



Licensed Material - Property of IBM

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. \$3-28

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.

©Copyright International Business Machines Corporation 1970, 1971, 1972, 1973, 1974, 1975

Page of LY21-0501-5 Issued 24 September 1976 By TNL: LN21-5423

Preface

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

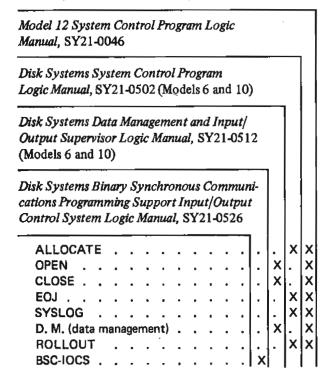
- Components Reference Manual, GA21-9236
- Halt Guide, GC21-5145
- RPG II Reference Manual, SC21-7504

Model 15

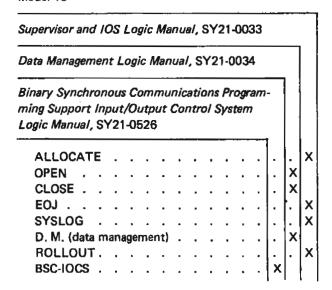
- Components Reference Manual, GA21-9236
 - System Messages, GC21-5076
 - RPG II Reference Manual, SC21-7504

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.



Model 15



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:

- 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.

Contents

SECTION 1. INTRODUCTION 1-1	Segment List
	Symbol Table
	Telecommunications Table
Compiler Operation	Text - RLD Record
Compiler Control Region 1-3	Disk Work Areas
Linkage Between Phases 1-3	
	SECTION 4. OBJECT PROGRAM 4-1
SECTION 2. PROGRAM LOGIC 2-1	
	Flowchart Techniques 4-1
Compiler Phase Descriptions 2-1	Overall Object Program Flow
Input and Compression Phases 2-1	Detailed Object Program Flow
Assign and Diagnostic Phases	Open Mainline (Chart CA)
Assemble Phases 2-1	Input Processing Control (Chart CB) 4-1
Pre-Assemble Phases	Output Processing Control (Chart CC) 4-1
Assemble II Phases	Output Fields and Records Code (Chart CD) 4-1
Overlay Phases	Input Mainline (Chart CE)
Dump Control Phase 2-1	Fetch Overflow (Chart CF)
Interphase Control Routines Description 2-22	Record ID (Chart CG) 4-6
and product to the state of the	Multifile and Matching Records Logic (Chart CH) 4-6
SECTION 3. DATA AREAS	Control Fields Logic and Move (Chart CI) 4-6
SECTION J. DATA AREAS	Chain and Read (Chart CJ)
Compiler Control Region	LR and Overflow Control Mainline (Chart CK) 4-6
Common	Move Input Fields Mainline (Chart CL) 4-6
Parameters to RPG II Halt Processor	Program Close Mainline (Chart CM) 4-7
	Calculations Object Code
IOB	
I/O Parameters	Calculations Specification Descriptions
Control Routine Save Area	Library of Subroutines
Compression Block Table (CZATAB)	Data Areas
Compression Work Area	Reserved Object Communications Area (ROCA) 4-73
Compression Formats	Trailer Table
Control Statement Compressions	Define the Table (DTT)
File Description Compressions	Define the File (DTF)
Extension Compressions	Alternating Collating Sequence and Translate Tables 4-76
Line Counter Compressions	Match Field Save Areas
Input Compressions	Control Field Save Area
Dump Control Compressions 3-18	Constants, Edit Words, and Edit Codes 4-76
Calculation Compressions	Error Recovery Procedure (ERP) Area 4-77
Output Format Compressions	Input and Output Buffers 4-77
Telecommunications Compressions	Completion Codes from Data Management 4-77
Alternate Collating Sequence, File Translate and Compile-	Input/Output Control Block (IOCB) 4-78
Time Table/Array Compressions	Overlays (Models 6, 10, and 12) 4-79
Chain Table	Overlay Concept
Compile-Time Symbol Table	Segments
Data Management Entry Points and Module Names	Overlay Priority
Compressions	Suboverlays
Error File	Overlay Technique
File Input/Output Table	Overlay Editor
Filename Table	Overlay Fetch Routine
Final Segment List (Models 6, 10, and 12) 3-57	Overlay Fetch Table
Final Segment List (Model 15)	How To Find an Overlay 4-84
General Storage Table	Overlays (Model 15)
Internal Symbol Table (Models 6, 10, and 12)	Overlay Category
Name Table	Dump Analysis
Object Code Block	
Phase Load Compression	
1 mov 2014 Comptosion	

APPENDIX A. FL	WO.	CH	AR	TIN	NG '	TEC	'HI	ИQ	UE	S	•	•	٠	A-l
Chart Numbering														A-1
Symbols														
Striped Process														
Library Block														A-2
Entry Block .														A-2
Exit Block .														A-2
Connectors .														
APPENDIX B. DU	MP	FA	CI	LIT	Y									B-1
Dump Control	Car	d F	orm	at										B-1
INDEX														X-1

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.

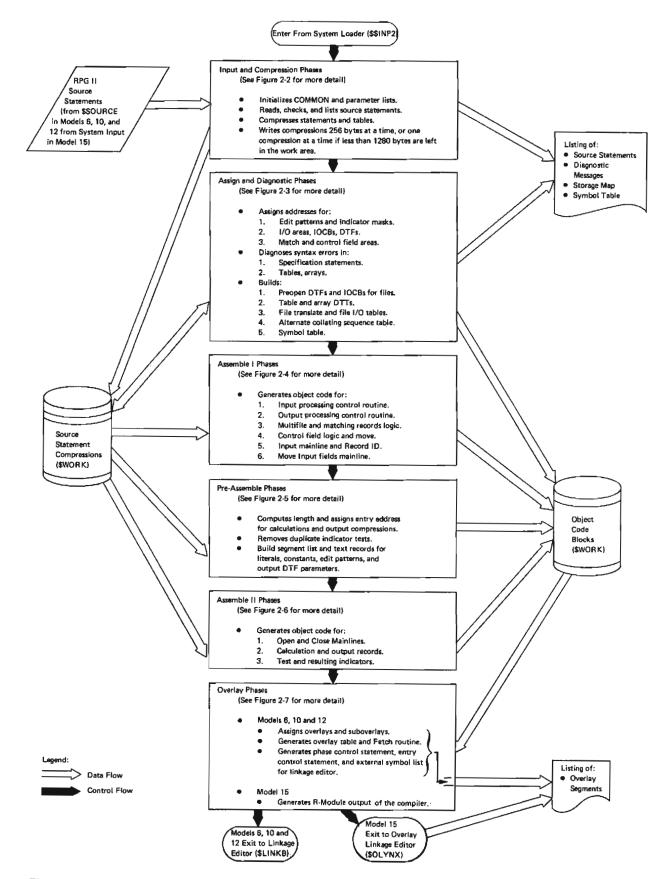
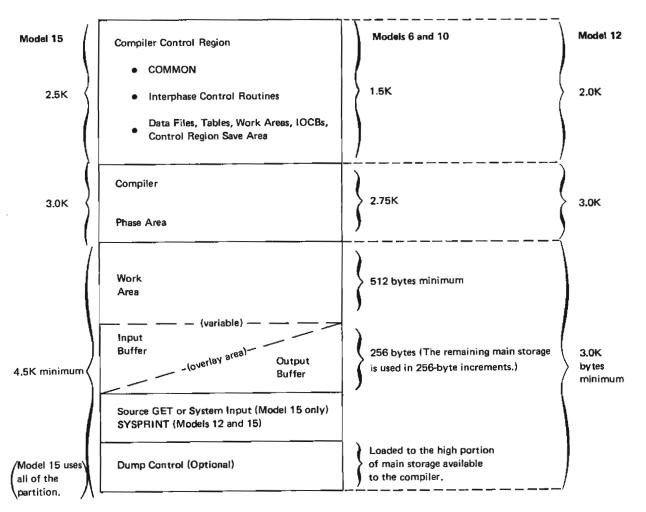


Figure 1-1. RPG II Compiler Overview

I

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.)

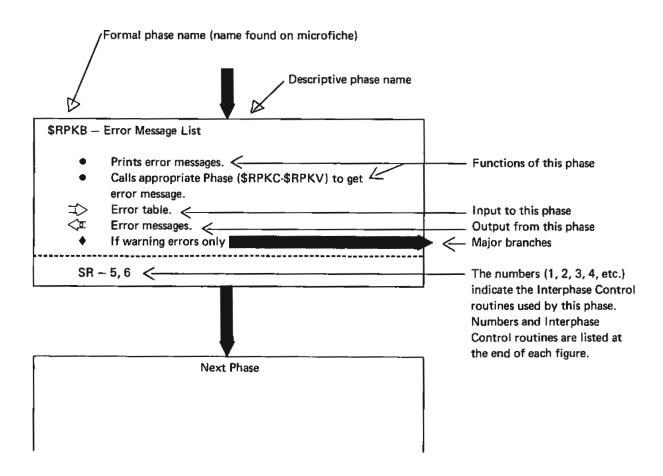


Figure 2-1. Explanation of Program Organization Charts

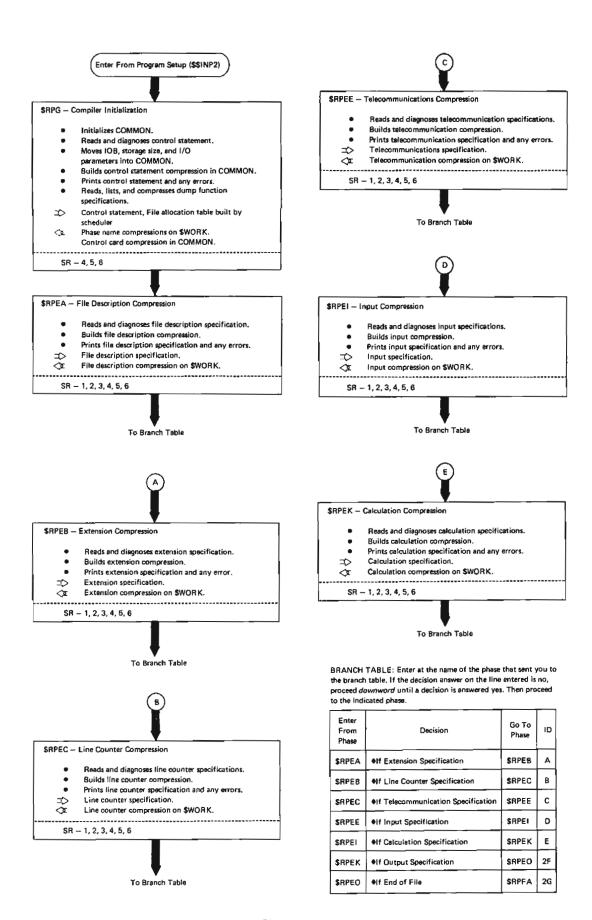


Figure 2-2 (Part 1 of 2). Input and Compression Phases

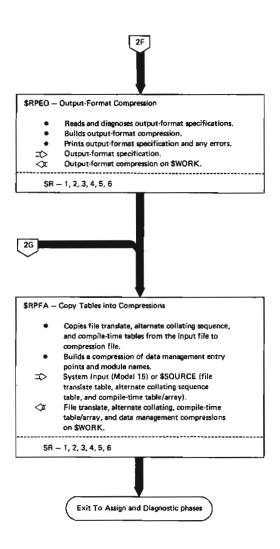


Figure 2-2 (Part 2 of 2). Input and Compressions Phases

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.

Page of LY21-0501-5 Issued 24 September 1976 By TNL: LN21-5423

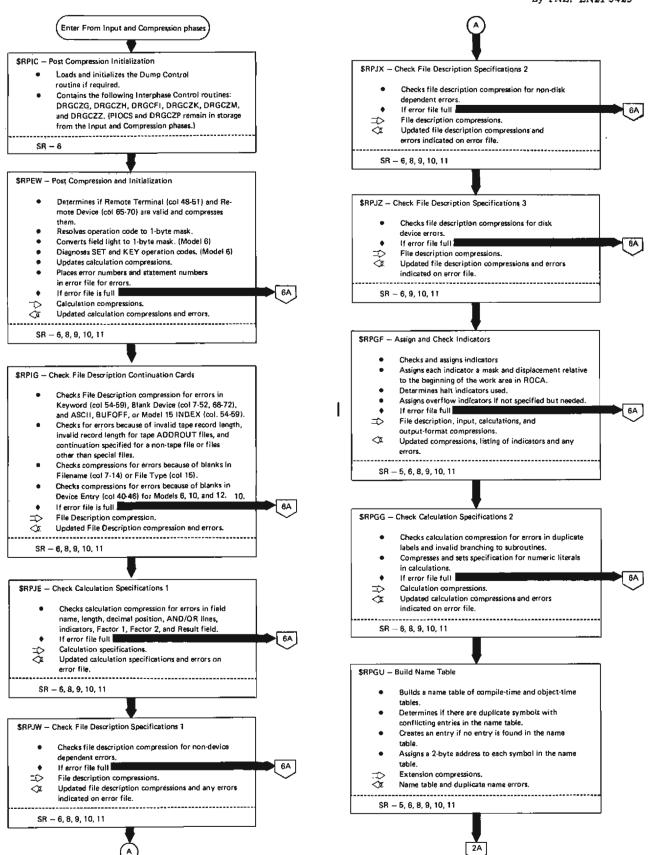
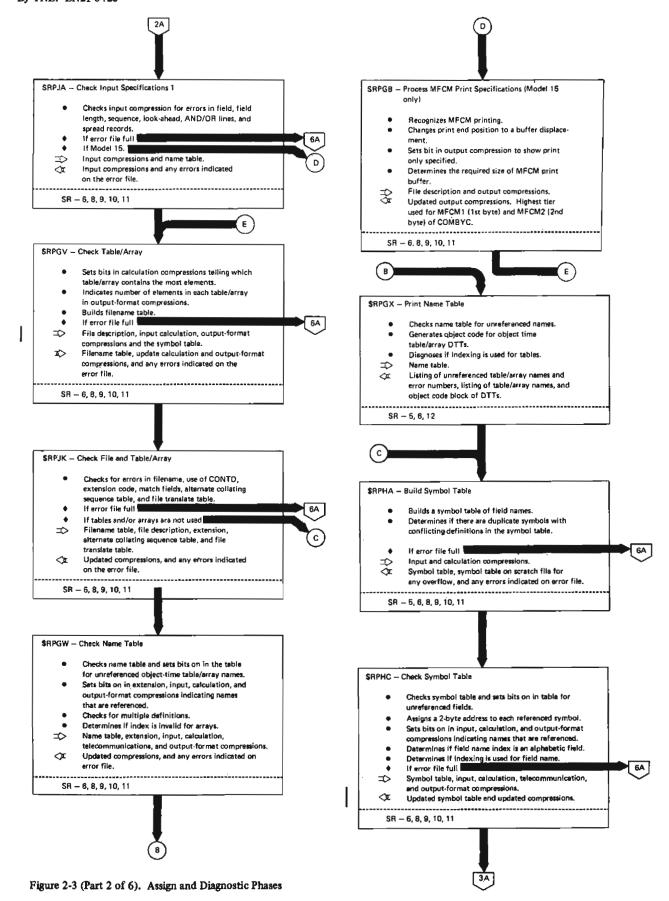


Figure 2-3 (Part 1 of 6). Assign and Diagnostic Phases



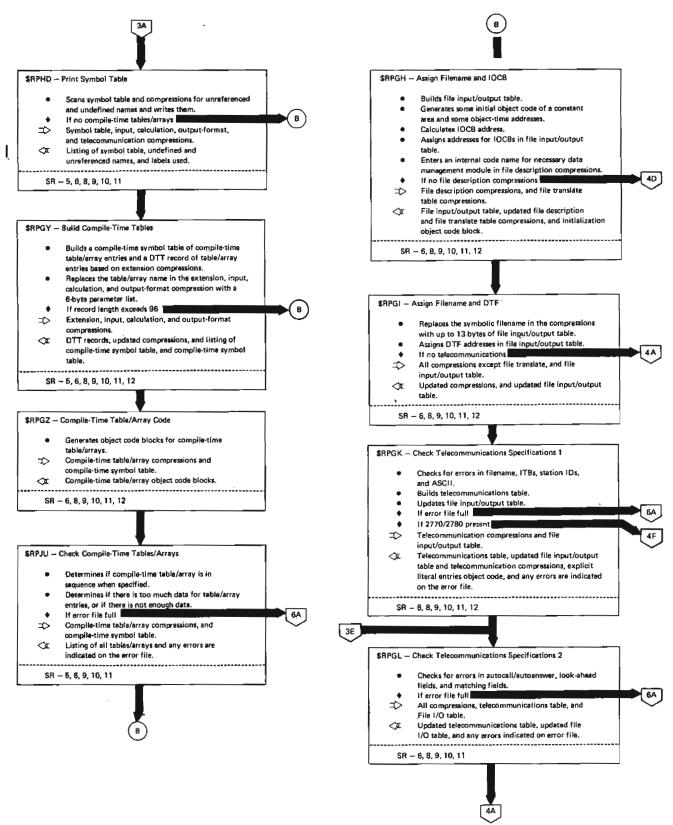


Figure 2-3 (Part 3 of 6). Assign and Diagnostic Phases

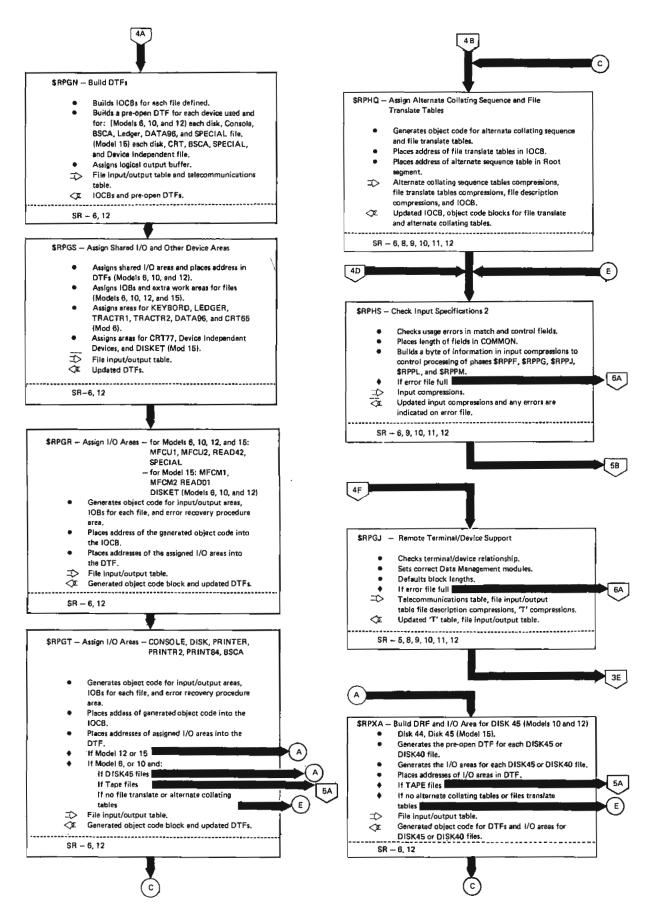


Figure 2-3 (Part 4 of 6). Assign and Diagnostic Phases

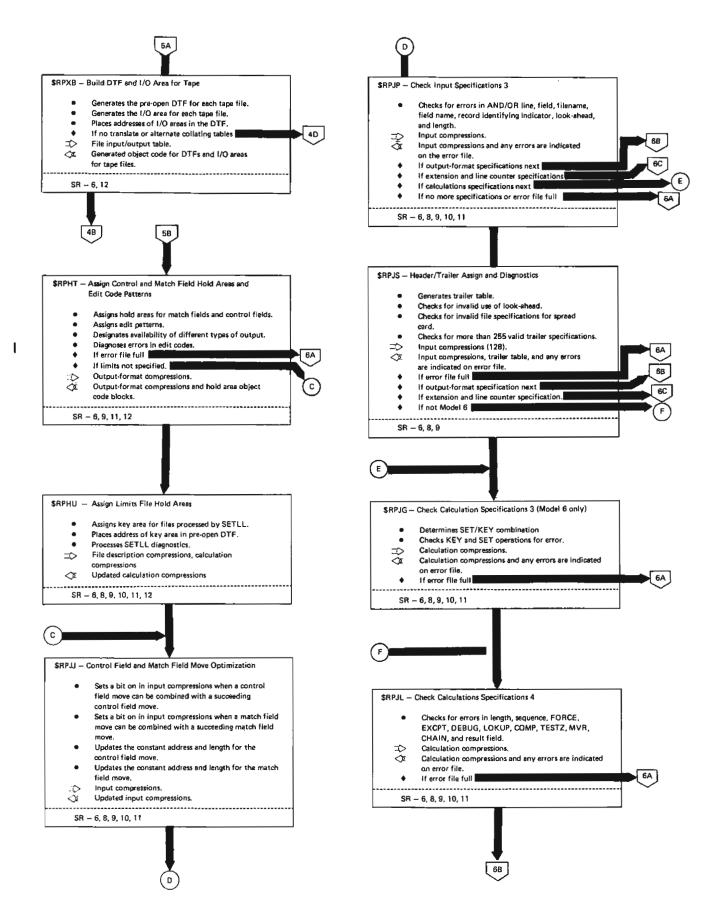
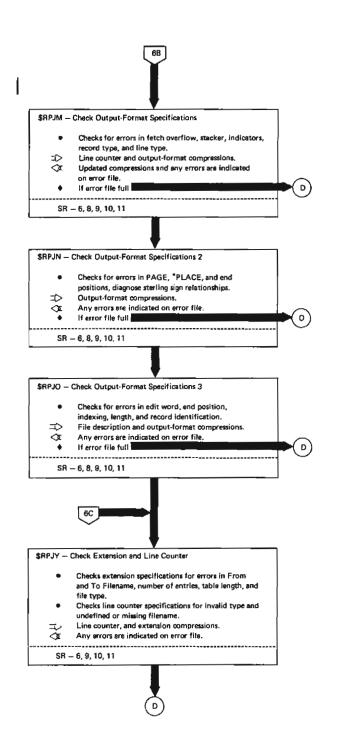
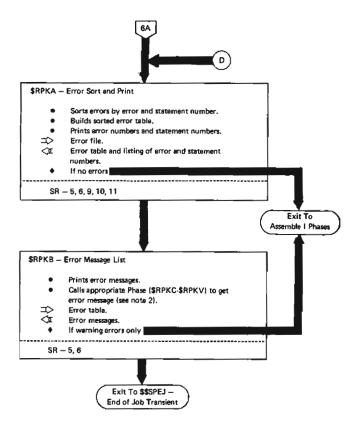


Figure 2-3 (Part 5 of 6). Assign and Diagnostic Phases





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

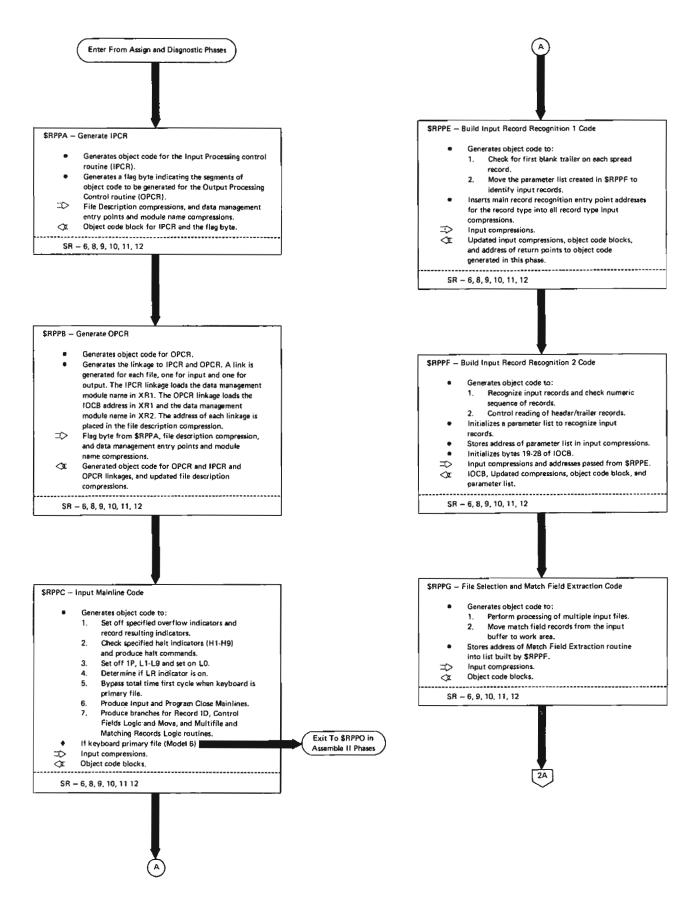


Figure 2-4. Assemble I Phases (Part 1 of 2)

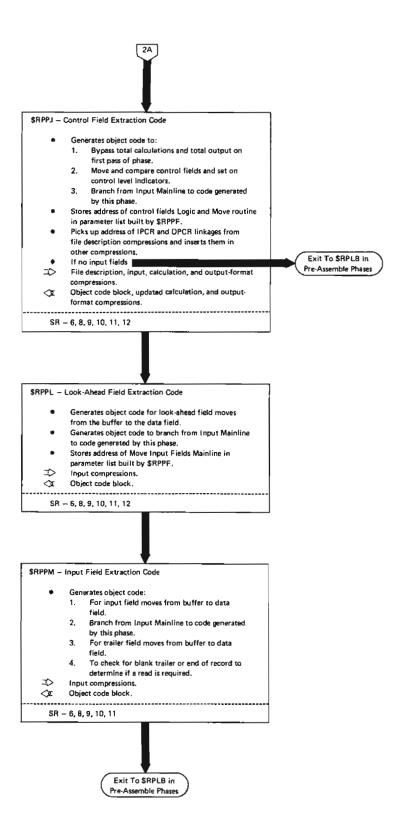


Figure 2-4. Assemble I Phases (Part 2 of 2)

Note: Interphase Control routines that can be used during the 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 - 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.

