...000295...00030*
46C0  F3009480 F0F0F0F3 F0F50398 00F0F0F0 F3F1F000 9880F0F0 F0F3F1F5 009C00F0  *0...000305...000310...000315...0*
46E0  F0F0F3F2 F0009C80 F0F0F0F3 F2F500A0 00F0F0F0 F3F3F000 A080F0F0 F0F3F3F5  *00320...000325...000330...000335*
```

Figure 4-17 (Part 2 of 6). Storage Dump of Sample Program SAMPL1 (Model 15)

Figure 4-17 (Part 3 of 6). Storage Dump of Sample Program SAMPL1 (Model 15)

```
                                              OUTPUT CONTROL ROUTINE
4A60  1000BF40 7A8000D2 0200C201 4800C087 5186 4A74 34084B0D 34024A90 7502059C   *....:..K..B.....................*
4A80  010D0E2C 004AB824 2C004AC7 24C0FF4F CC88010F F21D060C 8F488F48 9070000A   *...........3...|....2........'..*
4AA0  F2814838 2048D9F2 10411C01 4AE40A1C 014AE80A 89FF12F2 1006BD08 12F20415   *2.....R2.....U....Y....2.....2..*
4AC0  89FF10F2 1006BD08 10F20409 BD480EF2 0119F287 0EBD480E F281083C 3B4AE33C   *...2.....2.....2..2.....2.....T.*
4AE0  3B4AE700 0048D600 0048D73D 4D0EF202 0989010E BD440EF2 1109C202 48B4F410   *..X...O...P.(.2.........2..B...4.*
                                      CONSTANTS
4B00  008F4075 020EC201 4800C087 52AB 4004 00000240 00000002 40000000 01E2C1D4   *......B.....................SAM*
4B20  D7D3C540 D7D9D6C7 D9C1D440 F140C8C1 E2D3D6C1 C4C5C4D9 C5C3D6D9 C4E2C9D5   *PLE.PROGRAM.1.HASLOADEDRECORDSIN*
4B40  E3D640C1 D540C9D5 C4C5E7C5 C440C6C9 D3C54BD2 C5E8E240 C1D9C540 C9D540C1   *TO.AN.INDEXED.FILE.KEYS.ARE.IN.A*
4B60  E2C3C5D5 C4C9D5C7 E2C5D8E4 C5D5C3C5 40E2E3C1 D9E3C9D5 C740C1E3 F0F0F0F0   *SCENDINGSEQUENCE.STARTING.AT0000*
4B80  F0F540C1 D5C440C9 D5C3D9C5 C1E2C9D5 C7C9D540 C9D5C3D9 C5D4C5D5 E3E240D6   *05.AND.INCREASINGIN.INCREMENTS.O*
4BA0  C640F548 E2C1D4D7 D3C54007 D9D6C7D9 C1D440F2 40E6C9D3 D3D7D9C9 D5E340C6   *F.5.SAMPLE.PROGRAM.2.WILLPRINT.F*
4BC0  D9D6D440 E3C8C540 C9D5C4C5 E7C5C4C6 C9D3C540 E3D640E2 C8D6E640 E3C8C1E3   *ROM.THE.INDEXEDFILE.TO.SHOW.THAT*
                                                                   CONSTANTS CONSTANTS
4BE0  40C9E3E6 C1E240D7 D9D6D7C5 D9D3E840 D3D6C1C4 C5C44BF0 F5F5F0F5 F1D9C5C3   *.ITWAS.PROPERLY.LOADED.055051REC*
                              $$CSIP
4C00  D6D9C440 D5E4D4C2 C5D9 B408 09840108 C0874D12 AC004041 AD024040 8C0040F2   *ORD.NUMBER........(.........(...2*
                                          $$OUT
4C20  04073C42 0EC0874D 4CC0874C 90C0874D 49 B40809 B4010B8D 700EF281 34B84029   *.......(<..<...(...........2.....*
4C40  F2903288 803AB880 3AF21029 3CFF4C61 2E004C61 5A2C014C 6353B501 00860160   *2........2....</..</...<.......-*
4C60  10054623 00F2820D 8C500EF2 81038C62 0EC0874D 4CC0874D 12C0874E 59BD700E   *.....2...-.2.......(<..(...+....*
                                        $$SRBR
4C80  C0814D4C C0874054 C0874C90 C0874D49 34084D0F B50113C0 874F5F80 800FF281   *..(<..(...<...(...(.......|¬...2.*
4CA0  03BA0238 B80139C0 154F19AE 011917AC 012E0DAC 013417AC 0130158E 01304D11   *.........|....................(.*
4CC0  AD011930 F2823DAC 013819AF 013830AF 013433AC 01193300 874F75C0 874F19C0   *....2......................+...|..*
4CE0  874F5F8C 01194D11 AE011938 F220198D 800FF281 07C0874F 193A0233 AC013439   *.|¬....(.....2.....2....|.........*
                                           $$SRUA
4D00  C0874C87 C0874E75 AC010D2E C0874C8C FFFF 3408 4D36AC03 3841AF01 4117F201   *..<...+.......<.....(.........S.*
                                                            $$SRTC
4D20  2939FF40 F2100C8F 00404D37 C0874F9D 8C004C0 874C7901 9C010006 AE010019   *....2.....(...|......<...........*
                                                 $$SRBI
4D40  8E01004D 53AE0119 17BC400E B5010B85 10090001 34094F47 AD014C5E F202F935   *...(..................+...<;2.Z.*
4D60  0169C087 4F5FB801 38F21000 6C01164C B80102F2 90038A08 383CFF40 057E0340   *....|¬...2..%..<...2.......(3..(*
4D80  D65A7501 08AC0130 5AAE0030 40860130 36014E54 86016B4C 0040304F 00404E56   *O...........(.....+.......(...(+.*
4DA0  2C014DD9 002E014D D9608820 29F29025 0C0040C1 40062C01 4DC5532C 0140C371   *..(R...(R-...2....(A(O..(E...(C.*
4DC0  00000000 0000F202 0C0C044D 044DC50C 00000000 004C0500 4805BA01 38884029   *......2....(M(E.......<..........*
```

Figure 4-17 (Part 4 of 6). Storage Dump of Sample Program SAMPL1 (Model 15)

```
4DE0  F2900384 0153AC00 3738AF01 361C6C02 0337BCFF 3DAE0030 4DAF0030 5AF2A008  *2.....8.....(....2..*
4E00  8E030304E 56F2203C B5016960 00176A8E 006A4E58 3C034E02 0136C087 4F9DAD00  *...+.2.....%.....+...(S...|...*
4E20  6451F282 1F8C006A B5016900 874F848B 0138B804 02F29002 35016984 01587501  *./2.....|.....2.....*
4E40  18840169 C0874C88 AC034138 BC700ECD 874D4CFF FF000300 01134084E 74A3013F  *.....<.....(<.....+....*
                                                  $$SRMO          $$SRDF
4E60  1FF2820D B9FF41F2 1407BC70 0EAC0341 38C0874C 70134084F 12340246E F82C014F  *.2.....2.....<'..|...+...|*
4E80  18343C6C 4EC39980 0FF21004 3C9C4ECD 75010888 40398D40 0FF2960E 4F017734  *...%+....2....+.....2.....*
4EA0  860177AE 017734F2 8708AF01 19348601 19AE0119 34850200 36024F16 0F034F17  *.....2.....|...|.*
4EC0  4F13F282 1036014F 1436024F 146CFF00 00C0874E 8C3D004F 18F2811C 3C004F17  *|.2....|...|.%.....+...|.2....|.*
4EE0  36014F18 36024F18 0C034EF7 4ED00E00 4EF54F18 6C7F0300 C2024996 AE010D34  *.....|...+7+...+5|.%*..B.....*
                                                                  $$SRSB
4F00  B5011388 40398D20 3FF29603 B5011100 874CD801 03FFFF00 80134084F 5EBD800F  *.....2.....<.....|:....*
4F20  F2011E75 02136C01 16168502 10C0874F 90880139 BB0139F2 102103B7 4F7DF287  *2.....%.....|.....2....|'2.*
4F40  11880139 C0104F23 C0874F84 8A01398B 02388401 11750118 B4011300 874CDF34  *.....|...|.....<..*
      $$SRDI
4F60  084F9C78 4300F210 04F41000 037C000F 780102F2 90238C41 0EC0874D 4C3C014F  *.|....2..4....@.....2.....(<..|*
                                                                  $$SRBP
4F80  93F28704 3C024F93 34084F9C 7C00087B 07077A32 07F41000 02C0874C F334084F  *.2....|...|.@..#...:...4.....<T..|*
4FA0  C79C0030 174E0016 4FC97860 16F29005 4E011640 C8BD0030 F281098F 00304FC8  *G....+...|(.-.2..+...|....2.....|H*
                  $$LPRT                                                  OPEN
4FC0  C0874FA5 C0874D30 01040020 84080984 313B8C40 0EE20110 8A0414C0 875006E2  *.|...(.....S.....&.S*
4FE0  011BC087 5005B840 0FC09050 97E20112 8A06148A 0102C087 500B3C01 50934FEC  *....&.....&.S.....&...&.|.*
5000  E20113F2 870A3408 50937D00 00F2817D 9C001500 85011778 1000F210 04F41000  *S..2....&.'..2.'.....2..4..*
5020  03750118 8401177D 4002F201 67880102 F2902220 00504E18 0C02504D 504E3C00  *.....'..2.....2....&+...&(&+..*
5040  504C1E01 504D0885 010D1C30 00000085 01178801 029C001E 166C0108 15F41000  *&<..&(.....%...4..*
5060  02750118 9E001E08 780407F2 900495C0 1E088832 14AE031E 15880414 F29004AC  *.....2.....2...*
                                                                  OPEN
5080  001E15A0 001E1CF2 04033348 0EB80414 C0874FDF 8C410E85 01088513 09350230  *.....2.....|.....*
50A0  2EC20148 005F18DA DA6C0035 827A01C2 7A40C375 01997502 05F41000 8R750205  *.B...¬...%.E.:.8:.C.....4.....*
50C0  F4100082 1C00500A 10380048 C5F2900D 78C00079 1000F290 043C8750 ED4D0103  *4.....&.....E2..#.....2....&.(..*
50E0  4899F281 07750103 C0875004 F2870902 024884F4 10008F11 C2014800 3CF047D0  *..2.....&02..8...4....B....0..*
                INPUT MAINLINE
5100  0C0747CF 47D0C087 510A7802 C978F1C2 7B9FC378 FFC47A40 C37923C3 C0105140  *.....#.I#18#.C#.D:.C..C....*
5120  0C015134 51977502 99881000 F2101500 01518551 981E0151 3497B9C1 00B98001  *.....2.....A....*
5140  C0105170 880000F2 900674FF C47AE0C3 C0875389 C0875142 C0875206 3C32517E  *..."...2...:.D:.C.....6...=*
```

Figure 4-17 (Part 5 of 6). Storage Dump of Sample Program SAMPL1 (Model 15)

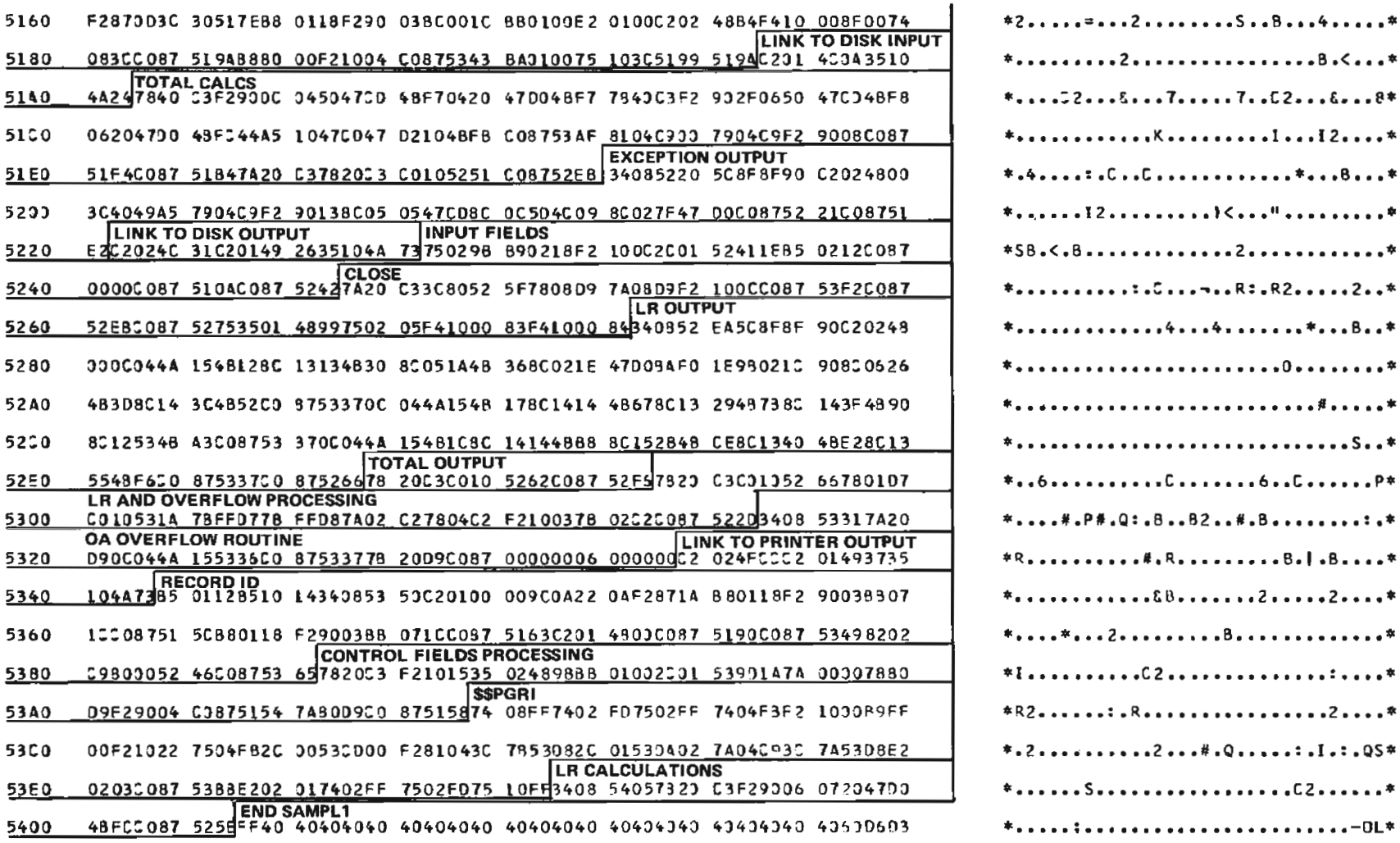


Figure 4-17 (Part 6 of 6). Storage Dump of Sample Program SAMPL1 (Model 15)