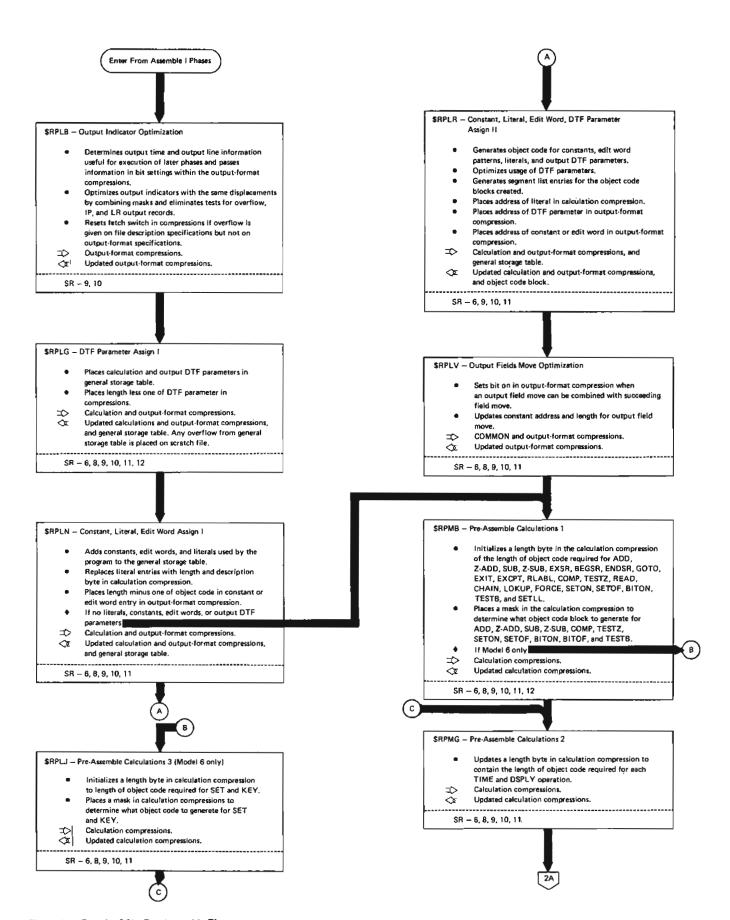
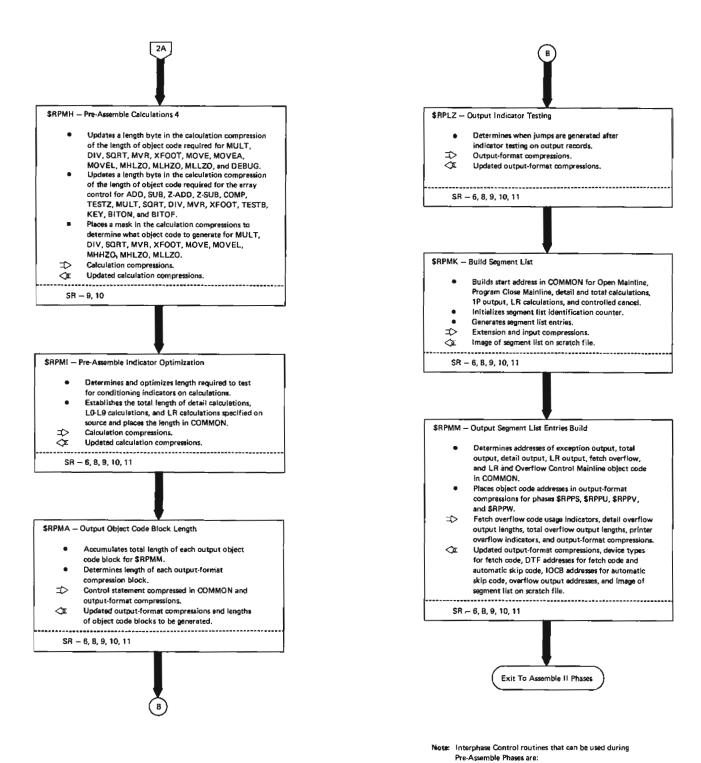


Figure 2-5 (Part 1 of 2). Pre-Assemble Phases



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,

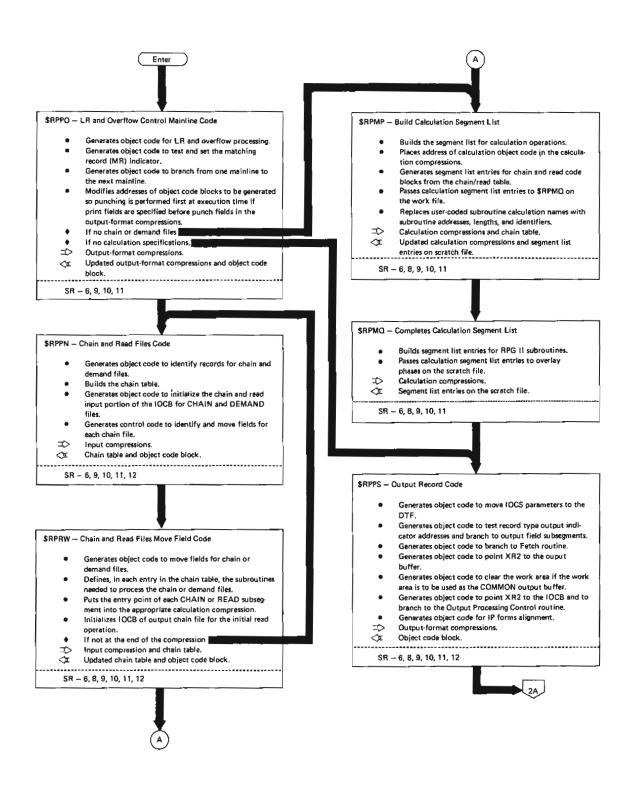


Figure 2-6. Assemble II Phases (Part 1 of 4)

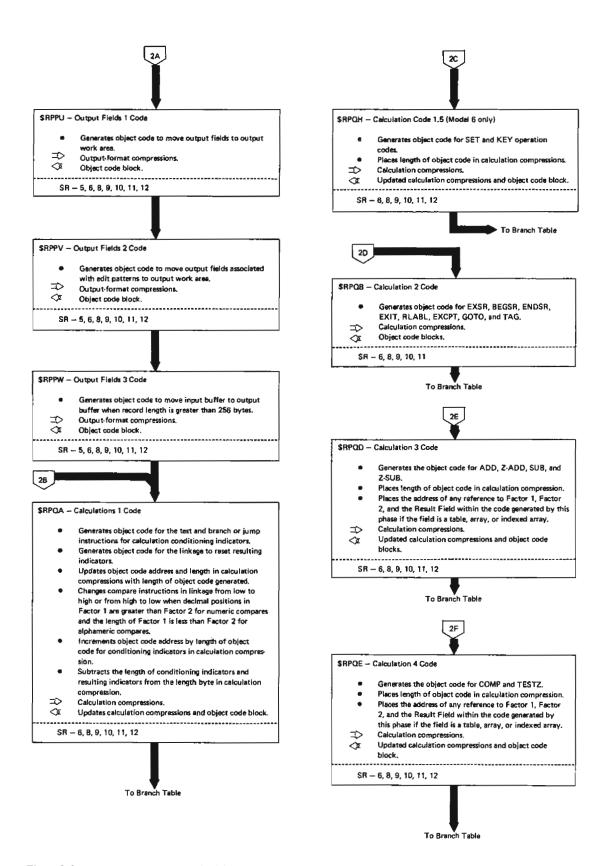


Figure 2-6. Assemble II Phases (Part 2 of 4)

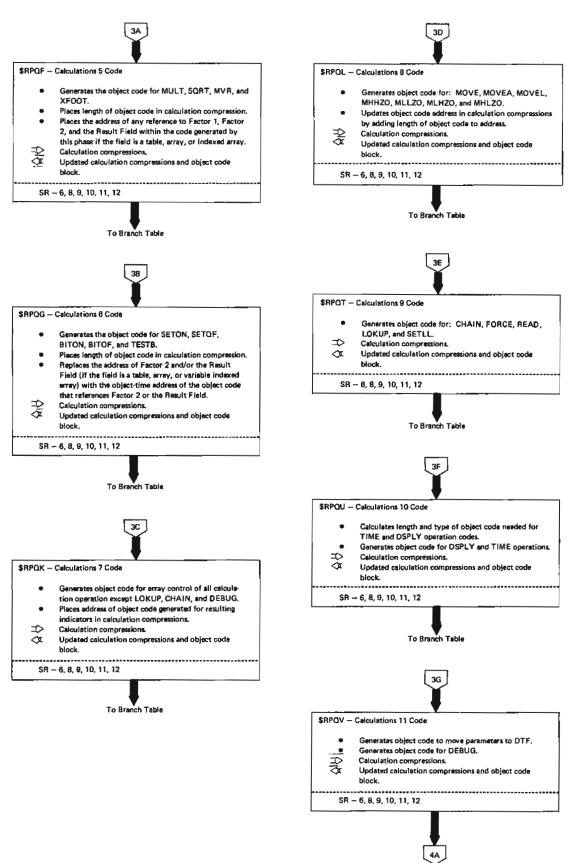


Figure 2-6. Assemble II Phases (Part 3 of 4)

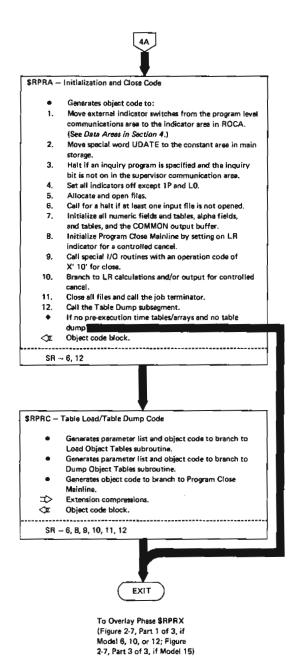


Figure 2-6. Assemble II Phases (Part 4 of 4)

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 downword 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 opera- tion code	\$RPQ8	2D
\$RPQB	 If ADD, Z-ADD, SUB, Z-SUB operation code 	\$RPQD	2E
\$RPQD	♦ If COMP or TESTZ operation code	SRPQE	2F
\$RPQE	♦ If MULT, SQRT, MVR, or XFOOT operation code	\$RPQF	ЗА
\$RPQF	♦ If SETON, SETOF, BITON, BITOF, or TEST8 operation code	\$RPQG	38
\$RPQG	If LOKUP, CHAIN, or DEBUG opera- tion 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	3 E
\$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.

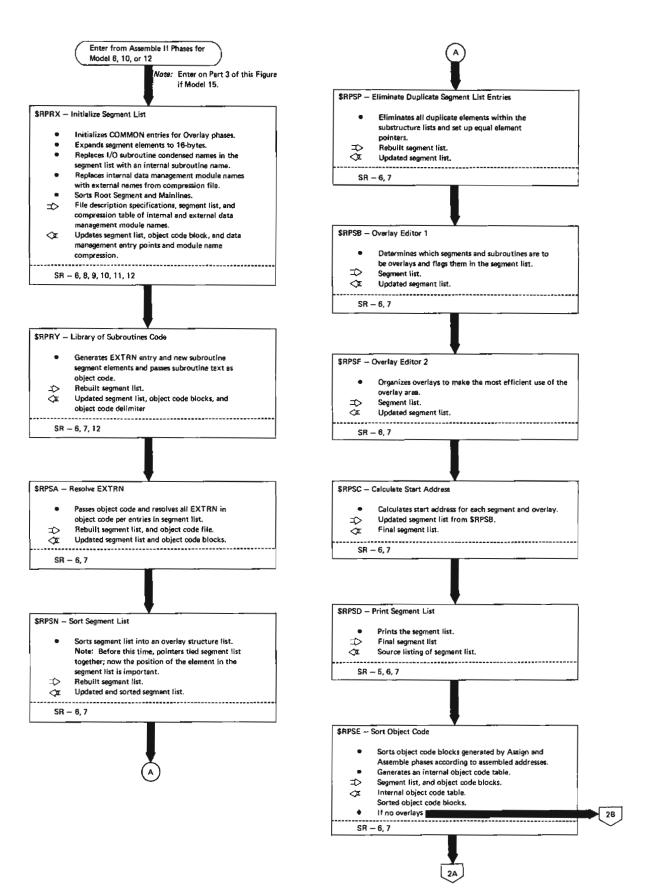


Figure 2-7 (Part 1 of 3). Overlay Phases

I

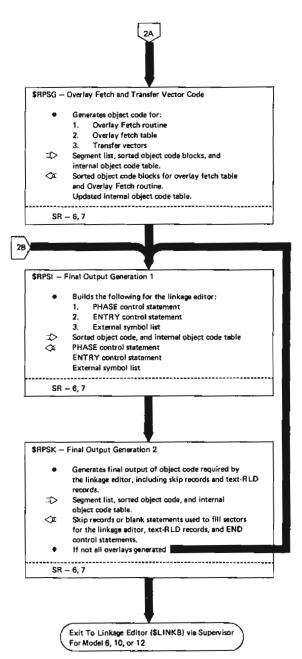


Figure 2-7 (Part 2 of 3). Overlay Phases

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.

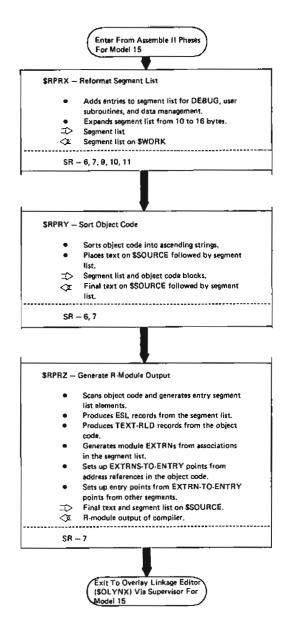


Figure 2-7 (Part 3 of 3). Overlay Phases

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 256byte buffer area in the control region of storage instead of normal object code area.

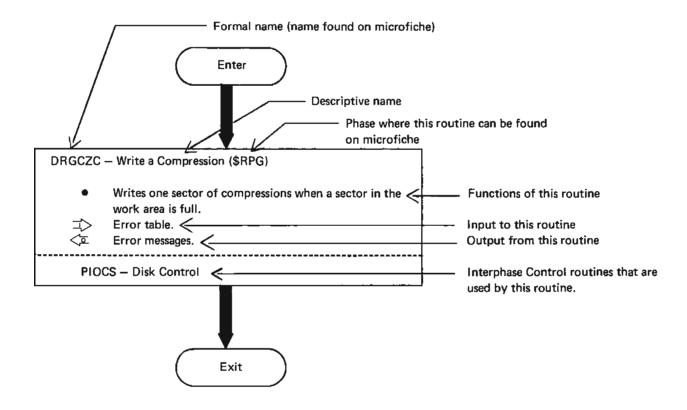


Figure 2-8. Description of the Format Used in Describing the Interphase Control Routines

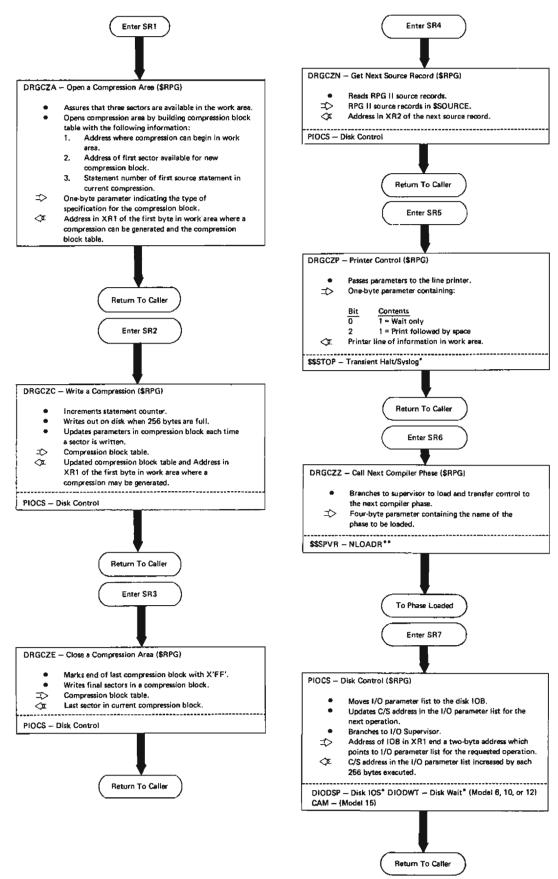


Figure 2-9 (Part 1 of 2). Interphase Control Routines

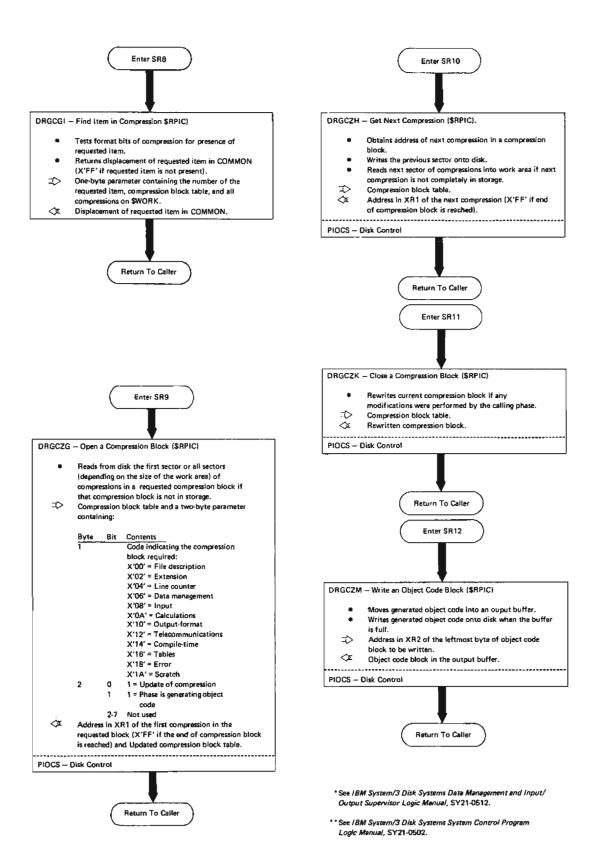


Figure 2-9. Interphase Control Routines (Part 2 of 2)

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:

- COMMON
- 2. Interphase Control routines
- 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'OA' 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.