```
                0101 H     008                                                                  SAMPL2


                0102 F*************************************************                  SAMPL2
                0103 F*                                                *                  SAMPL2
                0104 F* THIS PROGRAM -                                 *                  SAMPL2
                0105 F*                                                *                  SAMPL2
                0106 F*   1. MUST BE PRECEDED BY SAMPLE PROGRAM 1      *                  SAMPL2
                0107 F*      WHICH LOADS AN INDEXED FILE.              *                  SAMPL2
                0108 F*                                                *                  SAMPL2
                0109 F*   2. READS AN INDEXED FILE SEQUENTIALLY.       *                  SAMPL2
                0110 F*                                                *                  SAMPL2
                0111 F*   3. USES A BLOCK LENGTH FOR DISK WHICH        *                  SAMPL2
                0112 F*      IS DIFFERENT FROM THAT USED FOR           *                  SAMPL2
                0113 F*      LOADING THE FILE IN SAMPLE PROGRAM 1.     *                  SAMPL2
                0114 F*                                                *                  SAMPL2
                0115 F*   4. COUNTS THE NUMBER OF RECORDS READ SO      *                  SAMPL2
                0116 F*      THAT THE USER CAN QUICKLY VERIFY THAT     *                  SAMPL2
                0117 F*      100 RECORDS WERE LOADED.                  *                  SAMPL2
                0118 F*                                                *                  SAMPL2
                0119 F*************************************************                  SAMPL2
0001            0120 FDISKIN  IPE F 512 128 06AI     1 DISK                         01     SAMPL2
0002            0121 FPRINTER O   F  96  95      OF     PRINTER                             SAMPL2


0003            0201 IDISKIN  NS  01   1 CO                                               SAMPL2
0004            0202 I                                    1   6 KEY                      SAMPL2
0005            0203 I                                   82  94 DESC                     SAMPL2
0006            0204 I                                  126 128ORECNBR                   SAMPL2


0007            0301 C   01       COUNT     ADD  1         COUNT   30                    SAMPL2


0008            0401 OPRINTER H  204   1P                                                SAMPL2
0009            0402 O        OR       OF                                                SAMPL2
0010            0403 O                                     5 'KEY'                       SAMPL2
0011            0404 O                                    22 'DESCRIPTION'               SAMPL2
0012            0405 O                                    30 'PAGE'                      SAMPL2
0013            0406 O                          PAGE   Z  35                            SAMPL2
0014            0407 O         D  1     01                                               SAMPL2
0015            0408 O                          KEY       6                            SAMPL2
0016            0409 O                          DESC     21                            SAMPL2
0017            0410 O                          RECNBRZ  25                            SAMPL2
0018            0411 O         T 3   01 LR                                              SAMPL2
0019            0412 O                          COUNT  Z  3                            SAMPL2
0020            0413 O                                    26 'RECORDS WERE READ FROM'    SAMPL2
0021            0414 O                                    44 'THE INDEXED FILE.'         SAMPL2



INDICATORS USED
       LR OF 1P 01

 FIELD NAMES USED
STMT#  NAME   DEC  LGTH  DISP
 0013 PAGE     0   004   001C
 0004 KEY          006   0005
 0005 DESC         013   0012
 0006 RECNBR   0   003   0015
 0007 COUNT    0   003   0018
```

Figure 4-18 (Part 1 of 2). Source Listing of Sample Program SAMPL2 (Model 15)

```
        OVERLAY LINKAGE EDITOR CORE USAGE MAP     11/19/73

 START     CATEGORY   NAME AND        CODE LENGTH
ADDRESS               ENTRY     HEXADECIMAL  DECIMAL

 4000                 GLOBAL          04D2      1234
 44D2                 COMMON          001D        29
 4500         0       SAMPL2          0100       256
 4600         0       $#RTO2          00C4       196
 46C4         0       $#IPCR          004F        79
 46C5                   $@06D3
 4713         0       $#OPCR          009C       156
 4714                   $@0722
 47AF         0       $#CON0          0005         5
 47B4         0       $#CON1          002C        44
 47E0         0       $#CON2          0001         1
 47E1         0       $#CON3          0017        23
 47F8         2       $$ISIP          004F        79
 4847         2       $$SRMO          00A4       164
 48EB         2       $$SRDI          003E        62
 4910                   DMSRPD
 4909                   DMSRRD
 4929         2       $$SRIC          00D3       211
 4981                   DMSRIF
 49FC         2       $$SRRC          0088       187
 4AB7         2       $$SRRI          0032        50
 4AE9         2       $$SRTC          001C        28
 4AE9                   DMSRLO
 4AFA                   DMSRTC
 4AFD                   DMSRER
 4B05         2       $$SRBP          002F        47
 4B34         6       $$LPRT          00D1       209
 4C05        93       $#OPEN          008D       141
 4C92       126       $#INPT          009F       159
 4CF3                   $@0832
 4CFA                   $@0839
 4D27                   $@0866
 4CEB                   $@082A
 4CEF                   $@082E
 4D31        28       $#IHO1          0008         8
 4D39       126       $#IFLD          002C        44
 4D52                   $@08F8
 4D65       126       $#DCAL          0010        16
 4D75        93       $#CLOS          001F        31
 4D78                   $@09F7
 4D85                   $@0A04
 4D94       107       $#LROT          0030        48
 4DC4       126       $#DOUT          0032        50
 4DF6       126       $#TOUT          0008        11
 4E01       126       $#LROF          0024        36
 4E25        71       $#OFOF          0024        36
 4E49        28       $#OH02          000C        12
 4E55        35       $#MF01          0028        40
 4E7D       126       $#RCID          004C        76
 4EB4                   $@08A7
 4EC9       126       $#CFLD          0026        38

IL100 I  THE TOTAL CORE USED BY SAMPL2 IS  3823 DECIMAL.
IL101 I  THE START CONTROL ADDRESS OF THIS MODULE IS 4500.
IL104 I  TOTAL NUMBER OF LIBRARY SECTORS REQUIRED IS   10
         NAME-SAMPL2,PACK-SYSTEM,UNIT-R1,RETAIN-T,LIBRARY-O
```

Figure 4-18 (Part 2 of 2). Source Listing of Sample Program SAMPL2 (Model 15)

```
 OPTION-PL1     ID-$ IAR-4889 PMR-0072 PSR-0001 XR2-46A3 XR1-4019 ARR-4BEC
       GLOBAL (BUFFERS & IOBS)
4000   40004000 000300E2 D13E437C 10030000 129D0C14 AF421040 198012A5 00000000   *.......S...0.....................*
4020   E2010E40 7C080100 00129D04 14AE000F 40008820 90C09043 0B3B1058 FFF29014   *S...0.....................2..*
4040   0C014555 46EE3880 82F29008 3C484550 3C474060 C20245A4 F4100389 3C444550   *.....2.....&...._3...4.....E*
4060   C20245A4 F4100082 350246EA C08043AB 88408EF2 90960C05 47015457 F0F0F0F0   *B...4.....2.....0000*
4080   F5F54040 D9C5C3D6 D9C44005 F4042205 D94040F1 F1404040 40404040 40404040   *55..RECORD.NUMBER..11.....*
40A0   40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *................................*
       DUPLICATE LINES
4100   F0F0F0F0 F5F04040 40404040 40404040 40404040 40404040 40404040 40404040   *00050.....................*
4120   40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *................................*
4140   40404040 40404040 40404040 40404040 4009C5C3 D6D9C440 D5E4D4C2 C5D94040   *.................RECORD.NUMBER..*
4160   40404040 40404040 40404040 40404040 40404040 40404040 40404040 40F0F1F2   *.............................012*
4180   C0904000 00C714A1 0000419E 00200000 129D4000 00C71331 00004180 4537F0F0   *.....G.............G........00*