				СОММО	N	
Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explana	ation
COMAAC	00	0	\$RPG	\$RPGF \$RPKA	Control	card specification information:
				\$RPKE	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 Bit 3	1 = No object program produced (col 11) 1 = No list
					Bit 4	1 = Shared I/O (Models 6, 10, and 12 on
					Bit 5	1 = DEBUG operation (col 15)
					Bit 6-7	
						01 = United Kingdom format (col 21) 10 = World Trade date format (l) col 21
						11 = World Trade date format (J) col 21
COMABC	01	1	\$RPG	None		statement information; object program I 12-14)
COMACC	02	2	\$RPG	\$RPGH \$BPGI \$RPGJ \$RPGN \$RPGS	Bit 7	1 = Program exceeds 64K bytes of storage or disk overflow error
				\$RPGT \$RPXA \$RPXB		
COMADC	03	3	\$RPG	\$RPHC	Control	statement information:
				\$RPJM	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 Bit 5	1 = Repeat 1P lines (col 41) 1 = File translation (col 43)
					Bit 6	1 = MFCU or DATA96 zero suppression (col 44)

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	Error information for assign and diagnostic phases		
				\$RPGX \$RPHA \$RPHD \$RPHT \$RPGU	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		
СОМВСС	0D	12-13	\$RPPC	None	Bit 6 1 = Too many tables/arrays 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		
сомскс	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

				соммог	N
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	18	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	\$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
сомвос	1D	28-29	\$RPHT	None	Address of decimal point for edit pattern 1
СОМВОС	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 con-
COMDZC	1E	29-30	\$RPSG	None	dition opening of a file 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

				COMMO	N
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	Field information:
				\$RPGI	Bit 0 1 = Halt indicator H1 used
				\$RPHS	Bit 1 1 = Direct file without ADDROUT
				\$RPPC	Bit 2 1 = Limits file specification
				\$RPPJ	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					
COMGEC	28	39-40	\$RPSC	None	Length of available object storage					
СОМВНС	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.					
сомстс	29	40-41	\$RPMI	\$RPMK	Length of total calculations					
сомстс	29	40-41	\$RPMK	None	Address of total calculations					
COMGFC	2A	41-42	\$RPSC	None	Address of the overlay area					
СОМВКС	2B	42-43	\$RPGI	None	Address of the FROM file DTF					
COMFAC	2 B	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

				COMMO	N		
	Hexadecimal	_					
Name	Displacement		Defined	Modified	Explana	tion	
COMBMC	2F	46-47	\$RPJX	\$RPGH	D 0	A4	
				\$RPGI	Byte 0	Not us	
					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
						Die E	(Models 12 and 15)
						Bit 5	1 = MFCU2 print buffer needed (Models 12 and 15)
COMGIC	30	47-48	\$RPSB	\$RPSC	Length	of Ope	n-Close suboverlay
COMCFC	30	48	\$RPPL	None	Tempor	ary sav	e area for COMBWC
COMCQC	31	48-49	\$RPMM	None	Address	of firs	t fetch code block
COMCGC	32	49-50	\$RPPL	None	Address	of retu	urn to main code for move fields
COMGJC	32	49-50	\$RPSB	None	Address	of \$W	ORK
сомсіс	33	50-51	\$RPPO	None	Address	of last	record in LR Output routine
COMAQC	36	52-54	\$RPHS	\$RPHT	Bytes 0-		Address of the high-order byte of
	•				Byte 2		the control field save area.
					Dyte 2		Length of the save area.
COMEYC	37	54-55	\$RPLR	\$RPQU	Address	of DS	PLY constant
COMARC	3F	55-63	\$RPHS	\$RPPJ	Length (of each	control field:
					-		th of L1 field
					Byte 1	= Lengt	th of L2 field
					Byte 2	= Lengt	th of L3 field
					Byte 3	= Lengt	th of L4 field
					Byte 4	= Lengt	th of L5 field
					Byte 5	= Lengt	th of L6 field
					Byte 6	= Lengt	th of L7 field
					Byte 7	= Lengt	th of L8 field
					Byte 8	= Lengt	th of L9 field
СОМСМС	3B	56-57	\$RPMA	\$RPMK	Length	of 1P C	Output routine
СОМСМС	39	56-57	\$RPMK	None	Address	of 1 P (Output routine
COMBDC	3B	58-59	\$RPPL	None			of the operand portion in the elds to detail calculations
СОМСРС	3D	60-61	\$RPMA	\$RPMM	Length o	f Exce	ption Output routine
СОМСРС	3D	60-61	\$RPMM	None	Address	of Exce	eption Output routine

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

Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanat	tion
COMGPC	3E	61-62	\$RPSA	\$RPRY	Largest it	tem number in segment list
COMGPC	3E	61-62	\$RPRY	\$RPSA	Byte 1, B	Bit 0 1 = Entry point not found in segment list
СОМВРС	3F	63	\$RPGU	\$RPGW	Table and	d array information:
COMDI C	31	00	4 111 33	\$RPHA	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 head ing printed
сомвас	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 1 = OD
					Bit 4 Bit 5	1 = OD 1 = OC
					Bit 6	1 = OB
					Bit 7	1 = OA
COMAPC	40	64	\$RPMM	None	Overflow	segment needed
COMAOC	41	65	\$RPGU	\$RPGI	File info	rmation:
				\$RPPC	Bit 0	1 = ADDROUT file program specified
				\$RPMA	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

				COMMOI	
Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanation
COMANC	43	66-67	\$RPGN	\$RPGU	Address of next available byte in storage
				\$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	
				\$KPHU	
COMAMC	45	68-69	\$RPGY	\$RPGU	Address of end of Root Segment
				\$RPHA	
				\$RPHD	
				\$RPGH	
				\$RPGI	
				\$RPGK	
				\$RPGS	
				\$RPHQ	
				\$RPHT	
				\$RPJS	
				\$RPPA	
				\$RPGR	
				\$RPHU	
сомсхс	45	68-69	\$RPMK	\$RPHA	Address of normal close entry in Program Close
					Mainline

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

				соммо	N
Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanation
COMALC	47	70-71	\$RPIC	\$RPQA \$RPHS \$RPJA	Displacement of an item in a compression
СОМАНС	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
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	6 5	85	\$RPEA	\$RPEB \$RPEC \$RPEE \$RPEI \$RPEK \$RPEO	Save area for specification type being processed

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

	COMMON						
	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	\$RP\$G	Address of beginning of the sort strings of object code		
сомнис	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

				COMMO	\
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	\$RPGN \$RPPB \$RPGH \$RPGI \$RPHS \$RPJM	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	\$RPGK \$RPGL \$RPHS \$RPJW \$RPJZ \$RPEW \$RPHT	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 lise) Bit 6-7 01 = Model 12 10 = Model 15 11 = External buffers specified (Program Number 5704-RG2)

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

				COMMO	V	
Name	Hexadecimal Displacement	Bytes	Defined	Modified	Explanat	tion
COMELC	DMELC 71 113 \$RPJE None		specifica Bit 0	1 = ADD, SUB, Z-SUB, Z-ADD		
					Bit 1 Bit 2	1 = MULT, DIV, MVR, XFOOT, SQRT 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 Bit 7	1 = DEBUG 1 = GOTO, TAG, EXIT, EXCPT, BEGSR
						ENDSR, EXSR, RLABL
COMEMC	72	114	\$RPG	\$RPMH \$RPJX	Operatio specifica	n code information specified on calculation
				THE STATE OF THE S	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 Bit 7	1 = Halt for warning diagnostics 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 o	of Output Processing Control routine
COMEVC	74	115-116	\$RPG	None	Gold Ke	y 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	Annlica	tion program ID in binary:
COMETTO			••••		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	File info	ormation:
				\$RPGI	Bit 0	1 = Exception output
				\$RPHT	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

				соммог	V
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	7 D	124-125	\$RPG	\$RPGX	Address of first byte of print buffer
COMDTC	7 F	126-127	\$RPGX	\$RPGY \$RPGZ	Address of end of work area
сомссс	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 for all devices are given in IBM System/3 Disk Sys

I/O Parameters

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:

IOB		Byte	Contents
for all devices ar	tions of the input/output blocks (IOBs) e given in IBM System/3 Disk Systems nt and Input/Output Supervisor Logic	0-1	Address of current entry in compression block table
Manual, SY21-0	- · ·	2-3	Address of next sector available for compressions
I/O Parameters		4-5	Current compression name
into the disk IO	a contains parameters which are moved B by Interphase Control routine PIOCS. e parameters when this 18-byte area is	6-7	Address of first byte of current compression block
called by one of	the other Interphase Control routines for tput. The I/O parameters are located at	8-9	Address of the work area
	C'. These parameters are:	10-11	Address of the last byte in the work area
Byte	Contents	12-13	End of the current compression block in the work area
0-5	Parameters for a read operation	14-19	Three two-byte work areas
6-11	Parameters for rewriting a modified compression	20-21	Pointer to compression in main storage
12-17	Parameters for writing an object code block	22-23	Start address of the output buffer
	olock	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

Byte	Contents
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	Descript	ion		
128	1	\$RPEA	Langth o	of compression in binary		
129 130	1	\$RPEA	Indicates Item 129	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	Item 130	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
131	1		Item 131	Bit 7 1 = Item number 139 Not used		Bit 7 1 = Item number 147
132	1	\$RPEA	Bit 0 Bit 1 Bit 2 Bit 3	1 = Invalid compression 1 = Invalid filename (col 7-14) and compressed as X'0' 1 = Special IOS routine name invalid (col 54-59) and not compressed 1 = Filename (col 7-14) missing	Bit 5 1 : Bit 6 1 :	 Input or output table file BSCA printer type device ASCII continuation specified for this file Fixed length records Variable length records
		\$RPGV	8it 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 Bits 2-3	00 = Entry not specified (col 18) 01 = Descending sequence 10 = Ascending sequence 11 = Invalid sequence 00 = Entry not specified (col 17) 10 = End of file 11 = Invelid entry	Bits 4-5 Bits 6-7	00 = Entry not specified (col 66) 01 = Unordered sequence 10 = Invalid entry (assume A) 00 = Sequential file processed consecutively (col 28) (blank) 01 = Random 10 = Indexed file processed sequentially within limits 11 = Invalid entry
		\$RPJK	If no ant	ry is specified, bit 0 and 1 are set to 00 (not sp	ecified) or 1	0 (ascending sequence) by default
135	1	\$RPEA	Bits 0-1 Bits 2-3	00 = Entry not specified (col 39) 01 = Line counter specifications 10 = Extension specifications 11 = Invalid entry 00 = Sequential or direct file (col 31)	Bit 4 Bits 5-7	Reserved 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	Descripti	on			
136	12	\$RPEA	Filename	(col 7-14)		
			Byte 11	(Models 10	and 15) = V or D (file form	at for vari	able length tape files)
		\$RPIG	Byte 11	Models 10	and 15) = X'00'		
		\$RPGH	Bytes 0-7	,	Filename		
		ψiii Git	Dy 163 0-7		i lighallig		
			Bytes 8-9	Address of	of IOCB		
			Byte 10	Bits 0-3		Bits 4-7	
				1000 =	Consecutive processing	0001 =	Input file
				1001 =	Indexed file processed		Output file
					randomly by key	0011 =	Update/com-
				1010 =	Direct (disk address)		bined file
					file		Add/print file
				1011 =	Direct (record number)		Input + add file
				1100 -	file Indexed file	UIIU =	Output + add file
					Indexed file processed	0111 =	Update/combined
					sequentially within		+ add file
					limits	1000 =	Output unordered
							file
						1010 =	Output chain file
			Byte 11	Model 6		Model 10	
				X'10' =	CONSOLE (printer-	X-10' =	CONSOLE (Printer-
				VII 61	keyboard)		keyboard)
					KEYBORD (keyboard)		DISKET
					DISKET Special IOS routine	X-20. =	READ42 (1442 Card Read-Punch)
				X,80, =	•	Y'66' -	Special IOS routine
				_	BSCA with first-time	X'60' =	
				7,01	logic	X'80' =	
				X'90' =	CRT (display station)		BSCA with first-
					DISK (disk unit)		time logic
				X'A1' =	Multivolume Disk	X'A0' =	DISK (disk unit) A1
				X'E1' =	TRACTR1 (tractor 1)		Multivolume disk
				X'E2' =	TRACTR2 (tractor 2)	X'A1' =	Multivolume disk
				X'E9' =	LEDGER (ledger card	X'C0' =	DISK45
					device)	X,E0, =	PRINTER (line print-
					DATA96 (data recorder)		er — carriage 1)
				X'FF' =	Invalid entry	X,E8, =	PRINTR2 (line print-
				Model 12		VIEN	er – carriage 2)
					Console	Y.LU. =	MFCU1 (MFCU pri-
					DISKET	Y'EO' -	mary hopper)
				X'50' =	READ42 (1442 Card	^ LQ =	MFCU2 (MFCU
				VIET!	Read-Punch)		secondary hopper)
				X'5F' =		Model 15	i
				X'60' = X'80' =		X'01' =	Device inde-
					BSCA (first time logic)		pendent input
					DISK (5444 simulation	X'02' =	Device inde-
				,,,,o =	area)		pendent output
				X'C0' =	DISK40 or DISK45	X'18' =	
					(3340 main data area)		DISKET
				X'E0' =	PRINTER — carriage 1		READ42
					PRINTR2 - carriage 2		READ01
					MFCU1 - primary hopper	X'5F' =	SPECIAL IOS
					MFCU2 — secondary	V'60' -	routine
					hopper	X'60' =	
				X'FF' =	Invalid entry	X,80, =	DOUA

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

tem Number	Byte Length	Defined/ Modified	Descripti	on				
					··		Model 15 (continued)	_
							X'A0' = DISK44	
							X'C0' = DISK45,	
							DISK40	
							X'EO' = 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 fi	le			
		\$RPGH	*Bytes 10	1-11. Data management entry	point			
		\$RPGI	*Bytes 10	-11. Data management entry	point			
		\$RPPA	*Bytes 4-	5. Address of linkage to IPC	R routine	fo	or this file	
		\$RPPB	*Bytes 6-	7. Address of linkage to OPC	R routin	e 1	for this file	
137	2	\$RPEA		gth in binary (col 20-23)	•			
138	2	\$RPEA		ength in binary (col 24-27)				
				•				
139	1	\$RPEA	Model 6		Model 1		Blank device	
			X'10' =	CONSOLE (printer-			CONSOLE (printer-	
			34464	keyboard).	× 10 ·	_	keyboard)	
			X.12. =	KEYBORD	VIADI	_	DISKET	
			V/401	(keyboard)			READ42 (1442 Card	
				DISKET Special IOS routine			Read-Punch)	
			X,80, =		X'5F'	=	Special IOS routine	
				BSCA with first-time	X'60'		•	
			, J4 -	logic	X'80' =	=	BSCA	
			X'90' =	CRT (display station)	X'84'	-	BSCA with first	
				DISK (disk unit)			time logic	
				TRACTR1 (tractor 1)	X'A0'	=	DISK (disk unit)	
				TRACTR2 (tractor 2)	X,C0. :	=	DISK45	
				LEDGER (ledger card	X'E0'	=	PRINTER (line print-	
				device)			er - carriage 1)	
			X'F1' =	DATA96 (data recorder)	X,E8, :	=	PRINTR2 (line print-	
				or			er — carriage 2)	
				DATA29 (80 col card	X'F0' :	=	MFCU1 (MFCU pri-	
				device)	V/F51	_	mary hopper)	
				Invalid entry	Y.1-8.	_	MFCU2 (MFCU second	
			Model 12		VIEE	_	ary hopper)	
			X'10' =	·	X.L.L.	-	Invalid entry	
			X'40' =		Model 1	15		
			X'50' =	READ42 (1442 Card	X'01'	=	Device indepen-	
			V/F.5.	Read-Punch)			dent input	
			X'5F' =		X'02' =	=	Device indepen-	
			X'60' =	•			dent output	
			X'80' =				CRT77	
				BSCA (first time logic)			DISKET	
			∧ AU =	DISK (5444 simulation area)			READ42	
			X'CO' =	DISK40 or DISK45 (3340	_		READ01	
			700 -	main data area)			SPECIAL IOS routine	
			X'E0' =	PRINTER - carriage 1	X'60' =			
				PRINTR2 — carriage 2			DISK44	
				MFCU1 - primary hopper			DISK45, DISK40	

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'EO' = PRINTER X'E4' = PRINT84 X'FO' = 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 (co	of 35-38)		
142	2	\$RPEA	Cytinder 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				
145	2		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 cot 53 100 = SPECIAL used 010 = K in cot 53 110 = Invalid			
			Bits 3-4 00 = Blank or invalid 01 = U in col 70 10 = R in col 70 11 = N in col 70	in cot 70		
			Bits 5-7 Reserved			

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

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

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

Number	Byte Length	Defined/ Modified	Description
137	6	\$RPEB	Table or array name (col 27-32)
		\$RPGW/	Byte O.
		\$RPGY**	,
			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
			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 langth 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/	Byte 0
		\$RPGY**	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
			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:			X'F2' = MFCM1 X'F4' = MFCM2
Putto O Dita 2.7	Carrata		X'F8' = MFCU2
Byte 0, Bits 3-7	Sequence number		X'FF' = Invalid entry
Bytes 1-2	DTF address		•
Byte 3 (Model 6)	X'10' = CONSOLE (printer-keyboard)	Byte 4, Bits 0-3	0001 = Primary file
-, , <i>-,</i>	X'12' = KEYBORD (keyboard)		0010 = Secondary file
	X'40' = DISKET		1000 = Chained file
•	X'5F' = Special IOS routine		1001 = Demand file '
	X'80' = BSCA		1010 = Record address file
	X'84' = BSCA with first-time logic	Dia- 4 E	1100 = Table file
	X'90' = CRT (display station)	Bits 4-5	00 = No specified sequence
	X'A0' = DISK (disk unit)		01 = Descending sequence 10 = Ascending sequence
	X'E1' = TRACTR1 (tractor 1)		10 - Ascending sequence
	X'E2' = TRACTR2 (tractor 2)	Byte 5, Bits 0-1	00 = Sequential file
	X'E9' = LEDGER (ledger card device)	2,100,2100.	01 = Indexed file
	X'F1' = DATA96 (data recorder)		10 = Direct file
	X'FF' = Invalid entry		11 = ADDROUT file
		Bits 2-3	00 = Consecutive processing
Byte 3 (Model 10)			01 = Random processing
	X'40' = DISKET		10 = Indexed file processed sequentially by key
	X'50' = READ42 (1442 Card Read-Punch)		11 = Indexed file processed sequentially within limits
	X'5F' = Special IOS routine	Bits 4-6	001 = Display file
	X'60' = TAPE		011 = Update file
	X'80' = BSCA		100 = Combined file
	X'84' = BSCA with first time logic X'A0' = DISK (disk unit)		101 = Regular output file
	X'C0' = DISK45		110 = Input file
	X'E0' = PRINTER (line printer carriage 1)		111 = Unordered sequence
	X'E8' = PRINTR2 (line printer — carriage 2)	Bit 7	1 = Addition specified
	X'F0' = MFCU1 (MFCU primary hopper)	Byte 6, Bit 0	0 = Variable format
	X'F8' = MFCU2 (MFCU secondary hopper)	Byte o, bit o	1 = Fixed format
	X'FF' = Invalid entry	Bit 1	0 = Unblocked file format
			1 = Blocked file format
Byte 3 (Model 12)	X'10' = Console	Bits 2-3	01 = Extension specifications
	X'40' = DISKET	5,625	10 = External indicators
	X'50' = READ42 (1442 Card Read-Punch)	Bits 4-6	000 = Indexed file processed consecutively
	X'5F' = Special		010 = Indexed key (alphameric)
	X'60' = Tape		011 = Indexed key (packed)
	X'80' = BSCA Y'94' = BSCA (first time logic)		100 = Record identification (disk address)
	X'84' = BSCA (first time logic) X'A0' = DISK (5444 simulation area)		110 = Record number
	X'C0' = DISK40 or DISK45 (3340 main data area)	Bit 7	1 = Dual I/O
	X'EO' = PRINTER — carriage 1		
	X'E8' = PRINTR2 — carriage 2	Byte 7	Overflow mask
	X'F0' = MFCU1 — primary hopper	-,	
	X'F8' = MFCU2 - secondary hopper	Bytes 8-9	Record length
	X'FF' = Invalid entry		-
	·	Bytes 10-11	IOCB address
Byte 3 (Model 15)	X'01' = Device independent input		
	X'02' = Device independent output	 Indicates an ac 	dition or change to a previously defined area.
	X'18' = CRT77	•	RPGW or \$RPGY is loaded depending on
	X'40' = DISKET		ject-time table or a compile time table has
	X'50' = READ42	been foaded,	
	X'58' = READ01		
	X'5F' = SPECIAL IOS routine		
	X'60' = TAPE		
	X'80' = BSCA X'A0' = DISK44		
	X'CO' = DISK45, DISK40		
	X'EO' = PRINTER		
	X'E4' = PRINT84		
	X'F0' = MFCU1		
	X'F0' = MFCU1		

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

1 \$RPEC Compression length in binary 1 \$RPEC Indicates presence of item in compression when bits are on (Bit=1) 1 tem 129 Bit 0 1 = Item number 132 Bit 3 1 = Item number 13: 1 Bit 1 1 = Item number 133 Bits 4-7 Not used 1 Item 130 Not used 1 Item 131 Not used 1 RPEC Bit 0
Item 129
1 1 Item 131 Not used 2 1 \$RPEC Bit 0
1 \$RPEC Bit 0 1 = Invalid compression Bit 1 1 = Invalid or missing filename Bits 2-7 Not used 3 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'12' = KEYBORD (keyboard) X'40' = DISKET X'50' = READ42 (1442 Card X'5F' = Special IOS routine X'80' = BSCA X'58' = BSCA
Bit 1 1 = Invalid or missing filename Bits 2-7 Not used \$RPEC Filename (coi 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'10' = CONSOLE {printer- keyboard} X'12' = KEYBORD (keyboard) X'12' = KEYBORD (keyboard) X'5F' = Special IOS routine X'80' = BSCA X'5F' = Special IOS routine X'80' = BSCA with first-time Iogic X'80' = BSCA X'50' = TAPE Iogic X'80' = BSCA
\$RPGI Byte 0 Bits 3-7 Sequence Number Bytes 1-2 DTF address Byte 3 Model 6 Model 10 X'10' = CONSOLE (printer-keyboard) X'12' = KEYBORD (keyboard) X'12' = KEYBORD (keyboard) X'40' = DISKET X'40' = DISKET X'50' = READ42 (1442 Card Read-Punch) X'80' = BSCA X'5F' = Special IOS routine X'80' = BSCA with first-time Indicate the printer of the
Sequence Number Bytes 1-2 DTF address Byte 3 Model 6 Model 10 X'10' = CONSOLE (printer-keyboard) X'12' = KEYBORD (keyboard) X'40' = DISKET X'40' = DISKET X'50' = READ42 (1442 Card Read-Punch) X'80' = BSCA X'5F' = Special IOS routine X'84' = BSCA with first-time X'60' = TAPE logic X'80' = BSCA
Byte 3 Model 6 X'10' = CONSOLE (printer-keyboard) X'12' = KEYBORD (keyboard) X'40' = DISKET X'40' = DISKET X'50' = READ42 (1442 Card Read-Punch) X'80' = BSCA X'80' = BSCA with first-time logic X'80' = BSCA X'80' = BSCA
X'10' = CONSOLE (printer-keyboard) X'12' = KEYBORD (keyboard) X'40' = DISKET X'40' = DISKET X'50' = READ42 (1442 Card Read-Punch) X'80' = BSCA X'80' = BSCA X'84' = BSCA with first-time logic X'10' = CONSOLE (printer-keyboard) X'40' = DISKET X'50' = READ42 (1442 Card Read-Punch) X'5F' = Special IOS routine X'5F' = Special IOS routine X'60' = TAPE Special IOS routine X'60' = BSCA X'80' = B
X'AO' = DISK (disk unit) logic