41A0   F0F0F4F5 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *0045...........................*
41C0   40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *................................*
41E0   40404040 40404040 40404040 40404009 C5C3D6D9 C440D5E4 D4C2C5D9 40404040   *................RECORD.NUMBER....*
4200   40404040 40404040 40404040 40404040 40404040 40404040 404040F0 F0F9F0F0   *.............................00900*
4220   F0F0F5F0 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *0050...........................*
4240   40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *................................*
4260   40404040 40404040 40404040 40404009 C5C3D6D9 C440D5E4 D4C2C5D9 40404040   *................RECORD.NUMBER....*
4280   40404040 40404040 40404040 40404040 40404040 40404040 404040F0 F1F0F0F0   *.............................01000*
42A0   F0F0F5F5 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *0055...........................*
42C0   40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *................................*
42E0   40404040 40404040 40404040 40404009 C5C3D6D9 C440D5E4 D4C2C5D9 40404040   *................RECORD.NUMBER....*
4300   40404040 40404040 40404040 40404040 40404040 40404040 404040F0 F1F1F0F0   *.............................01100*
4320   F0F0F6F0 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *0060...........................*
```

Figure 4-19 (Part 1 of 5). Storage Dump of Sample Program SAMPL2 (Model 15)

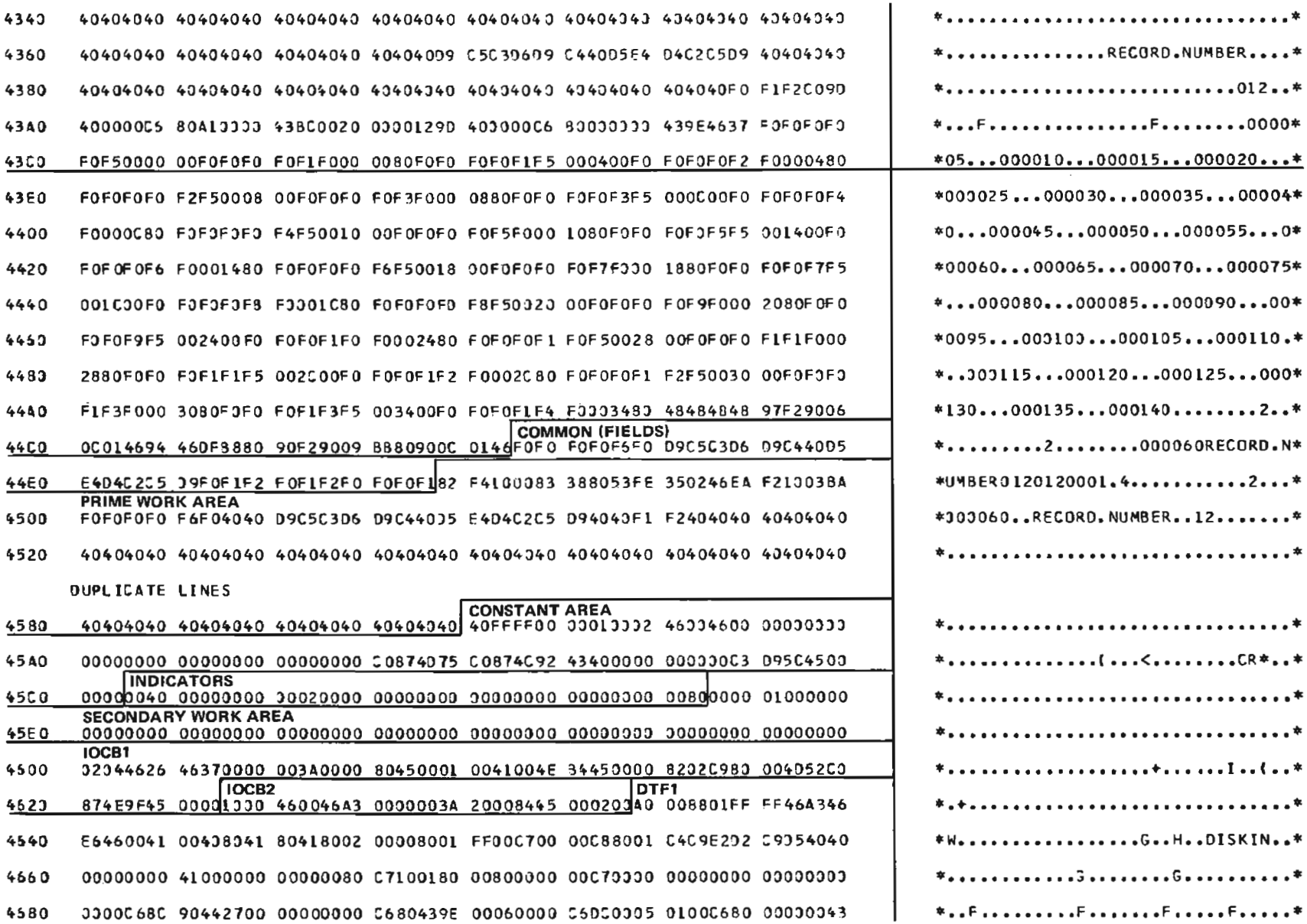
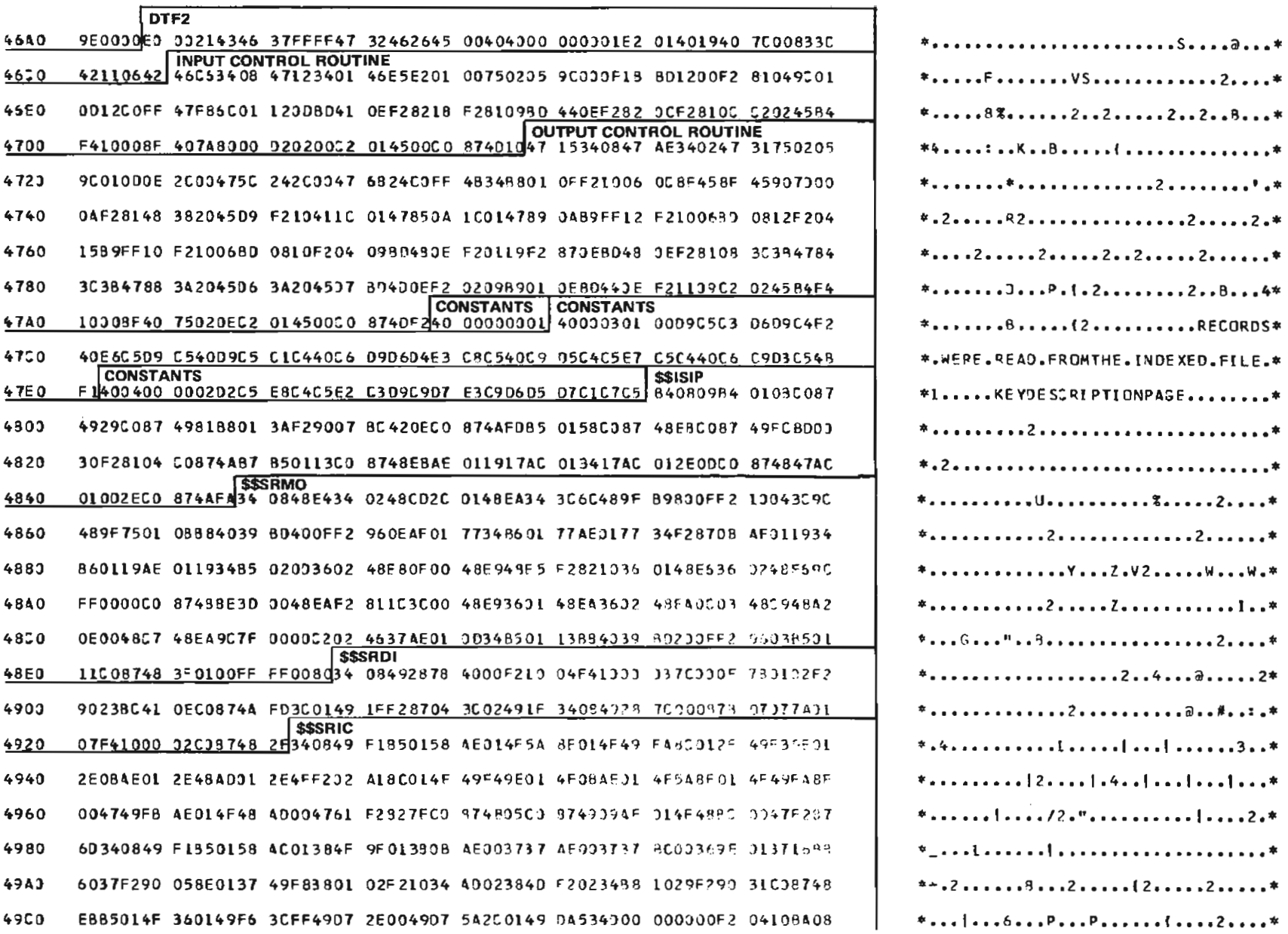


Figure 4-19 (Part 2 of 5). Storage Dump of Sample Program SAMPL2 (Model 15)




```
46A0  9E0000E0 00214346 37FFFF47 32462645 00404000 000001E2 01401940 7C00833C  *.....................S....@...*
46C0  42110642 46C53408 47123401 46E5E201 00750205 9C000F18 BD1200F2 81049C01  *.....F.......VS.............2....*
46E0  0D12C0FF 47F86C01 120D8D41 0EF28218 F2810980 440EF282 0CF2810C C2024584  *.....8%......2..2.....2..2..B...*
4700  F410008F 407A8000 D20200C2 014500C0 874D1047 15340847 AE340247 31750205  *4....:..K..B.....(.............*
4720  9C010D0E 2C00475C 242C0047 6824C0FF 48348801 0FF21006 0C8F458F 45907000  *.......*.............2........'.*
4740  0AF28148 382045D9 F210411C 0147850A 1C014789 0AB9FF12 F2100680 0812F204  *.2.....R2..............2.....2.*
4760  15B9FF10 F210068D 0810F204 098D480E F20119F2 870EBD48 0EF28108 3C394784  *....2.....2.....2..2.....2.....*
4780  3C384788 3A2045D6 3A2045D7 BD4D0EF2 02098901 0E80440E F21109C2 024584F4  *.......J...P.(.2........2..B...4*
47A0  10008F40 75020EC2 014500C0 874DF240 00000001 40000301 0009C5C3 D6D9C4F2  *.......B.....(2..........RECORDS*
47C0  40E6C5D9 C540D9C5 C1C440C6 D9D6D4E3 C8C540C9 D5C4C5E7 C5C440C6 C9D3C548  *.WERE.READ.FROMTHE.INDEXED.FILE.*
47E0  F1400400 0002D2C5 E8C4C5E2 C3D9C9D7 E3C9D6D5 D7C1C7C5 840809B4 0103C087  *1.....KEYDESCRIPTIONPAGE........*
4800  4929C087 49818801 3AF29007 8C420EC0 874AFD85 0158C087 48E8C087 49FCBD00  *.........2.....................*
4820  30F28104 C0874A87 B50113C0 8748E8AE 011917AC 013417AC 012E0DC0 874847AC  *.2.............................*
4840  0100ZEC0 874AFA34 0848E434 0248CD2C 0148EA34 3C6C489F B9800FF2 13043C9C  *.........U.........%.....2....*
4860  489F7501 08884039 BD400FF2 960EAF01 77348601 77AE0177 34F2870B AF011934  *..........2.............2......*
4880  B60119AE 01193485 02003602 48E80F00 48E949E5 F2821036 0148E536 0248E59C  *.............Y...Z.V2.....W...W.*
48A0  FF0000C0 87488E3D 0048EAF2 811C3C00 48E93601 48EA3602 48FA0C03 48C948A2  *..........2.....Z..........I...*
48C0  0E0048C7 48EA9C7F 0000C202 4637AE01 0D348501 13884039 B0200FF2 96038501  *...G...".."B..............2....*
48E0  11C08748 3F0100FF FF008034 08492878 4000F210 04F41000 037C000F 780102F2  *.................2..4...@.....2*
4900  9023BC41 0EC0874A FD3C0149 1FF28704 3C02491F 34084028 7C000978 07077A01  *.............2.........@..#..:.*
4920  07F41000 02C08748 2F340849 F1850158 AE014F5A 8F014F49 FA8C012F 49F34F01  *.4.........I.....|...|......3..*
4940  2E08AE01 2E48AD01 2E4FF202 A18C014F 49F49E01 4F08AE01 4F5A8F01 4F49FA8F  *.........|2....|.4..|...|...|...*
4960  004749FB AE014F48 AD004761 F2927FC0 974B05C0 97403D4F 014F48BC 0047F207  *.....|....!2."..........|....2.*
4980  6D340849 F1850158 AC01384F 9F013B0B AE003737 AF003737 BC003B9F 01371B88  *_...|.....|.....................*
49A0  6037F290 058E0137 49F83801 02F21034 AD02384D F2023488 1029F290 31C08748  *-.2.....8...2.....(2.....2.....*
49C0  E885014F 360149F6 3CFF49D7 2E0049D7 5A2C0149 DA534D00 000000F2 0410BA08  *...|...6...P...P......(....2....*
```

Figure 4-19 (Part 3 of 5). Storage Dump of Sample Program SAMPL2 (Model 15)

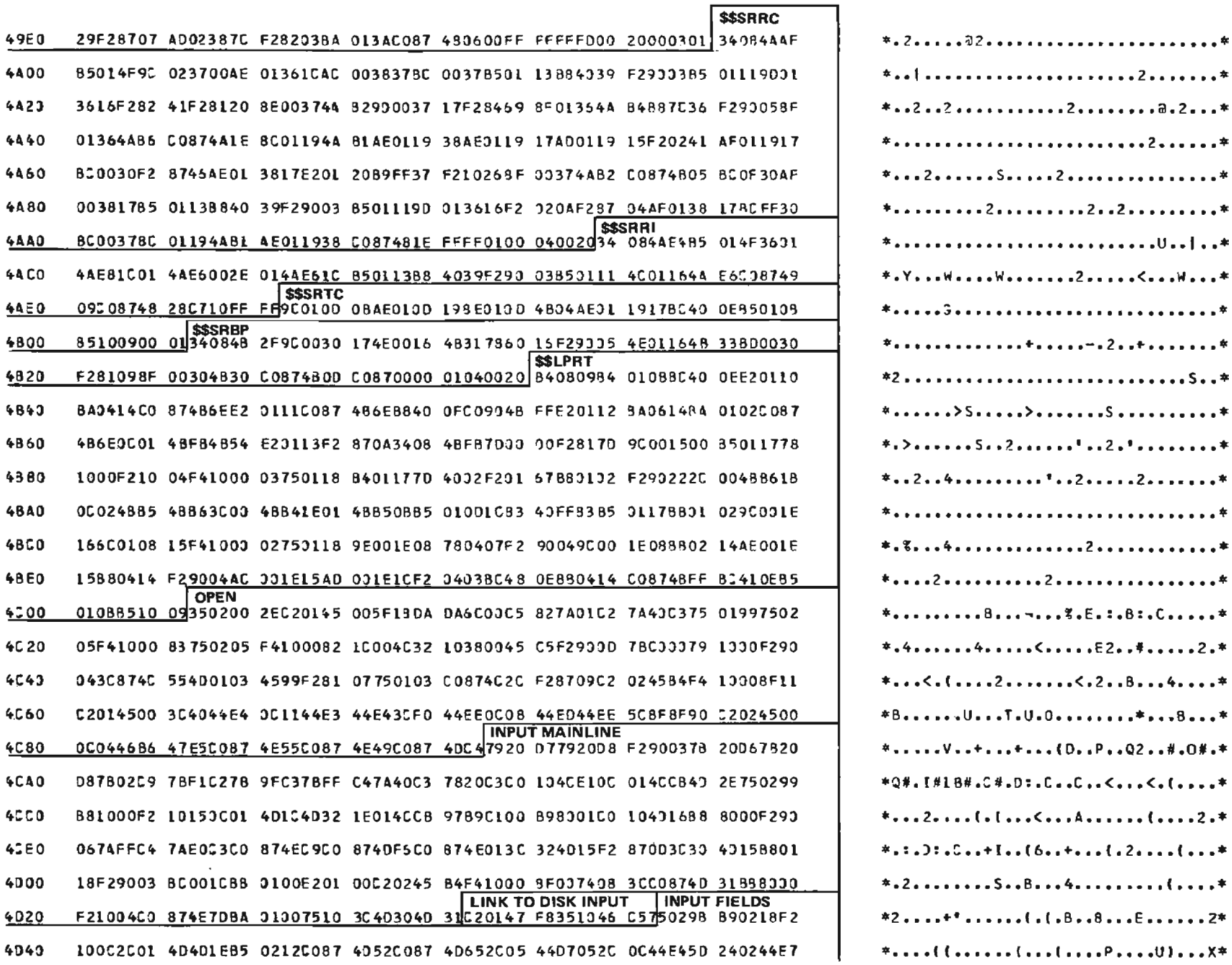


Figure 4-19 (Part 4 of 5). Storage Dump of Sample Program SAMPL2 (Model 15)

Figure 4-19 (Part 5 of 5). Storage Dump of Sample Program SAMPL2 (Model 15)

CHART NUMBERING

Flowcharts are identified in this publication in the following manner:

- A flowchart consisting of a single page is identified by a unique pair of letters.

  Example: AA, AB, AC

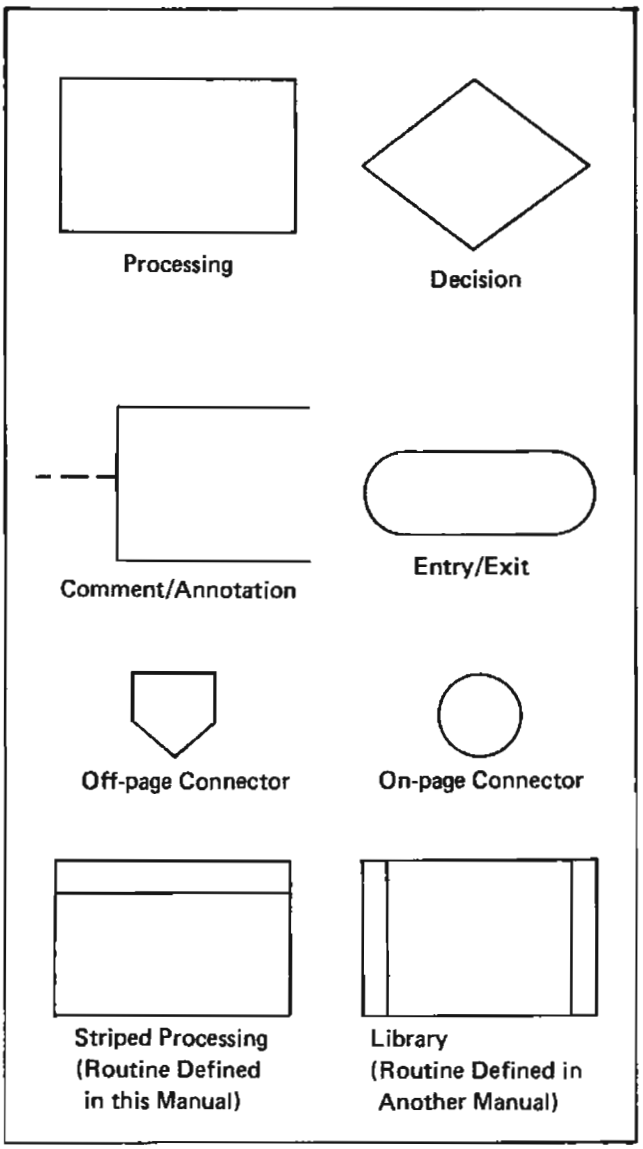
- If a flowchart consists of multiple pages, each page is identified by the same pair of letters, but each page has a unique number.

  Example: First page, CA-01
  Second page, CA-02
  Third page, CA-03, etc.

- Each part has been assigned two sets of flowchart identifying letters. Only after the first set has been completely used, ie., AA-AZ, will the second set be used.

SYMBOLS

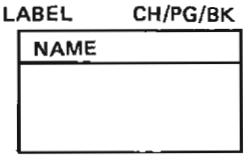
The flowchart symbols used in this PLM are:



Striped Processing Block

The striped processing block indicates entry to a module or routine which is flowcharted and/or described in this logic manual.

Example:

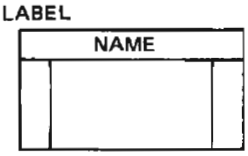


CH/PG/BK indicates the flowchart, page, and block identification where the module or routine is flowcharted. If it is not flowcharted, see the index for the location of the description of that routine.

Library Block

A library block indicates a function or module is documented in another manual.

Example:

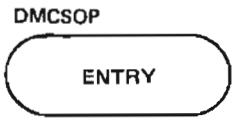


The NAME of the function/module is listed in the *Preface* of this manual under the name of the manual that contains the description of this function/module.

Entry Block

The label in the upper lefthand corner, just above the symbol, is the entry point in the listing for that part of the program.

Example:



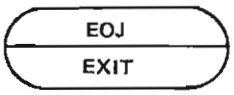
Exit Block

This block indicates that control is leaving this chart.

Example:

If control is being passed to a known function/module, which is documented in another manual, a striped exit block is used. The manual can be found via the Preface as with library blocks.

Example:



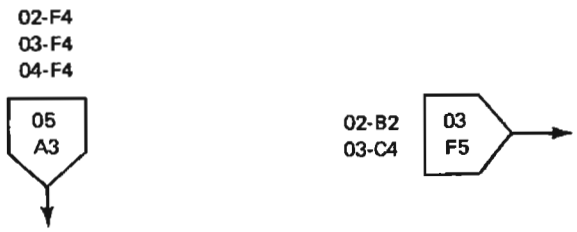
Connectors

Off-page connectors are used to reference between different page of the same chart ID. Off-page connectors leaving a page contain the page number and block number of their destination.

Example:

Off-page connectors contain the page and block number of their origin. If the entry point referenced by the off-page connector is referenced from more than one origin, all origins are given.