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

		-	**		Model 15 (continued)
			Pyrto A	Disc 0.2	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	
			Byte 12	Bits 0-3 Operation Code Bits 4-7 Number of parameters	
134	2	\$RPEC	Form lengt	th in binary	
	2	\$RPEC	Overflow n	umber 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 pre	sence o	f item in compression when bits are on (f	Bit=1)		
130	1	-	Item 129	Bit 0		m 131	Bit 0	1 = Item number 14
131	1			Bit 1	1 = Item number 133		Bit 1	1 = Item number 14
				Bit 2	1 = Item number 134		Bit 2	1 = Item number 15
				Bit 3	1 ≖ Item number 135		Bit 3	1 = Item number 15
				Bit 4	1 = Item number 136		Bit 4	1 = Item number 15
				Bit 5	1 = Item number 137		Bit 5	1 = Item number 15
				Bit 6	1 = Item number 138		Bit 6	1 = Item number 15
				Bit 7	1 = Item number 139		Bit 7	Reserved
			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			
				Bits 5	7 100 = Item number 145			
					110 ≈ Item numbers 145 and 146	i		
					(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	1 = TR (col 19-20)			
			Bytes 1-2		Sequence (col 15-16)			
		\$RPJA	Bytes 1-2		X,0000,			
		\$RPGW/	*Byte 1, Bit	4	1 = Field name definition resolved			
		\$RPGY**	*Bit	: 5	1 = Index resolved			
			*Bit	7	1 = Symbol in table (set by phase \$RPHA	۹)		
		\$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	4)		
		\$RPJJ	*Byte 0, Bit	5	1 = Control fields move optimized			
		•	*Bit		1 = Match fields move optimized			
			*Bit	7	1 = Move field optimized			
		\$RPJS	Bytes 0-1		Address of the trailer table for the currer	nt		
		*****	_,		trailer specification is placed in the traile			
					record type specification			
			Byte 0, Bit 0		1 = For all record type compressions (AN	ND/OR		
					lines included) of header portion of sprea	ad card		
					to flag as header record			
		\$RPPE	*Bytes 1-2		Address for the record ID routine to retu	ırn to wh	en the rec	cord is identified
		\$RPPF	*Bytes 1-2		Address of the record identification para	meter		
		\$RPPN	*Bytes 1-2		Address of parameter list for chain and d	lemand fi	les	

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

	Byte Length	Defined/ Modified	Description	
33	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 specification.	one of one trailer of current trailer specification is placed in the record type
134 1	1	\$RPEI	Bit 0	1 = Look-ahead field (col 19-20)
			Bits 2-3	00 = Unpecked 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 pe	opagated for all look-ahead fields specified. A decimal position defaults to zero.
		\$RPGW/	*Bit 0	0 = Alphabetic sequence entry (col 15-16)
		\$RPGY**		1 = Numeric sequence entry
			*Bit 1	0 = Object-time table or array name
				1 = Compile-time table or array name
		\$RPH\$	This byte is not de	efined 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. Numer	ic portion set to packed, binary, or sterling length
135 15	15	\$RPEI	Filename (col 7-1	4)
35		\$RPGI	Byte 0, Bits 3-7	Sequence number
35		anrai	-,, -,	

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

item Number	Byte Length	Defined/ Modified	Descript	ion	
			Byte 3	Model 6	Model 10
			Dyte 3	X'10' = CONSOLE (printer keyboard)	X'10' = CONSOLE (printer
				X'12' = KEYBORD	keyboard) X'40' = DISKET
				(keyboard) X'40' = DISKET	X'50' = READ42 (1442 Card Read-Punch)
				X'5F' = SPECIAL (special	X'5F' = Special IOS routine
				IOS routine) X'80' = BSCA	X'60' = TAPE X'80' = BSCA
				X'84' = BSCA (BSCA with	X'84' = BSCA with first
				first-time logic) X'90' = CRT (display station)	time logic X'A0' = DISK (disk unit)
				X'AO' = DISK (disk unit)	X'C0' = DISK45
				X'E1' = TRACTR1 (tractor 1)	X'EO' = PRINTER (line print-
				X'E2' = TRACTR2 (tractor 2)	er — carriage 1)
				X'E9' = LEDGER (ledger card	X'E8' = PRINTR2 (line print-
				device)	er — carriage 2)
				X'F1' = DATA96 (data	X'F0' = MFCU1 (MFCU pri-
				recorder)	mary hopper)
				X'FF' = Invalid entry	X'F8' = MFCU2 (MFCU secondary hopper)
					X'FF' = Invalid entry
				Madel 12 X'10' = Console	
				X'40' = DISKET	Model 15
				X'50' = READ42 (1442 Card	X'01' = Device inde-
				Read-Punch)	pendent Input
				X'5F' = Special	X'02' = Device inde-
				X'60' = Tape	pendent output
				X'80' = BSCA	X'18' = CRT77
				X'84' = BSCA (first time logic)	X'40' = DISKET
				X'A0' = DISK (5444 simulation area)	X'50' = READ42 X'58' = READ01
				X'C0' = DISK40 or DISK45	X'5F' = SPECIAL IOS
				(3340 main data area)	routine
				X'E0' = PRINTER — carriage 1	X'60' = TAPE
				X'E8' = PRINTR2 - carriage 2	X'80' = BSCA
				X'F0' = MFCU1 — primary hoppe	
				X'F8' = MFCU2 secondary	X'C0' = DISK45, DISK40
				hopper X'FF' = Invalid entry	X'EO' = PRINTER X'E4' = PRINT84
				, and and an arrange	X'F0' = MFCU1
					X'F2' = MFCM1
					X'F4' = MFCM2
					X'F8' = MFCU2
					X'FF' = Invalid entry
			Byte 4	Bits 0-3	Bits 4-5
			2,104	0001 = Primary file	00 = No specified sequence
				0010 = Secondary file	01 = Descending sequence
				1000 = Chained file 1001 = Demand file	10 = Ascending sequence
				1001 = Demand file 1010 = Record address file	Bit 6 1 = End of file
					Bit 7 1 = File translate
			Byte 5	Bits 0-1	Bits 4-6
				00 = Sequential file	001 = Display file
				01 ≈ Indexed file	011 = Update file 100 ≈ Combined file
				10 = Direct file 11 = ADDROUT file	100 - Compined life 101 = Regular output file

Figure 3-5. Input Compressions (Part 3 of 6) 3-30

item Number	Byte Length	Defined/ Modified	Description			
			-	110 = Input file		
			Bits 2-3 00 = Consecutive processing 01 = Random processing 10 = Indexed file processed sequentially by key 11 = Indexed file processed sequentially within limits	111 = Unordered sequence Bit 7 1 = Addition specified		
		Byte 6	Bit 0 0 = Variable format 1 = Fixed format Bit 1	Bits 4-6 000 = Indexed file processed consecutively 010 = Indexed key (alphameric)	Bytes 10-	Overflow mask Record length -11 IOCB address
			0 = Unblocked file format 1 = Blocked file format Bits 2-3 01 = Extension or line counter specifications 10 = External indicators	011 = Indexed key (packed) 100 = Record identification (disk address) 110 = Record number Bit 7 1 = Dual I/O	Byte 12 Bits 4-7 Number o	Bits 0-3 Operation code of parameters
	\$RPPJ	*Bytes 1	3-14 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 (c	ol 44-47)		
		\$RPJA	See note 2			
144	2	\$RPEI	TO field location in binary (col 4	8-51)		
		\$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/	If an array name, the following is set:
		\$RPGY**	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 O. 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	\$RPE1	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	\$RPE1	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	

N	ote	4		
ľ	Ote	- 1	ï	

Byte 0, Bit 0

Bits 4-7

Bits 4-7

Bits 4-7

0011 = C entry (col 25, 32, 39)

0100 = D entry

1001 = Z entry

0000 = Entry not specified

Byte 1

Character (col 27, 34, 41)

 $\underline{\underline{Note~2}};~$ 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.

<u>Note 3:</u>

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

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

^{*} 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.

ltem 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 = I tem number 137
			Bit 6 1 = 1tem 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 = 1tem 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/	*Byte 1, Bit 0 1 = Factor 1 resolved
		\$RPGY**	*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.
				of leading array control, or for move type operation code, the first byte byte mask (second byte is in item 134)
		\$RPMP	*Bytes 0-1, Addre	ess of generated object code
133	1	\$RPEK	Bit 0	1 = SR (col 7-8)
			Bit 2 Bit 3	1 = Half adjust (col 53) 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/	*Bit 0	0 = Object-time table (Factor 1)
		\$RPGY**	ATT: 4	1 = Compile-time table (Factor 1)
			*Bit 1	0 = Object-time table (Result Field)
			*Bit 2	1 = Compile-time table (Result Field) 0 = Object-time table (Factor 2)
			511.2	1 = Compile-time table (Factor 2)
			Bit 3	Not used
		\$RPGX	*Bits 4-7. Not us	sed
		\$RPLG	Permanent length	(bit 4 or 5 for SET or KEY)
		\$RPMB	Mask of object co	ode to be generated
		\$RPLJ	*Bits 0-2	000 = Tabset 100 = Space/skip and/or position print element
				010 = Ledger card eject
			*Dian 4.7	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		er of object code to be generated or second byte of a 2-byte mask for a move The first byte is in item 132.
	5	\$RPEK	Operation codes	

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	Bytas 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 alphameric 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 D. 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/	If an array name, the following are set:
		\$RPGY**	Byte O. 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)

tem 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		Model 10			
			X'10' =	CONSOLE (printer-	X'10' = CONSOLE (printer-			
			V/4.01	keyboard)	keyboard)			
				KEYBORD (keyboard)	X'40' = DISKET X'50' = READ42 (1442 Card			
			X'40' = X'5F' =	Special IOS routine	Read-Punch)			
			X'80' =		X'5F' = Special IOS routine			
			X'84' =	BSCA with first-time	X'60' = TAPE			
				logic	X'80' = BSCA			
				CRT (display station)	X'84' = BSCA with first-time			
				DISK (disk unit)	logic			
				TRACTR1 (tractor 1)	X'A0' = DISK (disk unit) X'C0' = DISK45			
				TRACTR2 (tractor 2) LEDGER (ledger	X'E0' = PRINTER - carriage 1			
			V [3 -	card device)	X'E8' = PRINTER — carriage 2			
			X'F1' =	DATA96 (data recorder)	X'F0' = MFCU1 (MFCU primary			
				Invalid entry	hopper)			
					X'F8' = MFCU2 (MFCU second-			
					ary hopper)			
					X'FF' = Invalid entry			
			Model 12		Model 15			
			X'10' =		X'01' = Device indepen-			
			X'40' = Y'60' =	DISKET READ42 (1442 Card	dent input Y'02' = Device indepen			
				Read-Punch)	X'02' = Device indepen- dent output			
			X'5F' =	· ·	X'18' = CRT77			
			X'60' =		X'40' = DISKET			
			X * 80' =	BSCA	X'50' = READ42			
				BSCA (first time logic)	X'58' = READ01			
				DISK (5444 simulation	X'5F' = SPECIAL IOS			
				area) DISK40 or DISK45	routine X'60' = TAPE			
				(3340 main data area)	X'80' = BSCA			
				PRINTER — carriage 1	X'A0' = DISK44			
				PRINTR2 - carriage 2	X'CO' = DISK45, DISK40			
				MFCU1 - primary hopper	X'EO' = PRINTER			
				MFCU2 - secondary	X'E4' = PRINT84			
				hopper	X'F0' = MFCU1			
			X'FF' =	invalid entry	X'F0' = MFCU1 X'F2' = MFCM1			
					X'F4' = MFCM2			
					X'F8' = MFCU2			
					X'FF' = Invalid entry			
			Byte 4 Bits 0-3		Bits 4-5			
			0001 =	Primary file	00 = No specified sequence			
				Secondary file	01 = Descending sequence			
				Chained file	10 = Ascending sequence			
				Demand file Record address file	Bit 6			
			1010 -	HECOLO BOOLESS HIS	1 = End of file			
					Bit 7 1 = File translate			

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

ltem Number	Byte Length	Defined/ Modified	Description				
			01 = Inde 10 = Dire 11 = ADI Bits 2-3 00 = Con 01 = Rand 10 = Inde sequents	ect file DROUT file secutive processing dom processing exed file processed sentially by key exed file processed sentially within	011 = U 100 = C 101 = F fi 110 = I	Display file Ipdate file Combined file Regular outpu Regul	rt
		\$APLN	For a literal, the for Byte 0. See note 3. Byte 1. See note 4.	ollowing are set:			
		\$RPMH	 Address of the Address of the 	rightmost byte of the	lement or field part of the eler	nent or field	or MHLZO operation code specific to be moved if MOVEL operation of the Result Field.
		\$RPLR	For a literal, the for Bytes 2-3. Address Bytes 4-5. These	s of rightmost byte of	literal		
		\$RPMP	If BEGSR and EX	SR are specified:			
			Bytes 0-1	Start address of sub	routine		
			Bytes 2-3	Length of subroutin	e		
			Bytes 4-5	Subroutine identifie	r		
		\$RPQD/ \$RPQG/ \$RPQH \$RPQE	Bytes 2-3. Object variable index)	code address of where	element addre	ss is to be pla	aced (for array table or array with
145	4	\$RPEK	Item acts as a 4-by Factor 2.	rte extension to item 1	44 when an alp	hameric or n	numeric literal is defined in
146	3	\$RPEK	Item acts as a 3-by or when filename		44 when an arr	ay with cons	tant index is specified in Factor 2
		\$RPGW/ \$RPGY**	-	a constant index, the fic value of the array in	_	t:	
		\$RPGI	If filename is spec	ified in Factor 2, the fo	ollowing are set	::	
			Byte 0, Bit 0 Bit 1 Bits 2-3	0 = Variable format 1 = Fixed format 0 = Unblocked file for 1 = Blocked file for 01 = Extension or li	nat	Byte 1 Byte 2	Overflow mask First byte of record length, second byte in item number 147
			2 2 0	specifications 10 = External indica			

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: Address of the leftmost byte of the element or field if MHHZO or MLHZO operation code specified. 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 3: Note 1: The indicators are reordered so that indicator strings that began 1. Alphameric length = Length minus one in this compression are placed back into the compression with 2. Numeric length = length minus one in the numeric portion; the longest first, in decreasing order by length. The start of an number of decimal positions in the zone portion indicator string is flagged by setting Bit 0 equal to zero in the displacement for the indicator. In the next compression, the Note 4: length of the string minus the length of the group in which it started is placed in the same relative position as in this com-Bit 0 1 = Look-ahead field pression. In all subsequent compressions through which the Bits 2-3 00 = Field name string continues, the indicator is replaced by X'0000', a no-01 = Array name test indicator. 10 = Table name Bits 4-5 00 = Table sequence not specified Note 2: 01 = Table is descending or field name is **UDATE** related field X'00' = ENDSRByte 0 10 = Table is ascending or, if RLABL X'01' = EXCPT specified, field name or mask and X'0E' = SETON displacement or indicator X'OF' = SETOF Bit 6 0 = Length is alphameric (see Note 10) X'20' = RLABL1 = Length is numeric X'23' = MVR Bit 7 1 = Shorter array X'2E' = TESTZ During phase \$RPHC, 1 = Field X'40' = EXITis used X'41' = GOTO X'42' = EXSR X'50' = DEBUG * Indicates an addition or change to a previously defined area. X'51' = DSPLY** Either phase \$RPGW or \$RPGY is loaded depending on X'52' = FORCE whether an object-time table or a compile-time table has X'53' = READ been loaded. X'60' = Z-ADD X'61' = Z-SUBX'62' = SORTX'63' = XFOOTX'64' = MHHZO X'65' = MHLZO X'66' = MLHZOX'67' = MLLZO X'68' = MOVEX'69' = MOVEL X'6A' = MOVEA X'6C' = BITON X'6D' = BITOF X'6E' = TESTB X'71' = KEY Model 6 only X'72' = SET X'80' = BEGSR X'81' = TAG X'84' = TIME (Model 15 only) X'CE' = COMP X'CF' = LOKUP X'DO' = CHAINX'D3' = SETLL X'EO' = ADDX'E1' = SUB X'E2' = MULT X'E3' = DIV X'FF' = Invalid entry Byte 1, 1 = Field light 1 Rit 0 (for Bit 1 1 = Field light 2 KEY or Bit 2 1 = Field light 3

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

1 = Field light 4

1 = Field light 5

1 = Field light 6

1 = Field light 7

1 = Field light 8

Model 6 only

Bit 3

Bit 4

Bit 5

Bit 6

Bit 7

SET

operation codes)

Item Number	Byte Length	Defined/ Modified	Description					
128	1	\$RPEO	Compression length in binary					
129 130	1	\$RPEO	Indicates presence of an item in the compression. Item 129 Bit 0					
131	1		Item 131 Bit 0 1 = Item number 148 Bit 1 1 = Item number 149 Bits 2-7 Not used					
32	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)

	ltem Number	Byte Length	Defined/ Modified	Description	
Ì	133	1	\$RPEO	Bits 0-1	00 = Blank or invalid type (col 15) 01 = Heading or detail records 10 = Total records 11 = Becords to be written during calculation time (F in col 15)
				Bit 2 Bit 3 Bit 4 Bits 5-7	11 = Records to be written during calculation time (E in col 15) 1 = ADD (col 16-18) 1 = AND (col 14-16) 1 = OR (col 14-15) 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 Bit 6 Bit 7	1 = Not (col 23) 1 = Not (col 26) 1 = Not (col 19)
			\$RPGW/ \$RPGY**	*Bit 1	0 = Object-time table name 1 = Compile-time table name
			\$RPLB	*Bit 0 *Bits 1-2	1 = No field lines follow 01 = 1P indicator 10 = LR indicator 11 = Overflow indicator
				*Bit 3 *Bit 4	1 = First record for output type 1 = Last record or field for output record type
	135	1	\$RPEO	Bit 0 Bit 1 Bits 2-3	1 = PAGE, PAGE1, or PAGE2 (col 32-37) 1 = Blank After (col 39) 00 = Unpacked or alphameric (col 44) 01 = Packed 10 = Binary
				Bit 4 Bit 5 Bit 6	1 = *PLACE (col 32-37) 1 = Ledger card control identification (N in col 40) 1 = *PRINT (col 32-37) Model 10 or 12
				Bit 7	1 = MFCU print (* col 40) Model 10 or 12
1			\$RPGB	Bit 7	1 = MFCM print (Model 15 only) osition on record (MFCM record specifications only).
			\$RPJO \$RPLR		dress if more than one byte of Data Management parameters
			\$RPLG	Х'00' ≃ по ра	arameter required neter required
	136	2	\$RPEO	Space before, Bits 0-3	/after (col 17-18) X'0' — X'9' = Space before entry
				Bits 4-7	X'F' = No entry X'0' — X'9' = Space after entry X'F' = No entry
			\$RPJO	record for V	lited output (field specification only); high end position on LR on tape (record specification only, Item 135 and byte 0 used); high punch position on record (MFCM record is only).
			\$RPLR	•	dress (item number 135 and Byte 0 of item number 136)
			\$RPLV	•	arameter required meter required
	137	1	\$RPEO	Skip before i	n binary (col 19-20) (optional)

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

em umber	Byte Length	Defined/ Modified	Description				
8	1	\$RPEO	Skip after in bi	nary (col 20-21) (optional)			
ı	15	\$RPEO	Filename (col 7				
		\$RPGI	Byte 0, Bit 0 Bits 3-7	1 = MFCU print Sequence number			
			Bytes 1-2 Byte 3 Model (DTF address	Model 1	0	
			•	CONSOLE (printer-		CONSOLE (printe	er-
			X 10	keyboard)	λ 10	keyboard)	•
			X'12'	- KEYBORD	X'40' =	DISKET	
				(keyboard)	X'50' =	READ42 (1442 C	ard
			X'40'	DISKET		Read-Punch)	
			_	Special IOS routine		Special IOS routing	ie
				BSCA		TAPE	
			X'84' =	BSCA (BSCA with		BSCA	
			****	first-time logic)	X'84' =	BSCA with first-	
				= CRT (display station)	VIAOI -	time logic	
				DISK (disk unit)		DISK (disk unit) DISK45	
				TRACTR1 (tractor 1) TRACTR2 (tractor 2)		PRINTER (line pr	int_
				E LEDGER (ledger card	7 EU -	er — carriage 1)	1140-
			Y.E9. =	device)	X'FR' =	PRINTR2 (line pr	int-
			Y'E1' -	DATA96 (data	7 EQ -	er - carriage 2)	IIIL-
			AFT.	recorder)	X'F0' =	MFCU1 (MFCU p	ri-
			X'FF'	Invalid entry		mary hopper)	· -
			7.1		X'F8' =	MFCU2 (MFCU	
						secondary hopper	1
					X'F F' =	Invalid entry	
			Model 1	-	Model 1		
				Console	X'01 =	Device Indepen-	
				DISKET		dent Input	
			X'50' =	READ42 (1442 Card	X'02' =		
			V:-F:	Read-Punch)	VIAN	dent Output	
				Special Tape		CRT77	
			X'60' = X'80' =	BSCA		DISKET READ42	
				BSCA (first time logic)		READ01	
				DISK (5444 simulation		SPECIAL IOS	
			7 AV -	area)	7.01	routine	
			X'CO' =	DISK40 or DISK45	X'60' =		
			700 -	(3340 main data area)	X'80' =		
			X'EO' =	PRINTER — carriage 1		DISK44	
				PRINTR2 — carriage 2		DISK45, DISK40	
				MFCU1 — primary hoppe		PRINTER	
				MFCU2 — secondary		PRINT84	
				hopper		MFCU1	
			X'FF' =	Invalid entry		MFCM1	
					X'F4' =	MFCM2	
					X'F8' =	MFCU2	
					X'FF' =	Invalid entry	
			Byte 4 Bits 0-3		Bits 4-5		Bit 7
				,	-	cified sequence	1 = File translate
						ding sequence	
					10 = Ascend	ing sequence	
				Demand file Record Address file	Bit 6		

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 01 = Random processed sequentially by key 11 = Indexed file processed sequentially within limits Bits 4-6 001 = Display file 100 = Combined file 110 = Regular output file 111 = Input file 111 = Unordered sequence 111 = Addition specified				
			Byte 6 Bit 0 0 = Variable format 1 = Fixed format Bit 1 0 = Unblocked file format 1 = Blocked file format 1 = Blocked file format Bits 2-3 01 = Extension or line counter specifications 10 = External indicators Byte 7 Overflow mask Bytes 8-9 Record length 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 Bytes 10-11 IOCB address				
		\$RPPJ	Bytes 12-13. Address of OPCR routine				
140	2	\$RPEO \$RPGF \$RPLB	Output indicator (col 24-25) (optional) Mask and displacement 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.				
	143-field n 143-array r 143 and 14	ame (6 byte name (6 byte 14-array witl					

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

itam Numbe	Byte r 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 \$RPJO*	Converted end position for Print Only on MFCM files. 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 = Le	ength 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	-	mulative length of combined Output Fields moves if a constant is present If constant, address is adjusted by the length added to it			

Note 1 (Items 143, 145):

Bit 0 Bits 2-3	1 = Look-ahead field 00 = Field name 01 = Array name	Bit 6	0 = Length if alphameric 1 = Length is numeric
Bits 4-5	10 = Table name 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	** Either ph	s an addition or change to a previously defined area. hase \$RPGW or \$RPGY is loaded depending on an object-time table or a compile-time table has ded.

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

Number	Byte Length	Defined/ Modified	Description					
128	1	\$RPEE	Compression length in binary					
129	1	\$RPEE	Indicates pre Item 129	Bit 0 Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6	tem in compression when bits are on (Bit=1) 1 = Item number 132 1 = Item number 133 1 = Item number 134 1 = Item number 135 1 = Item number 136 1 = Item number 137 1 = Item number 138			
130	1		Item 130	Bits 0-3	1 = Item number 139 1000 = Item number 140 1100 = Item number 140			
				Bits 4-7	extension) 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			
				Bit 4 Bit 5 Bit 6 Bit 7	extension} 1 = Item number 152 1 = Item number 153 1 = Item number 154 1 = Item number 155			

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

ltem Number	Byte Length	Defined/ Modified	Description	
132	2	\$RPEE	Byte O, Bit O	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
155	6	\$RPEE	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 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 2 (Items 140, 143, 144, 147, 148, 151):			
Note 1 (Items 140, 143, 144, 147, 148, 151):	Bit 0 Bits 2-3	1 = Look-ahead field 00 = Field name		
1. Alphameric length = Length minus one		01 = Array name 10 = Table name		
Numeric length = length minus one in the numeric portion; number of decimal positions in the zone portion	Bits 4-5	00 = Table sequence not specified 01 = Table is descending or, if RLABL specified, field name or mask and displacement of indicator		
Figure 3-8. Telecommunications Compressions (Part 5 of 5)	Bit 6	0 = Length is alphameric (see Note 1) 1 = Length is numeric		

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 displace-
		ment of the indicator
	6	0 = Length is alphameric
		1 = Length is numeric
13-14		First element address (rightmost
		byte)
15-16		DTT address
12-10		DIT addiess

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:

Contents

Byte

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
0 (Bits 4-7)	0001 = Input file 0010 = Output file
0 (Bits 4-7)	0010 = Output file
0 (Bits 4-7)	0010 = Output file 0011 = Update/combined file
0 (Bits 4-7)	0010 = Output file 0011 = Update/combined file 0100 = Add/print file
0 (Bits 4-7)	0010 = Output file 0011 = Update/combined file 0100 = Add/print file 0101 = Input + add file
0 (Bits 4-7)	0010 = Output file 0011 = Update/combined file 0100 = Add/print file 0101 = Input + add file 0110 = Output + add file
0 (Bits 4-7)	0010 = Output file 0011 = Update/combined file 0100 = Add/print file 0101 = Input + add file
0 (Bits 4-7)	0010 = Output file 0011 = Update/combined file 0100 = Add/print file 0101 = Input + add file 0110 = Output + add file 0111 = Update/combined + add file
0 (Bits 4-7)	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
0 (Bits 4-7)	0010 = Output file 0011 = Update/combined file 0100 = Add/print file 0101 = Input + add file 0110 = Output + add file 0111 = Update/combined + add file

Page of LY21-0501-5 Issued 24 September 1976 By TNL: LN21-5423

				By 114L. ER21-3423
1 (Model 6)	X'18' = CONSOLE (printer-	1 (Model 15)	X'01' = Device	independent input
- (keyboard)			independent output
	X'14' = KEYBOARD		X'1C' = CRT7	
			X'40' = DISKI	
	(keyboard)		X'50' = READ	
	X'40' = DISKET		X'58' = READ	
	X'80' = BSCA		X'60' = TAPE	
	X'84' = BSCA (BSCA with first-		X'80' = BSCA	
	time logic)			with first-time logic
	X'94' = CRT (display station)		X'A0' = DISK	
	X'A0' = DISK (disk unit)			volume DISK (5444)
	X'A1' = Multivolume Disk		X'CO' = DISK'	, ,
	X'E4' = TRACTR1 (tractor 1)			olume DISK45
	X'E4' = TRACTR2 (tractor 2)		X'EO' = PRIN'	
	X'E8' = LEDGER (ledger card		X'E4' = PRIN'	
	device)		X'F0' = MFCU	
	X'F4' = DATA96 (data		X'FO' = MFCU	
	recorder)		X'FC' = MFCN	
	X'FF' = Invalid entry		X'FF' = Invalid	
	XIII - invalid entry	•		
1 (Model 10)	X'10' = CONSOLE (printer-	External module na	ames are form	ed from a 16-
1 (1.20002 - 0)	keyboard)	character EBCDIC	alphabet. Eac	ch character is trans-
	X'40' = DISKET	lated into a hexade	cimal equivale	ent when the module
	X'50' = READ42 (1442)	name is compressed	i. In this way	, a 4-byte module
	X'60' = TAPE	name is compressed	-	-
		_	_	s and the correspond-
	X'80' = BSCA	ing internal hexade		_
	X'84' = BSCA with first-time			
	logic	EBCDIC Chai	racter	Corresponding Internal
	X'A0' = DISK (5444)	in Module Na	me .	Hexadecimal Character
	X'A1' = Multivolume DISK (5444)			
	X'C0' = DISK45	A		0
	X'C1' = Multivolume DISK45	В		1
	X'E0' = PRINTER (line printer-	С		2
	carriage 1)	D		3
	X'E1' = PRINTR2 (line printer-	F		4
	carriage 2)	G		5
	X'F0' = MFCU1 (primary	н		6
	hopper)	I		7
	X'F0' = MFCU2 (secondary	L		8
	hopper)	M		9
	порры			
2 (Model 12)	X'10' = Console	0		A
	X'40' = DISKET	P		В
	X'50' = READ42 (1442 Card)	R		C
	Read-Punch)	S		D
	X'5F' = Special	T		E
	X'60' = Tape	U		F
	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			

Page of LY21-0501-5 1ssued 24 September 1976 By TNL: LN21-5423

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			Byte	Bit	Contents
format:			12		Device type
Byte		Meaning	13	0-3	0001 = Primary file
0		Length of entry			0010 = Secondary file 1000 = Chain file
1-2		Statement number of error			1001 = Demand file 1010 = Record address file
3-4		Binary number of error number		4-5	1100 = Table 00 = No sequence
	The compression block table (CZATAB) gives the address of the error file.				10 = Ascending file 01 = Descending file
				6 7	1 = End of file (col 17) 1 = File translate
FILE INP	UT/OUT	PUT TABLE			
This table		d by about CDDCII and according to	14	0-1	00 = Sequential file
		d by phase \$RPGH and remains in by the remaining Assign phases.			01 = Indexed file 10 = Direct file
_					11 = ADDROUT file
		e entry in the table for each file in the		2-3	00 = Consecutive processing
		e file description specifications except ry is first. The format of this table is:			01 = Random processing 10 = Indexed file processed sequentially by key
Byte	Bit	Contents			11 = Indexed file processed sequentially within limits
0-7		Symbolic filename		4-6	110 = Input file 100 = Combined file
8	0	1 = Look-ahead field			011 = Update file
	1	1 = *PRINT or * in column 40			101 = Ordered output specified
	2	(Models 10 and 12) 1 = *PRINT used (Models 10			<pre>111 = Unordered output specified</pre>
	2	and 12)			001 = Display file
	3	1 = From Filename with address placed in COMMON at		7	1 = Add records
		СОМВКС	15	0	0 = Variable record 1 = Fixed record
	4	1 = To Filename with address placed in COMMON at		1	0 = Unblocked file
		COMBLC		-	1 = Blocked file
	5	1 = Line counter specifications for this file		2-3	01 = Extension code specified (col 39)
	6	1 = Printer type file or BSCA printer file			10 = External indicator used to condition file
9		Sequence number of file descrip- tion compression in which file- name is defined		4-6	000 = Consecutive processing 010 = Key (alphameric) 011 = Packed key 100 = By ADDROUT file
10-11		IOCB address		7	110 = By relative record number 1 = Dual I/O

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

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

	tollowing	tormat:	
Record length	Byte	Bit	Contents
Key start location	0	0-1	Type
			00 = Subroutine
Key length			01 = Table
			10 = Mainline segment
Overflow indicator mask			11 = Subsegment
		2	Segment is subsegmented or first of a
External indicator			substructure list if Type = 10
		3	User subroutine if Type = 00
Index in storage		4-6	100 = Segment is main overlay or
			start of suboverlay
Number of extents	•		110 = Segment is contained in main
			overlay
Operation code and parameters			101 = Segment is contained in sub-
DTP 1			overlay
DTF length		7	Overlay uses both areas (on mainline
DVE - 44			only)
DTF address	1		Overlay priority (X'FF' = duplicate
Unreferenced			segment)
Officierenced	2-3		Address of object code
	4-5		Length of object code
	6-7		Substructure pointer
	8-9		Identifier for each segment
phase \$RPGV and used by	10-11		Duplicate chain
table contains one 12-byte entry	12-13		Volatile duplicate flag
ecified in this format:	14-15		Transfer vector size

FILENAME TABLE

Bit

11-15

Byte

16-17

18-19

20-21

22

23

24

27

28

29

32

Byte

30-31

25-26

Bit

Contents

Block length

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

Contents

			111
0-7		Filename	ch
8-9		Record length	
10-11	0	1 = Filename not used	F
10-11			
	1	1 = Extension specified but not found	\$1
	2	1 = Line counter specified	to
	-	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	

Statement number

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

INAL SEGMENT LIST (MODEL 15)

RPRX builds the 16-byte segment list and \$RPRZ adds

oit. The	format	of 16-byte segment list is:
Byte	Bit	Contents
0		Туре
		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	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. Length of object code or:	3	0 1 2 3	Usage mask 1 = Total calculations 1 = Detail calculations 1 = Program Close Mainline 1 = LR and Overflow Control Mainline 1 = Open Mainline
		 ID of referenced segment if TYPE = A0, 30, or 20. X'0000' if TYPE = 10 or 00. 		4 6 7	1 = Total output 1 = Detail output
6-7		Controlling identifier (a pointer which chains entries together), or ID if referencing segment if TYPE = A0,	4-5		Object code address (built by phase \$RPLR)
		30, or 20.	6-29		Object code
8-9		Identifier for the segment or X'0000' if TYPE = A0, 30, or 20.	INTERN	AL SYM	BOL TABLE (MODELS 6, 10, AND 12)
10-13		Reserved		-	ol table built by phase \$RPSE is used by PSI, and \$RPSK. The 3-byte table
14-15		Name: — Compressed name from 16-character	format is	:	
		alphabet if TYPE = 00 or 20. - Last two characters of user sub-	Byte	Bit	Contents
		routine name if TYPE = 10 or 30. - End + I address of referenced segment if TYPE = A0 and reference	0-1		Disk address of the first sector of sorted object code blocks
		segment TYPE = 40. - End + 1 address of this segment if TYPE = 40, 80, or C0.	2		Number of sectors of sorted object code blocks
		1111 – 40, 60, 61 CO.	3	0	1 = Object code sorted
GENER	AL STO	RAGE TABLE		1	1 = Mainline routine is an overlay
and cont	ains con	ge table is built by phase \$RPLN stants, edit patterns, literals, and meters. Phase \$RPLR modifies and		2	1 = Mainline routine contains suboverlays
		The format of the general storage		3	1 = Mainline processed by final out- put phases
Byte	Bit	Contents		4	0 = Root segment to be generated 1 = Overlays to be generated

0

1-2

5

6

7

Not used

1 = Mainline completely in storage

1 = Number of sectors of sorted object code blocks exceeds 225

Length of table element

Statement number

NAME TABLE

Phase \$RPGU builds a 16-byte entry for each objecttime 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 2-3	1 = Look-ahead field 00 = Field name 01 = Array name
	4-5	10 = Table name 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
	6	and displacement of the indicator 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 t	ext		
n+1	Length of the relocation dictionary (RLD). This entry is zero if no RLD is present			
n+2-end	Relocati	on dictionary (RLD)		
		ocation dictionary entry contains two 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'CO' = 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
	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
0-7		Segment 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	Byte	Contents
12-13	Record available indicator	0	T (denotes text-RLD record)
14-15	Address of the leftmost byte of this station ID entry	1	Length minus 1 of object text contained in the record
16	Number of bytes in this station ID entry	2-3	Address of the rightmost byte of object text in the record
17-18	Address of the leftmost byte of remote station ID entry	4-63	Object text begins in byte 4; 1-byte RLD (relocation dictionary) entries are inserted beginning in byte 63
19	Number of bytes of remote station ID entry		from right to left. Unused bytes (at least one) between text and RLD contain X'00'. RLD points to the
20-21	Address of leftmost byte of Dial Number entry		right end of the address displaced from beginning of text.
22	Number of bytes of Dial Number entry	Each RLD entry commost text byte in the	ntains the displacement from the left-
23-24	Address or polling characters		
25-26	Address of corresponding file input/	Model 15	
23-20	output table entry		System Services Logic Manual,
27	Remote device selected	SY21-0034, for tex	t-RLD record format for the Model 15.
28	Lines to space after	DISK WORK AREA	AS
29	Lines to space before		er requires that two disk work areas be
30	Line count	show the general us	and \$SOURCE. Figures 3-9 and 3-10 age of these areas. For more detailed
31	Page size		phases mentioned, see the description where in this manual.
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:

Initiating	Non-Ov		Over	
Phase	\$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		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 root phase	
	Puts out 10-byte segment list and sorted text for each mainline segment		Puts out 10-byte segment list and sorted text for each mainline segment	
\$RPSG (called only				10-byte segment list
for overlay)				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 main- line 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	r every overlay generated			

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

\$RPEA		
Anno. (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	
R. Modules	S-ESL RECORDS T-TEXT-RLD RECORDS S- T S- T- E-END RECORD	

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:

- 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 PRO-GRAM, BUT IT INDICATES THE CONDI-TIONAL 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

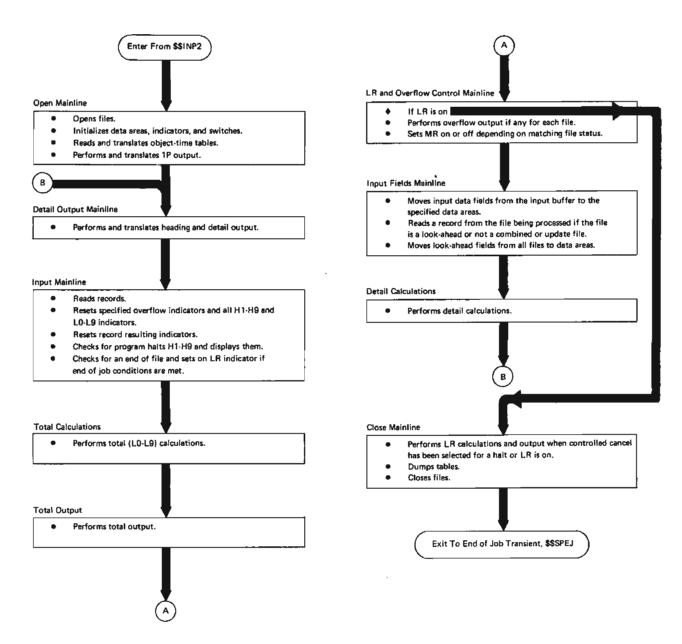


Figure 4-1. Object Program Flow

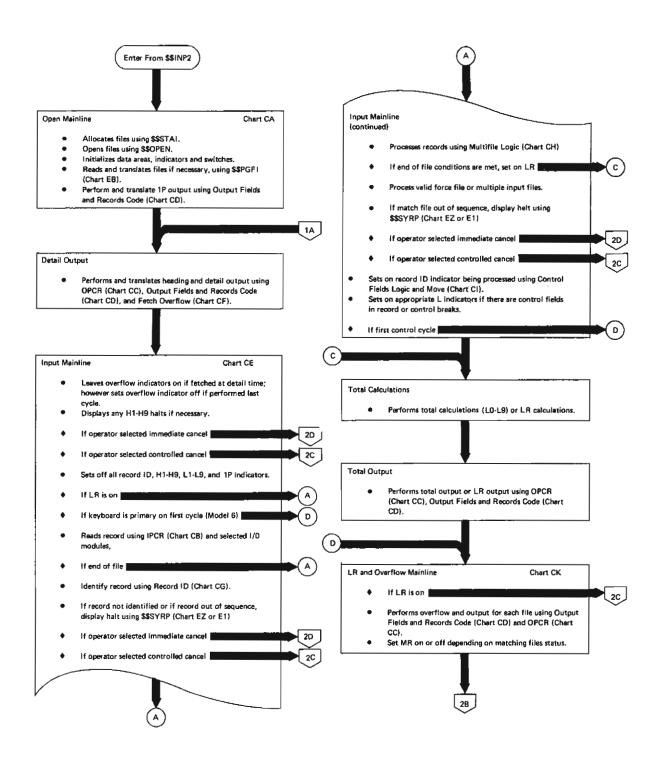


Figure 4-2 (Part 1 of 2). Intermediate Object Program Flow

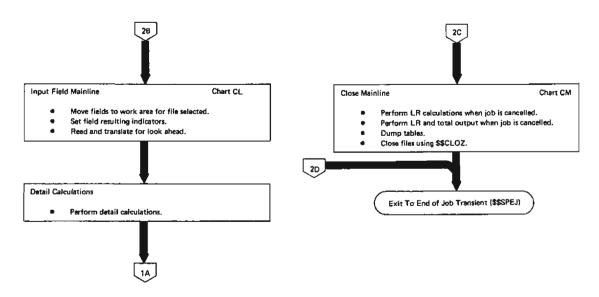


Figure 4-2. Intermediate Object Program Flow (Part 2 of 2)

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

- Resets specified overflow indicators and all H1-H9 and L0-L9 indicators.
- 2. Resets record identifying indicators.
- Checks for program halts (H1-H9) and displays them.
- 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 1D (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.
- 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:

- Calls the Program Close Mainline if the LR indicator is on.
- 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.
- 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

in the mistraction

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.