Example:



On-page connectors contain the location of a block on the same page. On-page connectors always contain the location of the destination block.

A dump facility is available in this compiler as a debugging aid and for FE program support. It can be used during compilation to dump main storage, disk, and compressions. The dump facility can also be used to print text blocks and to halt on phase load during compilation. If it is not requested, it has no effect on the operation of the compiler. The dump facility makes it possible to submit additional information with APARs, or to supply additional information when requested to do so by FE Field Support (Tech Ops).

The dump facility is requested and controlled by dump control cards. When requested, a routine ($RPZY) is loaded into the high portion of main storage available to the compiler (see *Compiler Control Region*). This routine will intercept all calls to load succeeding RPG compiler modules and all requests to output text. It will also produce the output requested on the dump control cards. This routine requires 1.5K of main storage to execute.

The dump control cards must precede the RPG source program. They will be read, listed, and compressed by the first phase of the compiler ($RPG). Dump control card defaults are made at this time and the assembled addresses, used to specify lower and upper limits (see dump control card format), are converted to the logical main storage address.

The second control routine phase of the RPG compiler ($RPIC), will load $RPZY into main storage if valid dump control cards were found by $RPG. $RPIC passes control to $RPZY to allow for initializing the table of dump functions and the modifying of several address constants in the COMMON area. This enables $RPZY to trap phase calls and text output calls.

Note: Because of the point at which $RPZY is loaded, it is not possible to use this function to obtain dumps before $RPEW is loaded (see *Assign and Diagnostic Phases*).

One function may be specified in each dump control card and up to ten dump control cards may be submitted for each compilation. If more than one function is to be performed between phases, they will be performed in the sequence implied by the order of the cards. The dump control cards must precede the header card for the source program.

Some fields of the dump control card have different meanings depending on the function to be performed. The dump control card format is as follows:

Dump Control Card Format

Columns	*Meaning*	
6	Must contain a 'D' to identify it as a dump control card.	
7	Dump function identification character C – Main storage dump D – Disk dump H – Halt on phase load I – Compression dump T – Print text blocks	
9-10	Start phase ID	Different meanings depending on dump function identification character in column 7. See the specific function for detail.
12-13	End phase ID	
15-18	Lower limit	
20-23	Upper limit	
25	Compression/ file ID	

Column 7 = C (Function = Main storage dump)

- Start phase ID— The first dump will be given after this phase is done and before the next phase is loaded.
- End phase ID — The last dump will be given after this phase is done.
- Lower limit – Assembled lower limit in hexadecimal (from the phase listing). Default is to the start of the partition.
- Upper limit – Assembled upper limit in hexadecimal (from the phase listing). Default is to the end of the partition.

Column 7 = D (Function = Disk Dump)

- Start Phase ID – Same as main storage dump.
- End Phase ID – Same as main storage dump.
- Lower Limit – The relative sector number in decimal of the first sector to be dumped. The default is 0001.
- Upper Limit – The relative sector number in decimal of the last sector to be dumped. The default is 0032.
- Comp/File ID – Specifies the file to be dumped. 'W' is $WORK and 'S' is $SOURCE. The default is $SOURCE.

Column 7 = H (Function = Halt on Phase Load)

- Start Phase – The program halts when the request to load this phase is made.

Column 7 = I (Function = Compression Dump)

- Start Phase ID – Same as main storage dump.
- End Phase ID – Same as main storage dump.
- Lower Limit – Compression number in decimal (statement number from the source listing), of the first compression to be dumped. The default is 0001.
- Upper Limit – Compression number in decimal (statement number from the source listing), of the last compression to be dumped. The default is 9999.
- Comp/File ID – Compression identification character for the compressions wanted.

  F – File Description Compressions
  E – Extension Compressions
  L – Line Counter Compressions
  T – Telecommunication Compressions
  I – Input Compressions
  C – Calculation Compressions
  O – Output Compressions
  D – Data Management Name Compressions
  P – Phase/Dump Compressions
  S – Scratch Compressions
  A – ALTSEQ Compressions
  B – Compile Time Table Compressions
  X – Extra Compressions
  M – Error Compressions

  The default is F if none of the above are specified.

Note: Only those compressions identified by the compression identification character, that fall between the limits specified, will be printed. For example, assume the source listing contains the following: lines 0001 through 0003 are file specifications, lines 0004 and 0005 are input specifications, lines 0006 through 0014 are calculation specifications, and the remainder are output specifications. If the dump control card is coded as shown below, only the input compressions for lines 0004 and 0005 will be printed.