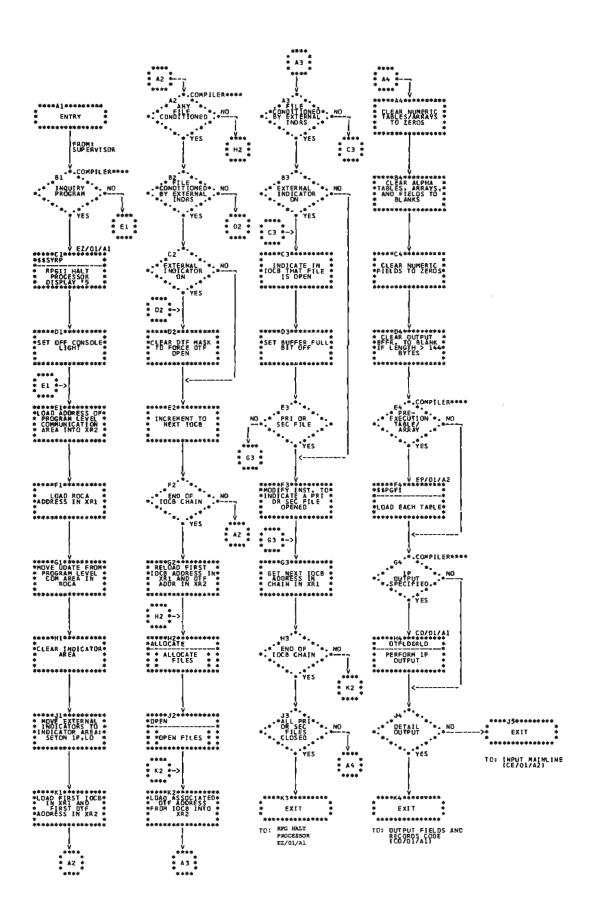


Chart CA. Open Mainline

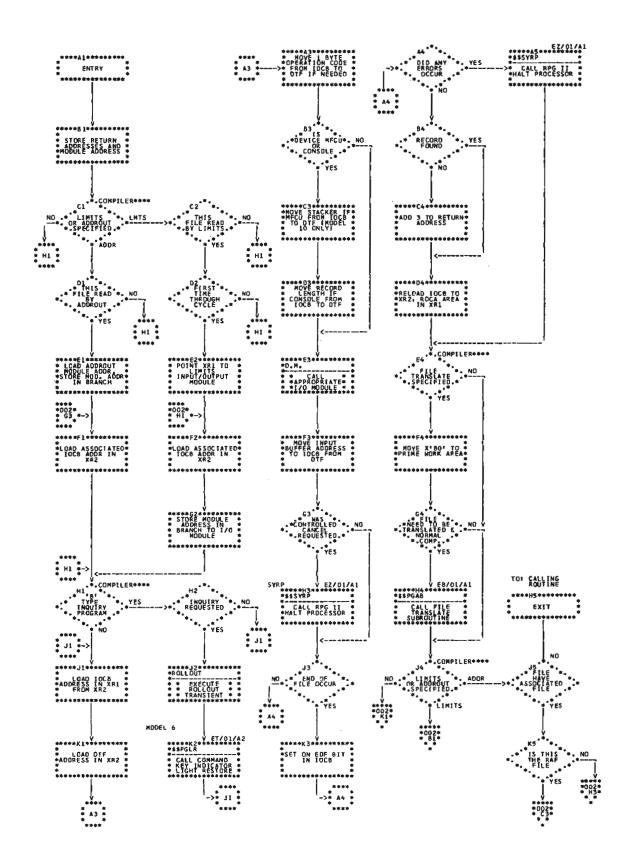


Chart CB (Part 1 of 2). Input Processing Control Routine

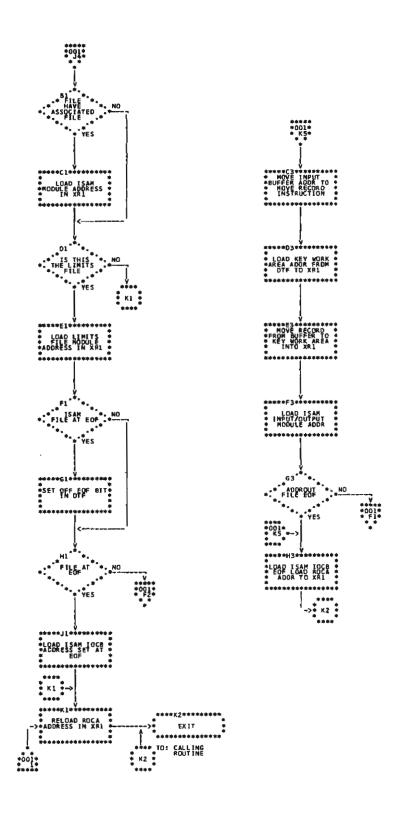


Chart CB (Part 2 of 2). Input Processing Control Routine

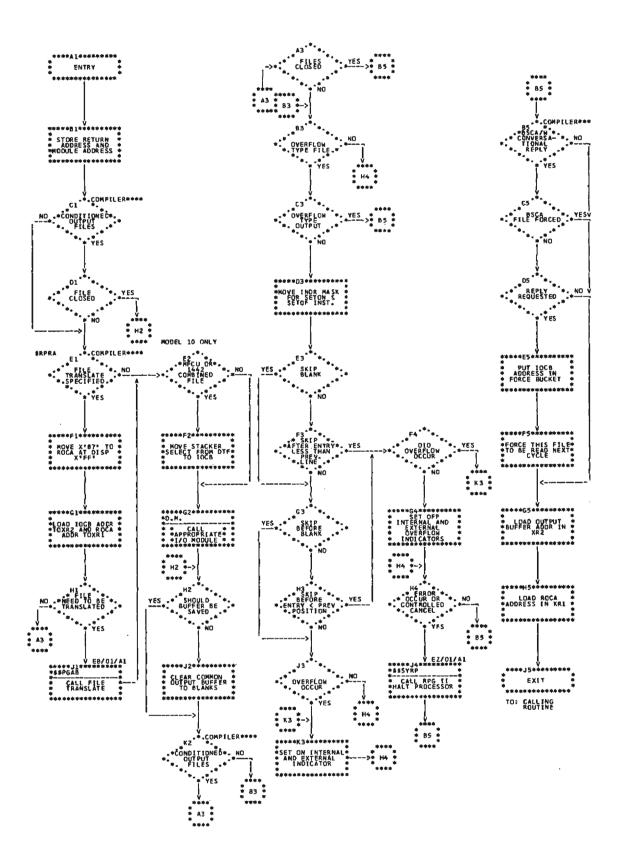


Chart CC. Output Processing Control Routine

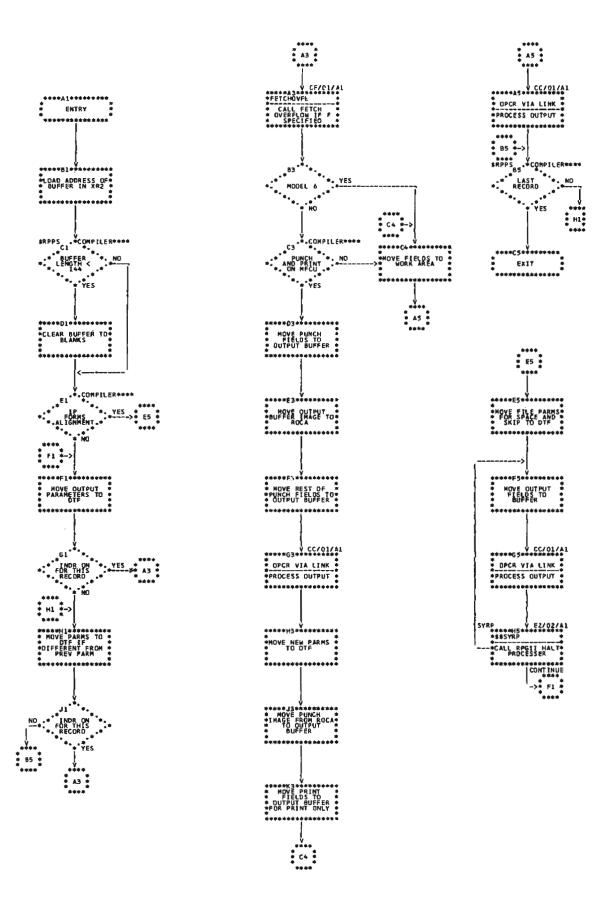
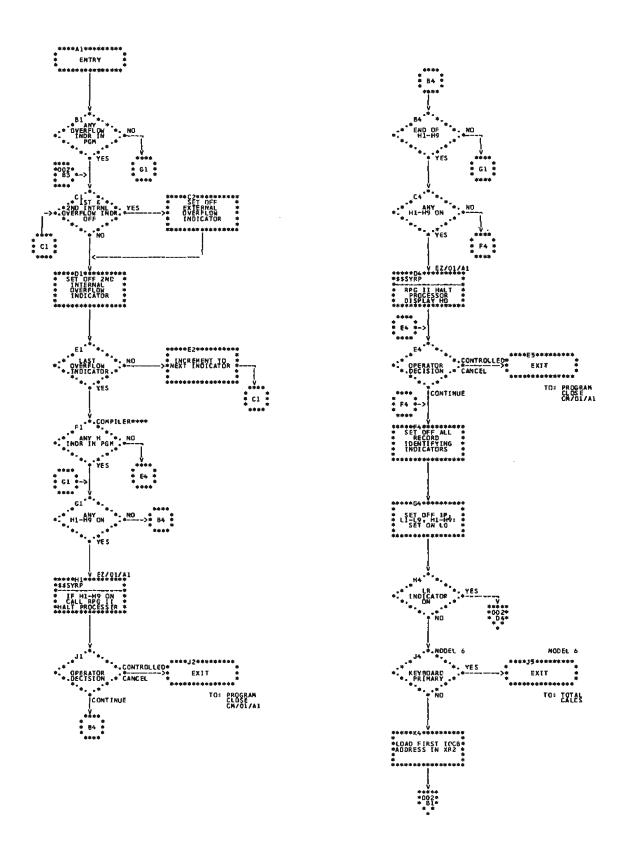


Chart CD. Output Fields and Records Code



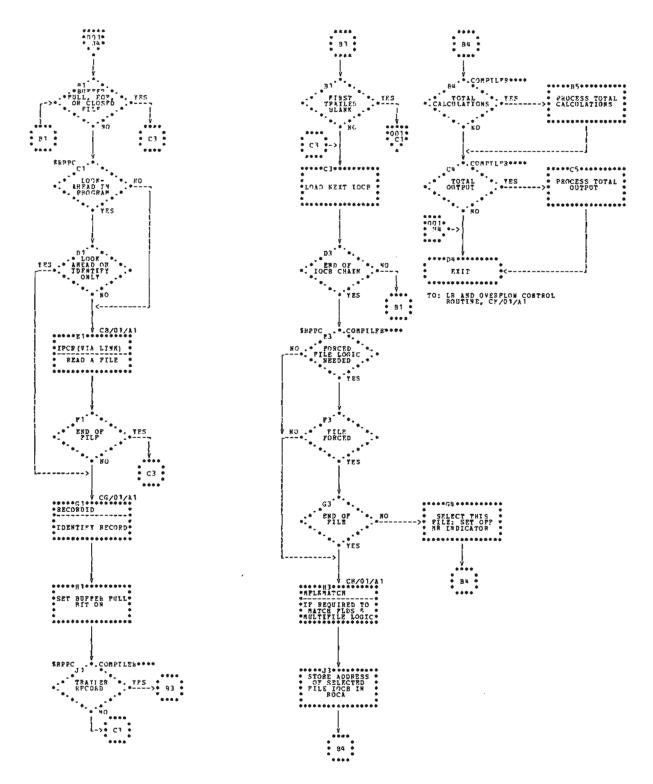


Chart CE (Part 2 of 2). Input Mainline

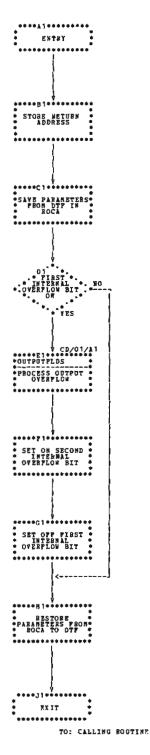


Chart CF. Fetch Overflow

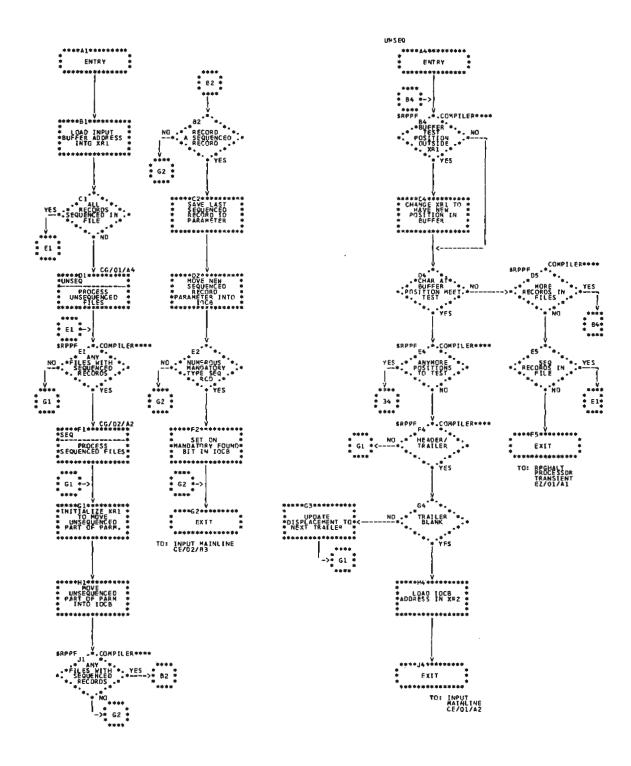
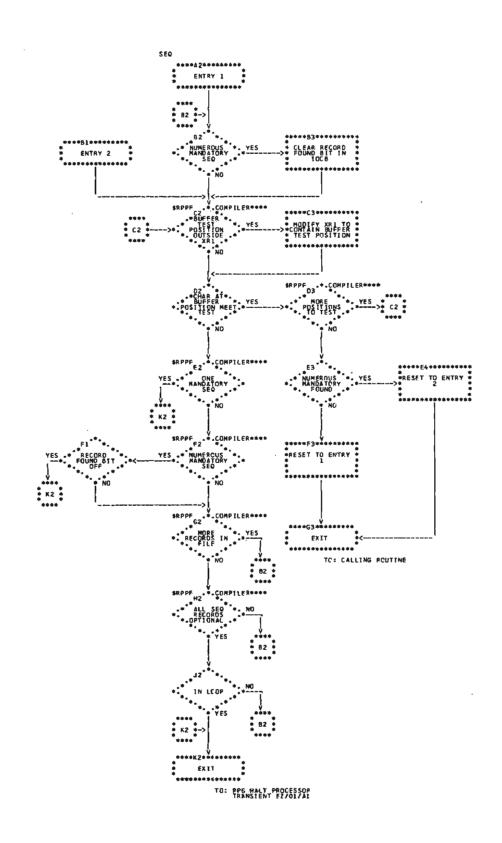
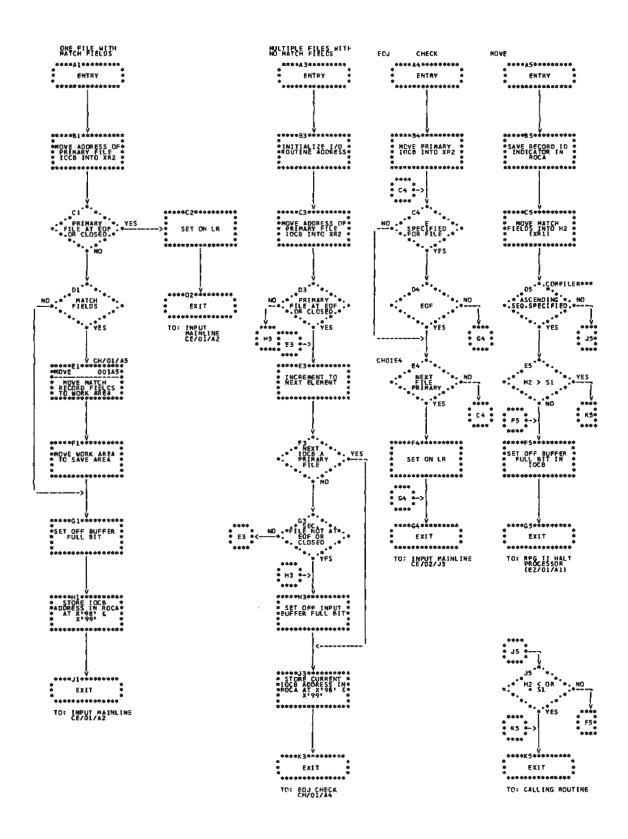


Chart CG (Part 1 of 2). Record ID





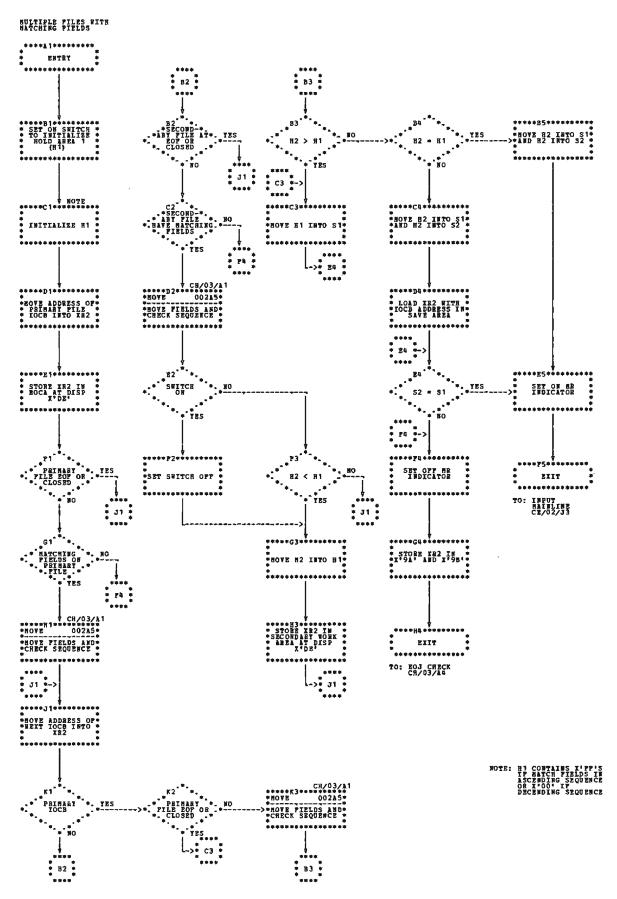


Chart CH (Part 2 of 2). Multifile Logic

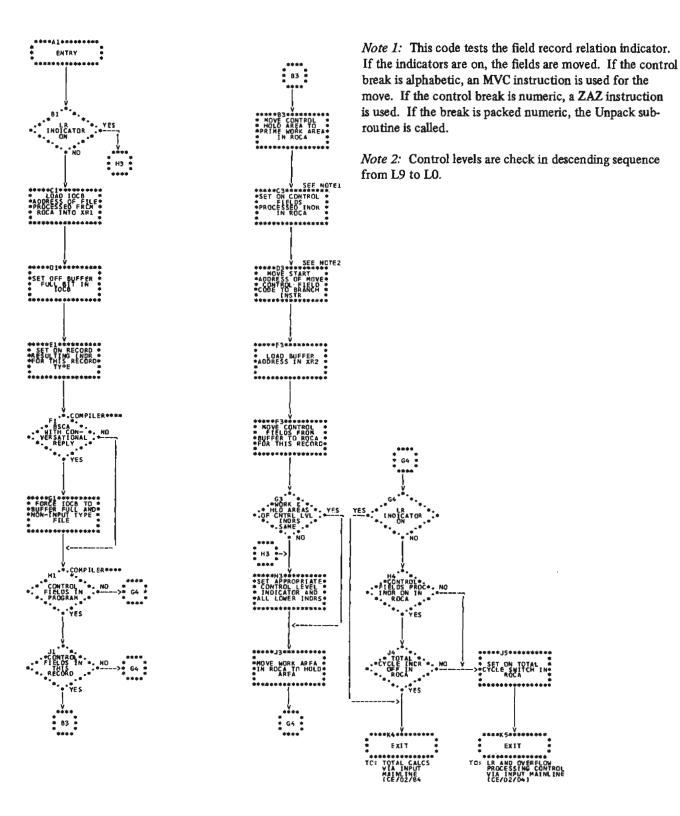


Chart CI. Control Fields Logic and Move

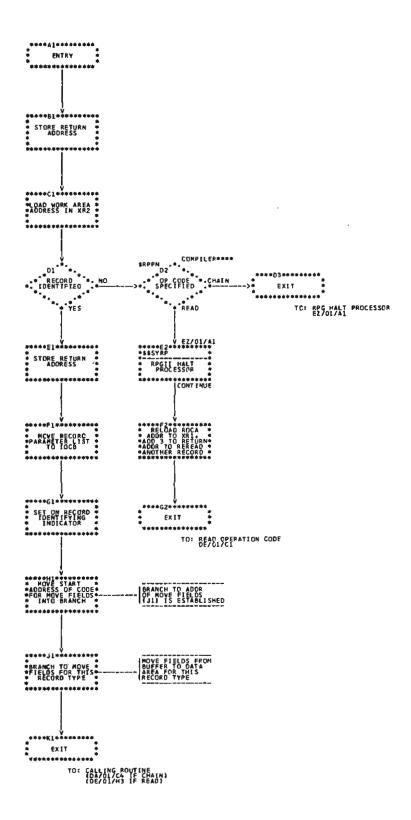


Chart CJ. Chain and Read

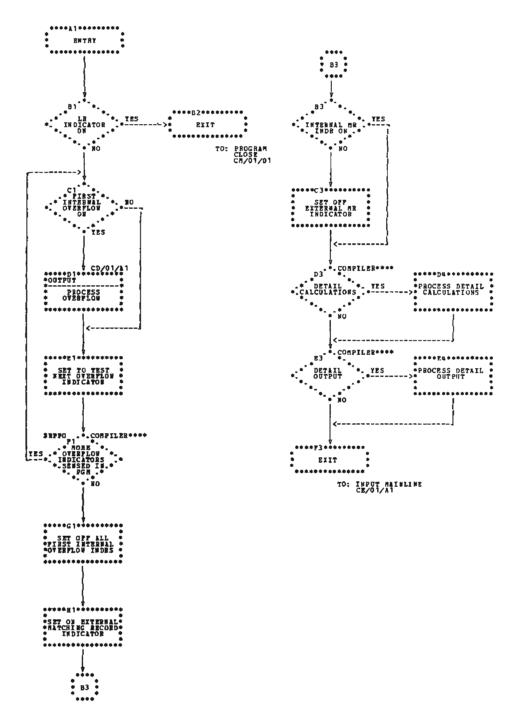


Chart CK. LR and Overflow Control Mainline

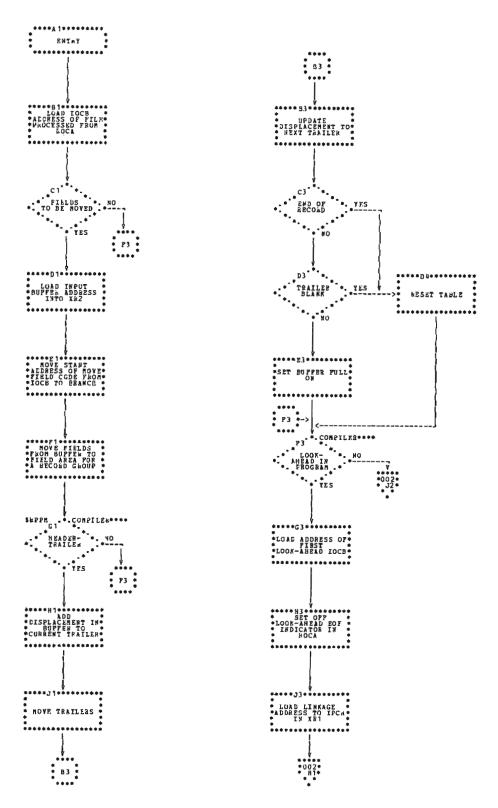


Chart CL (Part 1 of 2). Move Input Fields Mainline

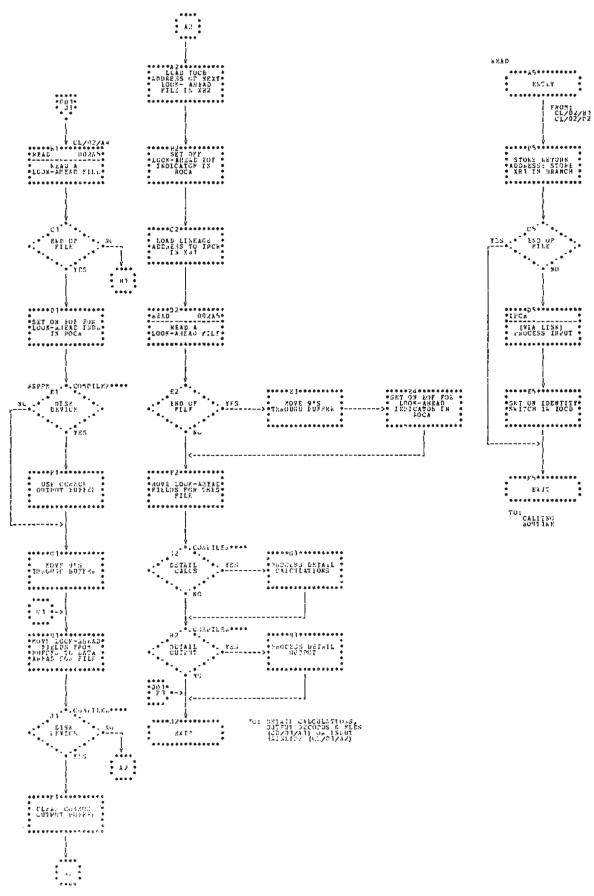


Chart CL (Part 2 of 2). Move Input Fields Mainline

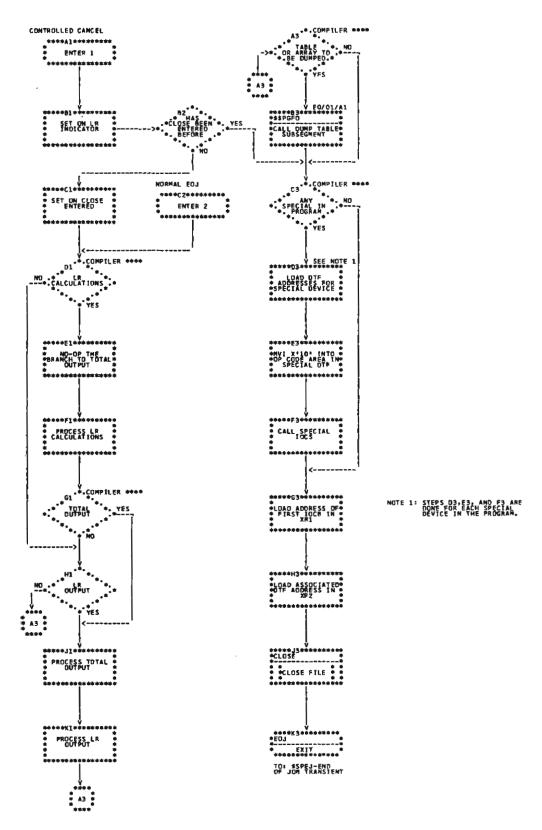


Chart CM. Program Close Mainline

CALCULATIONS OBJECT CODE

This section discusses the sequence of operations performed by the object code generated for each permisible 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 Calcula-	
	tion sheet)	
F2	Factor 2 (refers to heading on RPG II Calcula-	
	tion sheet)	
IAR	Instruction address register	
10CB	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 Cal-	
	culation 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:

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.
- 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:

$$\frac{7}{17.6534}$$

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:

- 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\(\neq\text{DF2.}\)]

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.
- 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:

- Print DSPLY.
- 2. Display contents of Factor 1 if specified.
- 3. Display contents of Result Field if specified.
- 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.
- 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 Sub-* routines).
- 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 address 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:

- Display Mode Place keyed field into main storage location and display on printer.
- Store Only Mode Place keyed field into main storage location (no display).
- 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
- 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:

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

- 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:

- 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):

- 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.
- 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 right-most 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 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:

- Load XR2 with a negative constant, 1 minus LRF.
- 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.
- Move a flag byte of 0 to byte 0 of ROCA requesting the time and date in decimal units.
- Place time and system date in bytes 1-12 of ROCA using Supervisor call with TIME of day RIB request (X'96').
- 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.