```
Lower Limit = 0001
Upper Limit = 0020
Comp/File = I
```

Column 7 = T (Function = Print Text Blocks)

- Start Phase ID – Start printing the text blocks generated by this phase.
- End Phase ID – Stop printing the text blocks after this phase is complete.

SNAPSHOT OPTION

A HOOK is available 6 bytes before the print buffer for access to PHASE, by PATCH, for dynamic storage area dumps.

The SNAPSHOT option is available during any phase when the following linkage is inserted at the desired area:

```
*SNAPSHOT ENTRY POINT
*
*LINKAGE FORMAT
*         B    @            BRANCH TO SET ARR TO PARM LIST
*         DC   XL1'XX'      PARM FOR TYPE OF DUMP
*         DC   XL2'LLLL'    LOWER CORE OR COMPRESSION UNIT
*         DC   XL2'HHHH'    UPPER CORE OR COMPRESSION UNIT
*RETURN   EQU *
*
*THE FIRST PARM BYTE HAS THE FOLLOWING VALUES AND MEANINGS
*         FF   DUMP CORE
*         81   DUMP FILE $WORK
*         01   DUMP FILE $SOURCE
*         00   'F' COMPRESSIONS
*         02   'E' COMPRESSIONS
*         04   'L' COMPRESSIONS
*         06   DATA MANAGEMENT NAME COMPRESSIONS
*         08   'I' COMPRESSIONS
*         0A   'C' COMPRESSIONS
*         0C   DEBUG COMPRESSIONS
*         0E   SPARE COMPRESSIONS
*         10   'O' COMPRESSIONS
*         12   'T' COMPRESSIONS
*         14   COMPILE TIME TABLE COMPRESSIONS
*         16   ALTSEQ COMPRESSIONS
*         18   ERROR COMPRESSIONS
*         1A   SCRATCH COMPRESSIONS
*
```

@ is the address of the print buffer, (found in COMEXC) minus 6 bytes.

IBM® Technical Newsletter

This Newsletter No. LN21-5423
Date 24 September 1976

Base Publication No. LY21-0501-5
File No. S3-28

Previous Newsletters None

IBM System/3
Disk Systems
RPG II
Logic Manual

This technical newsletter is a part of version 5, modification 00 of IBM System/3 Model 15 RPG II (Program Number 5704-RG1) and also applies to Model 15 RPG II (Program Number 5704-RG2). This technical newsletter provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent versions and modifications unless specifically altered. Pages to be inserted and/or removed are:

Cover, Edition Notice	3-43, 3-44
iii, iv	3-47 through 3-50
2-5 through 2-8	3-50.1, 3-50.2 (added)
3-1, 3-2	3-53 through 3-56
3-13, 3-14	3-56.1, 3-56.2 (added)
3-19, 3-20	B-2.1, B-2.2 (added)
3-23 through 3-26	

Changes to text and illustrations are indicated by a vertical line at the left of the change.

Summary of Amendments

General corrections and clarification.

Information added for Program Number 5704-RG2.

Note: Please file this cover letter at the back of the manual to provide a record of changes.

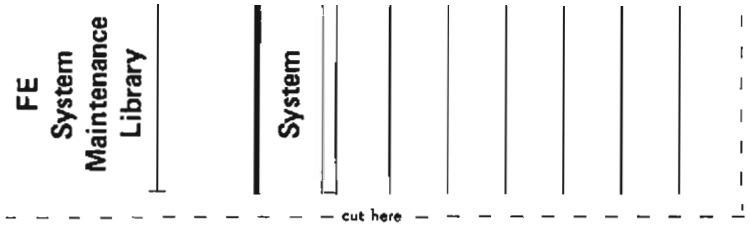
IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901

Printed in U.S.A.

IBM System RPG II Logic (S3-28) Printed in U.S.A. LY21-0501-5

International Business Machines Corporation
General Systems Division
5775D Glenridge Drive N.E.
Atlanta, Georgia 30301
(USA Only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)



SY31-0207-1

IBM System/3 Printed In USA SY31-0207-1

IBM®

International Business Machines Corporation
Field Engineering Division
112 East Post Road, White Plains, N.Y. 10601