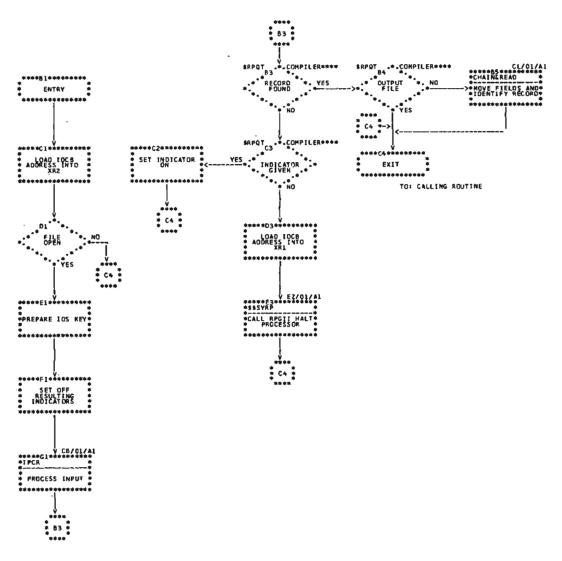


Chart DA. CHAIN Operation Code

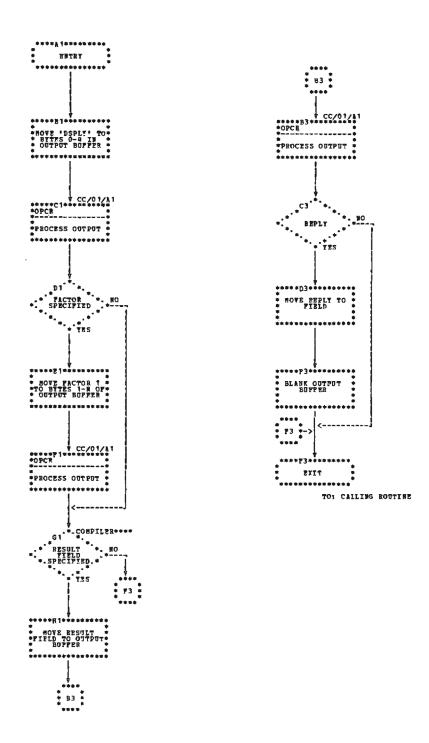


Chart DB. DSPLY Operation Code

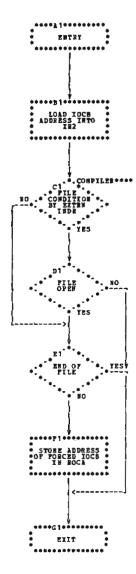


Chart DC. FORCE Operation Code

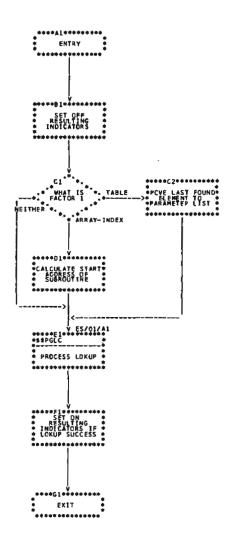


Chart DD. LOKUP Operation Code

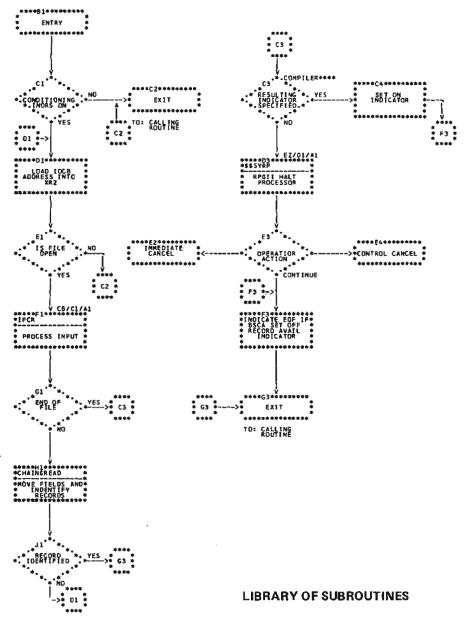


Chart DE. READ Operation Code

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.

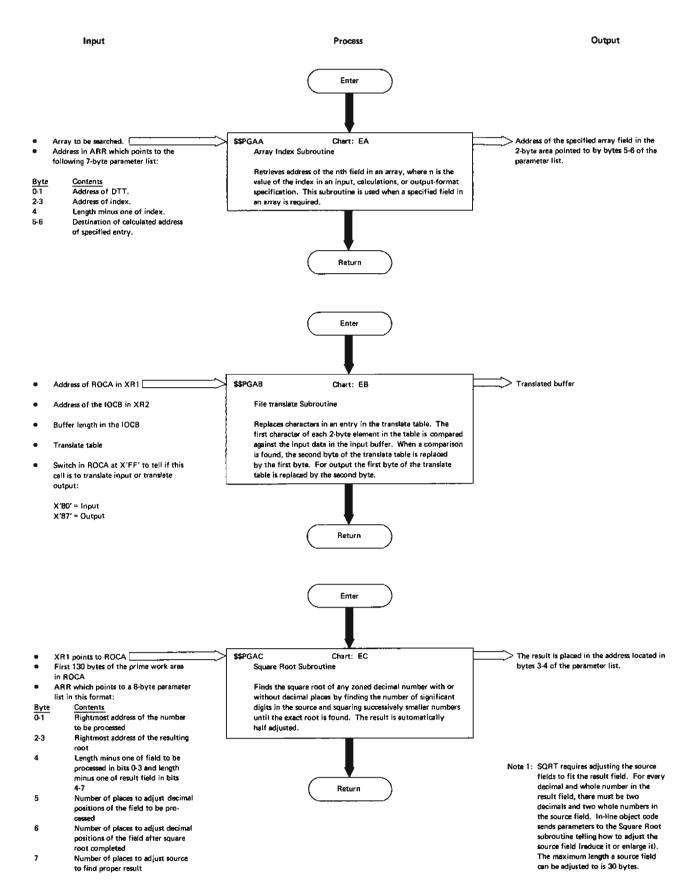


Figure 4-3 (Part 1 of 9). Library of Subroutines

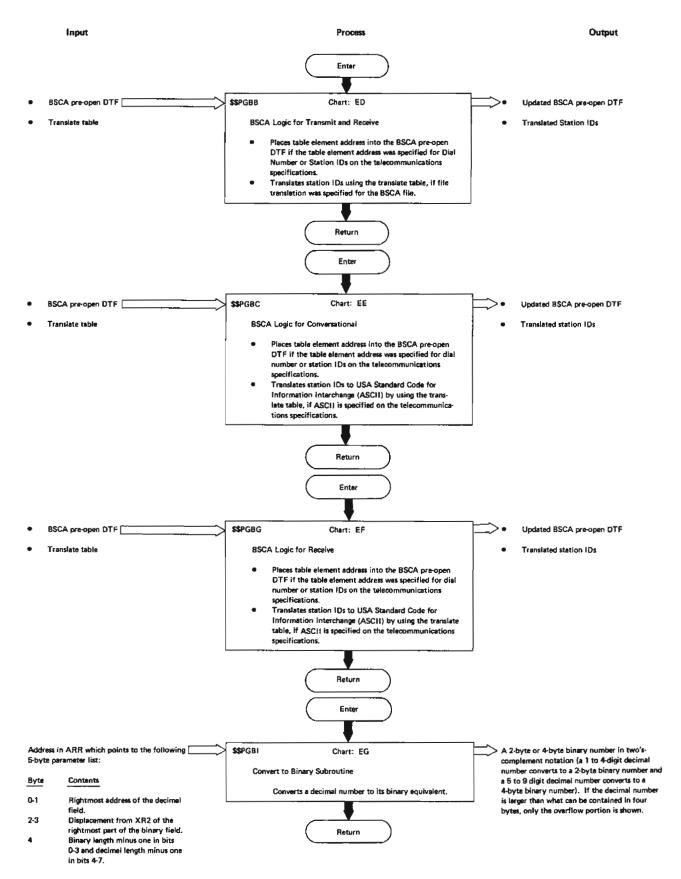


Figure 4-3 (Part 2 of 9). Library of Subroutines

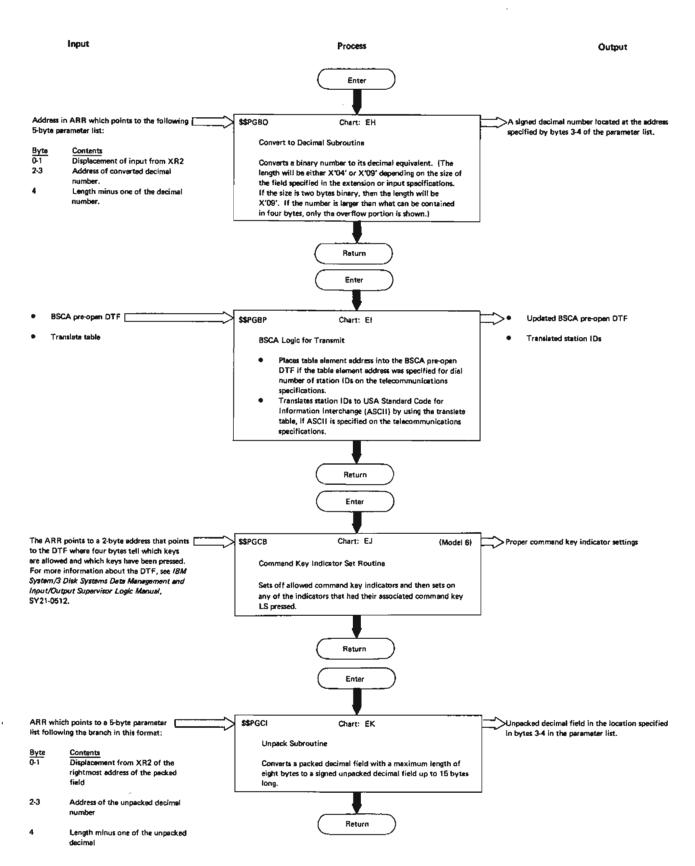


Figure 4-3 (Part 3 of 9). Library of Subroutines

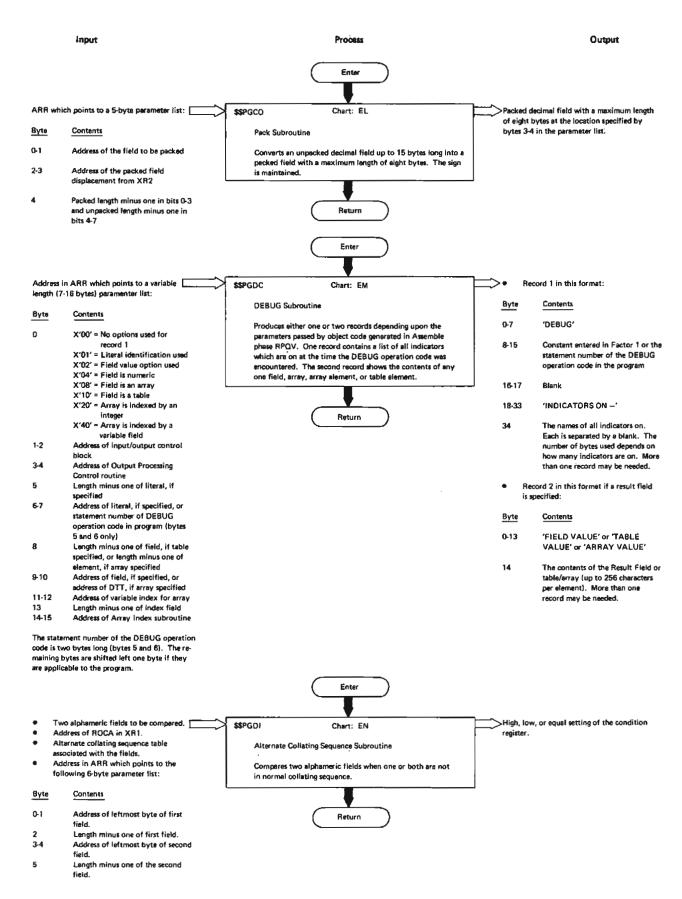
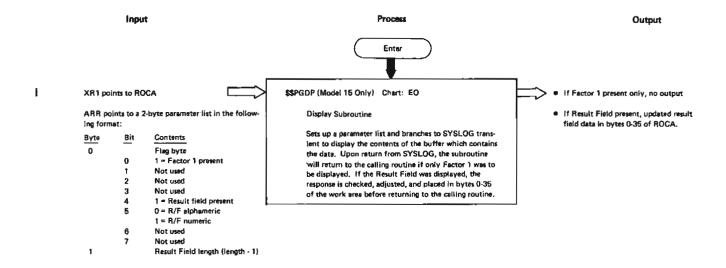


Figure 4-3 (Part 4 of 9). Library of Subroutines



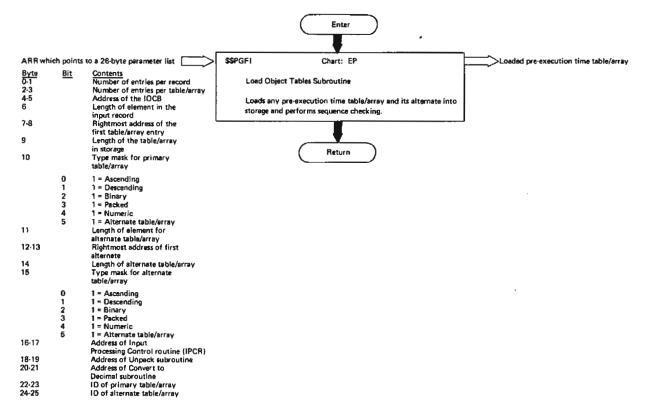


Figure 4-3 (Part 5 of 9). Library of Subroutines

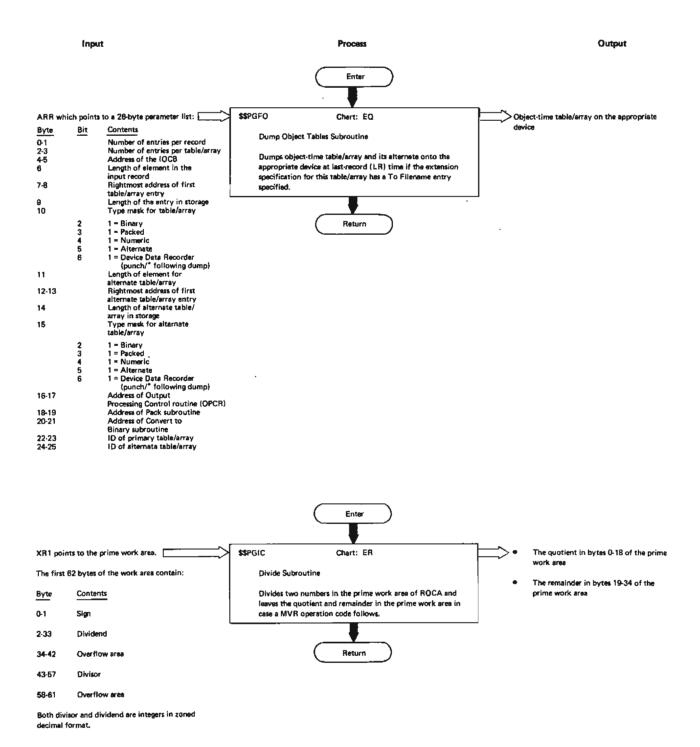


Figure 4-3 (Part 6 of 9). Library of Subroutines

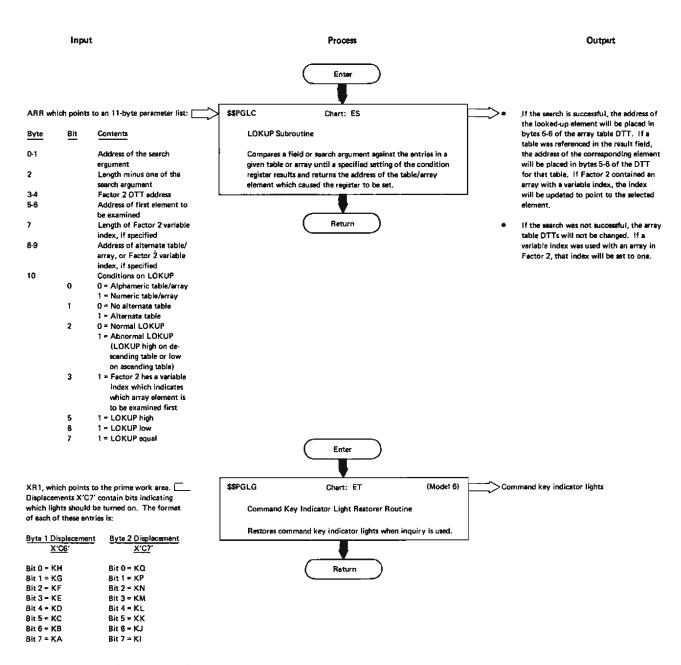
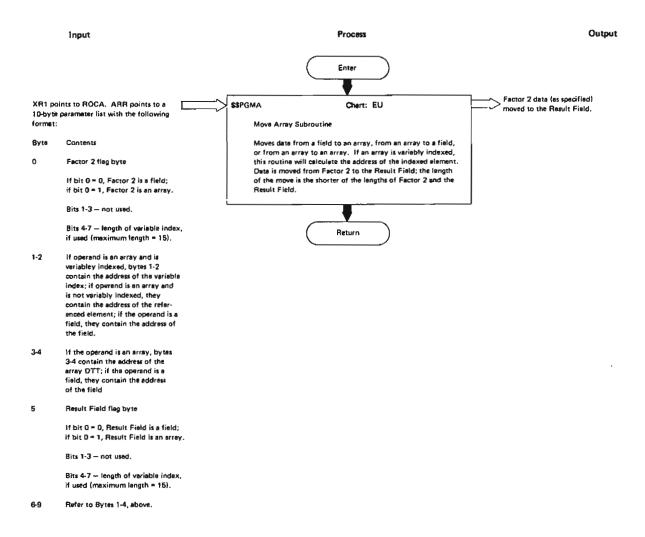


Figure 4-3 (Part 7 of 9). Library of Subroutines



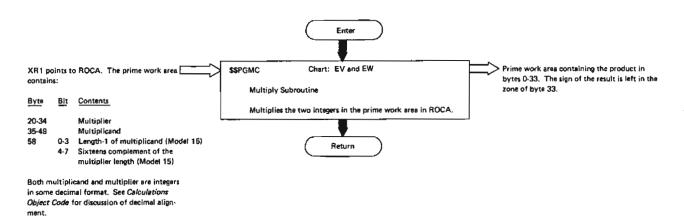


Figure 4-3 (Part 8 of 9), Library of Subroutines

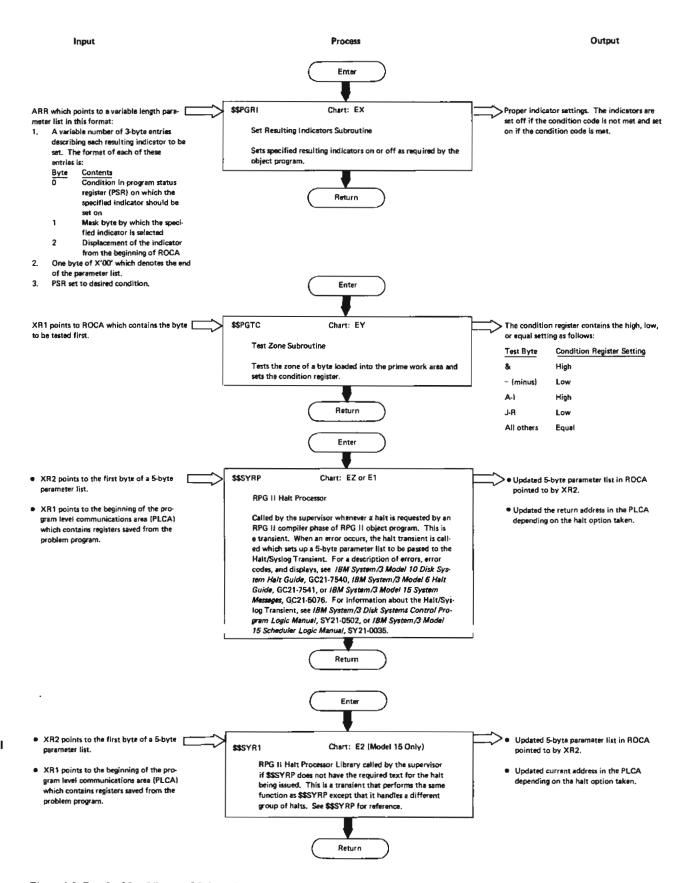


Figure 4-3 (Part 9 of 9). Library of Subroutines

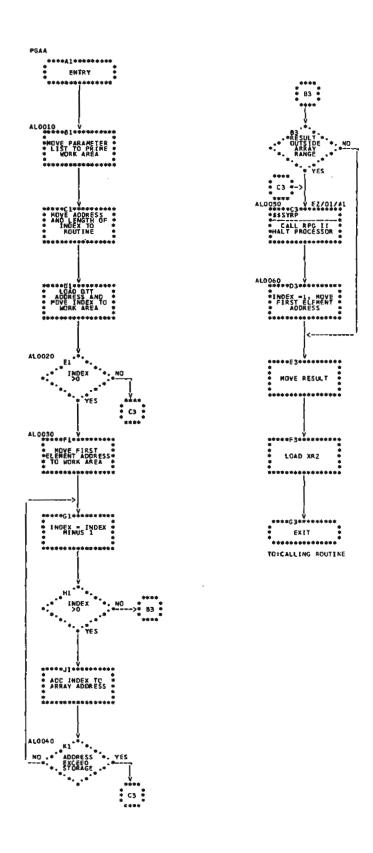


Chart EA. Array Index Subroutine

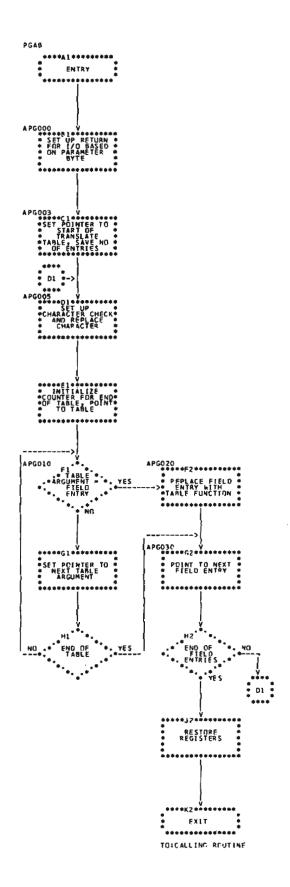


Chart EB. File Translate Subroutine

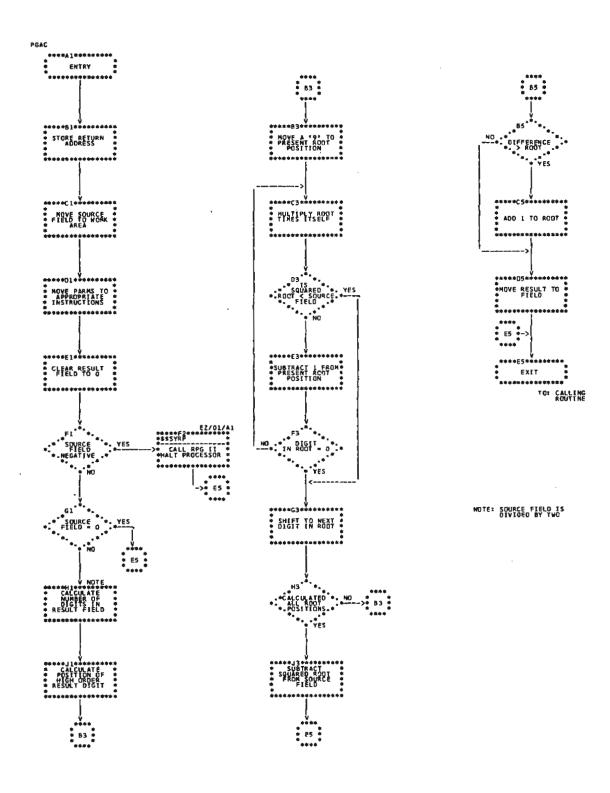


Chart EC. Square Root Subroutine

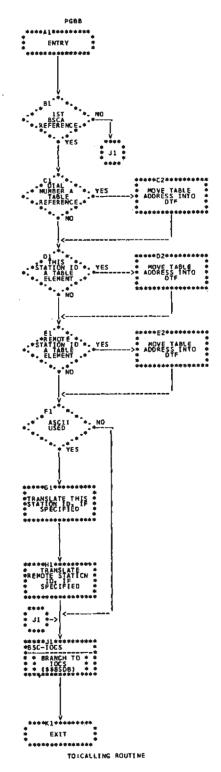


Chart ED. BSCA Logic for Transmit and Receive

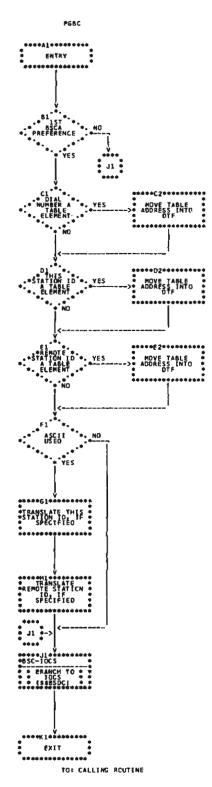


Chart EE. BSCA Logic for Conversational

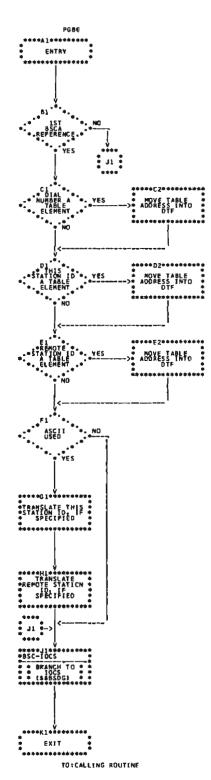


Chart EF. BSCA Logic for Receive

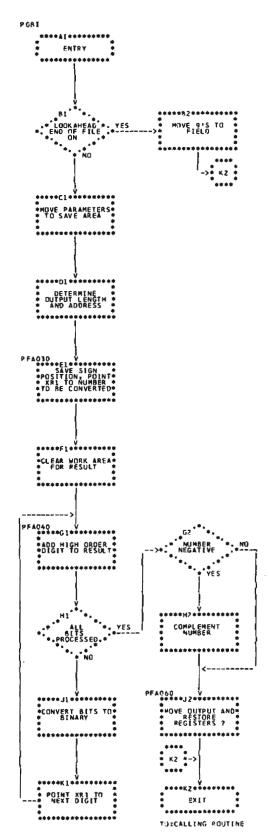


Chart EG. Convert to Binary Subroutine

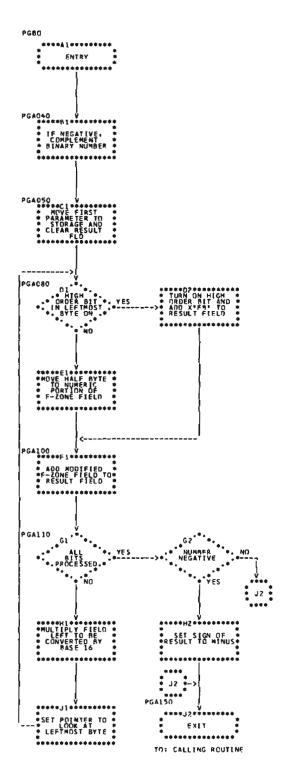


Chart EH. Convert to Decimal Subroutine

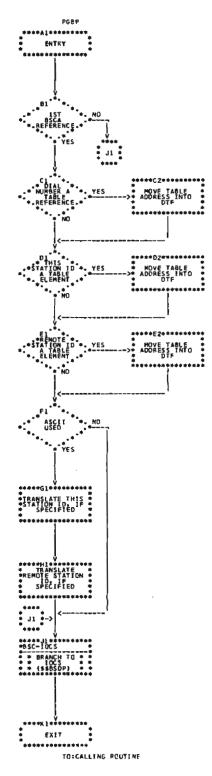


Chart El. BSCA Logic for Transmit

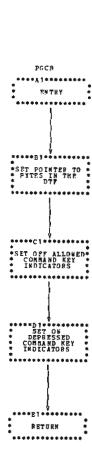


Chart EJ. Command Key Indicator Set Routine (Model 6)

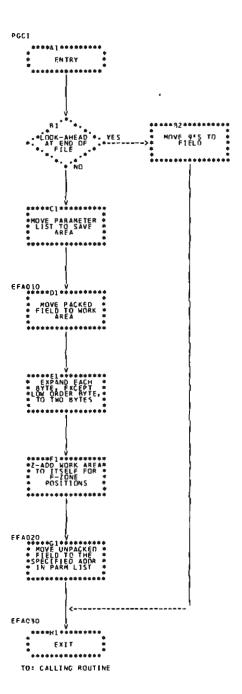


Chart EK. Unpack Subroutine

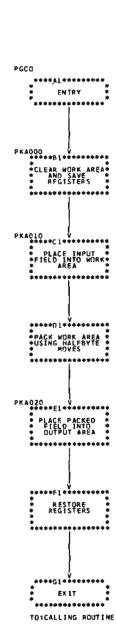


Chart EL. Pack Subroutine

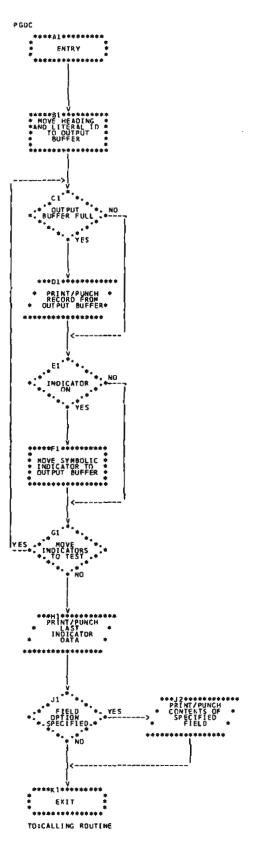


Chart EM. DEBUG Subroutine

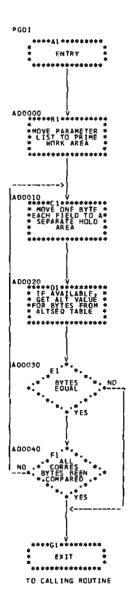


Chart EN. Alternate Collating Sequence Subroutine

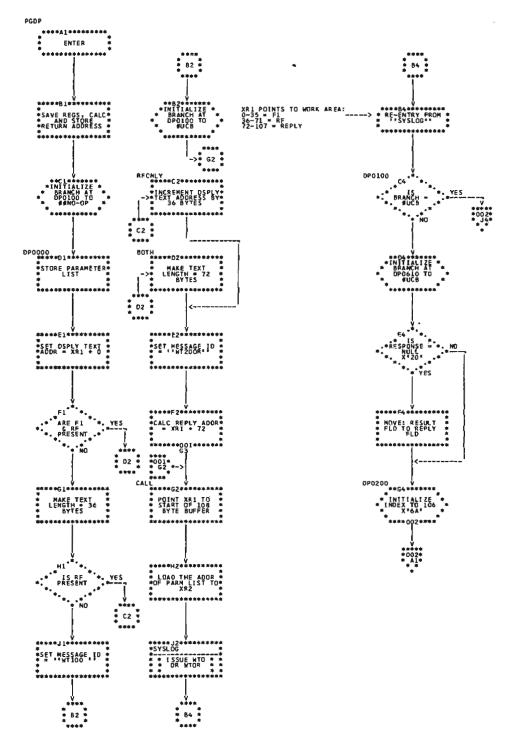


Chart EO (Part 1 of 2). Display Subroutine (Model 15)

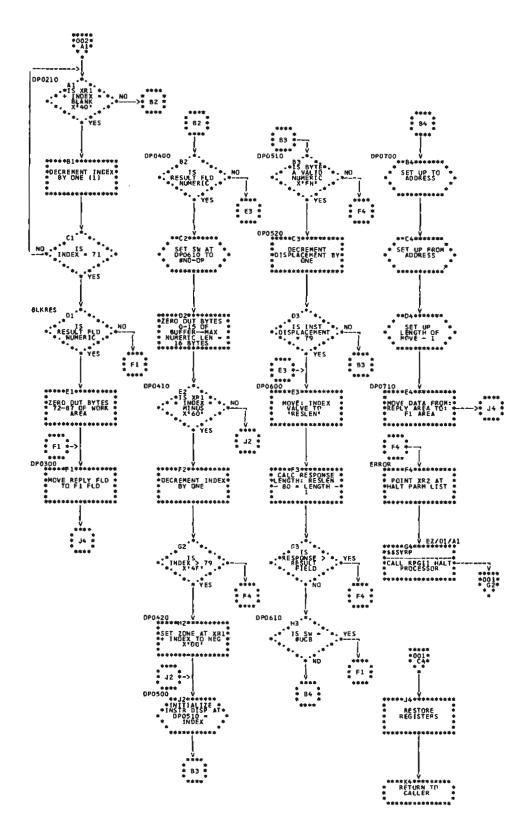


Chart EO (Part 2 of 2). Display Subroutine (Model 15)

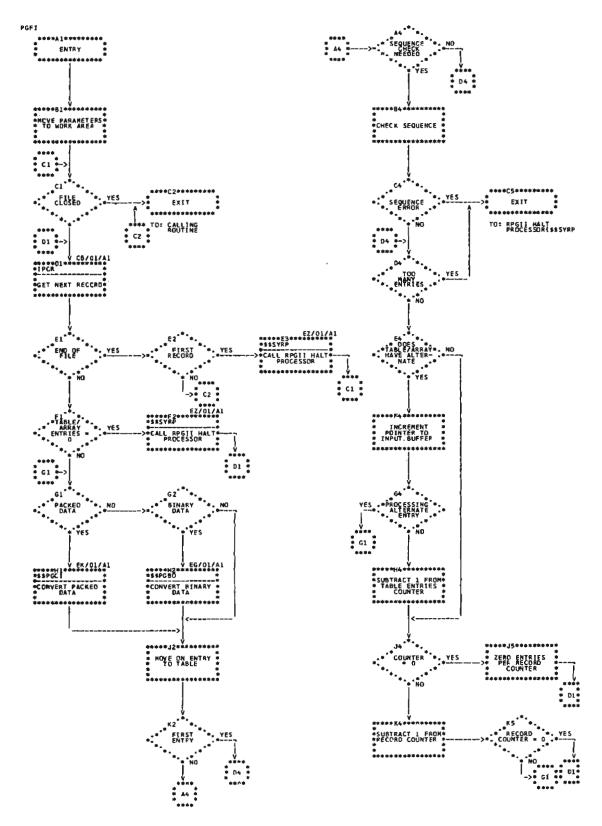


Chart EP. Load Object Tables Subroutine

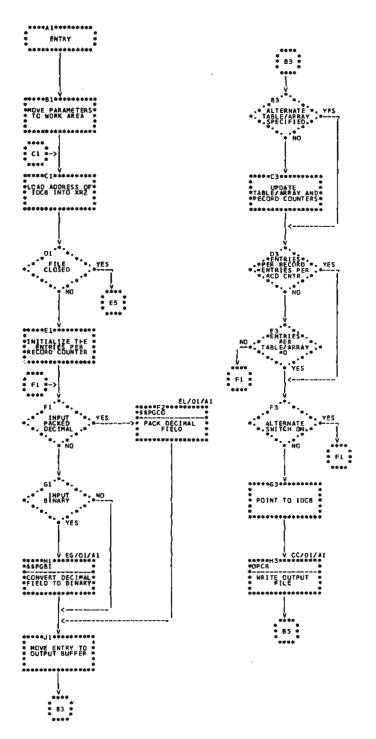


Chart EQ. Dump Object Table Subroutine

TOICALLING ROUTIME

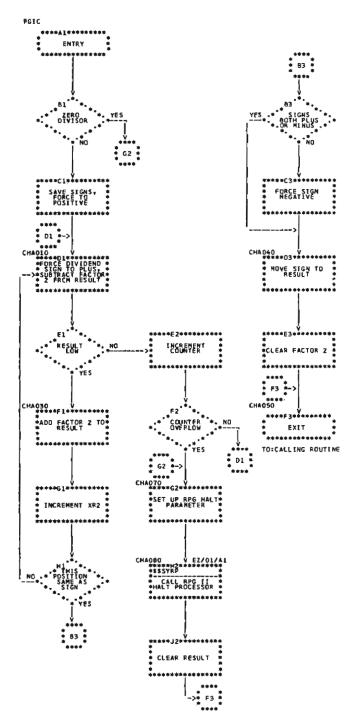


Chart ER. Divide Subroutine

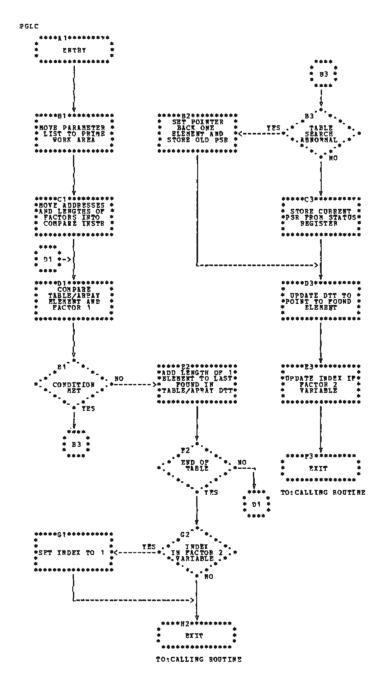


Chart ES. LOKUP Subroutine

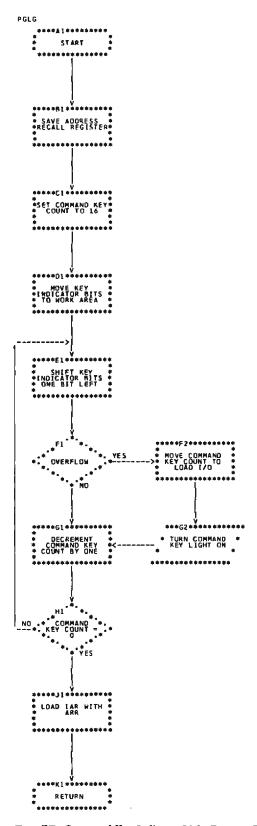


Chart ET. Command Key Indicator Light Restorer Routine (Model 6)

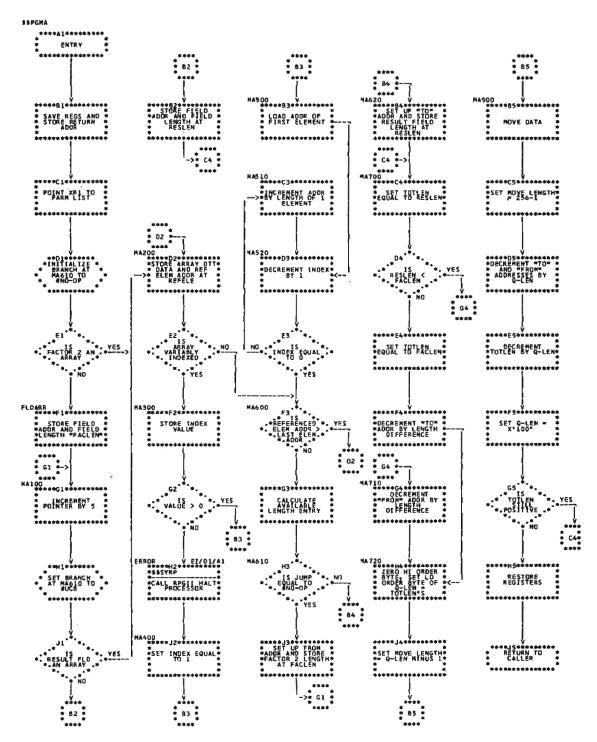


Chart EU. Move Array

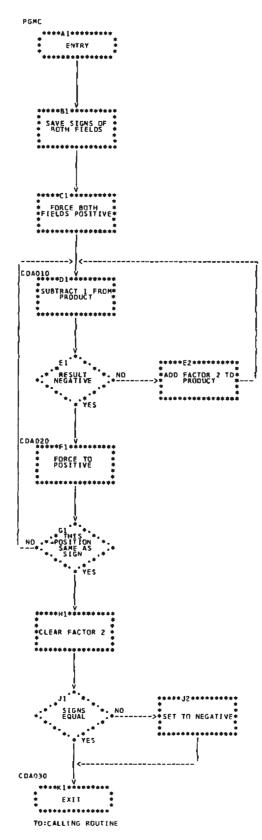


Chart EV. Multiply Subroutine (Models 6, 10, and 12)

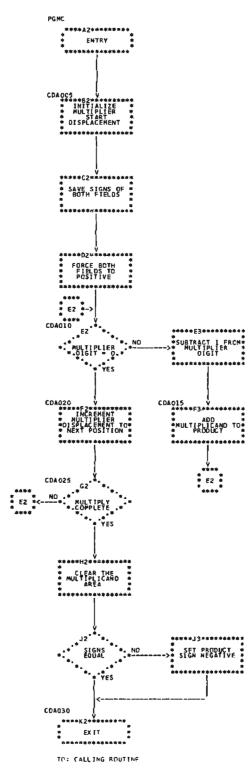


Chart EW. Multiply Subroutine (Model 15)

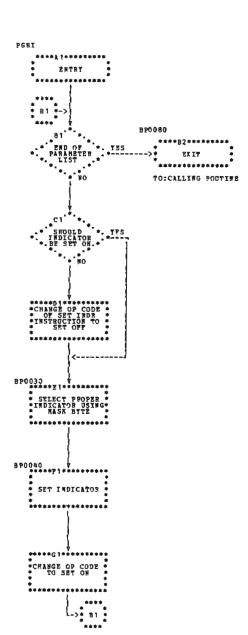


Chart EX. Set Resulting Indicators Subroutine

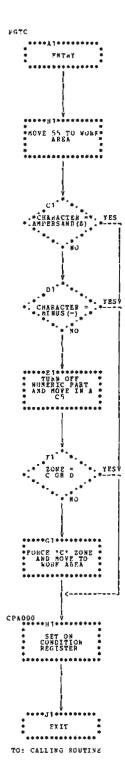


Chart EY. Test Zone Subroutine

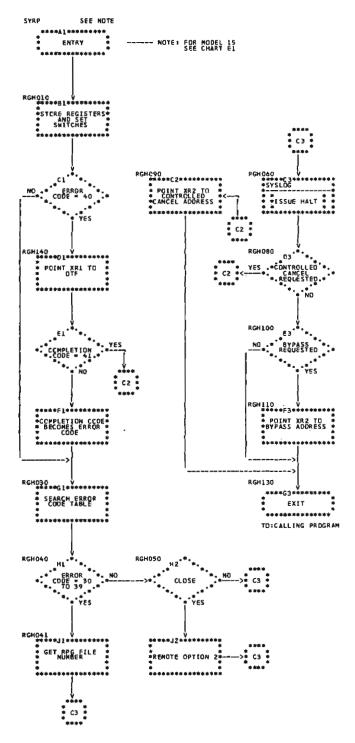


Chart EZ. RPG Halt Transient (Models 6, 10, and 12)

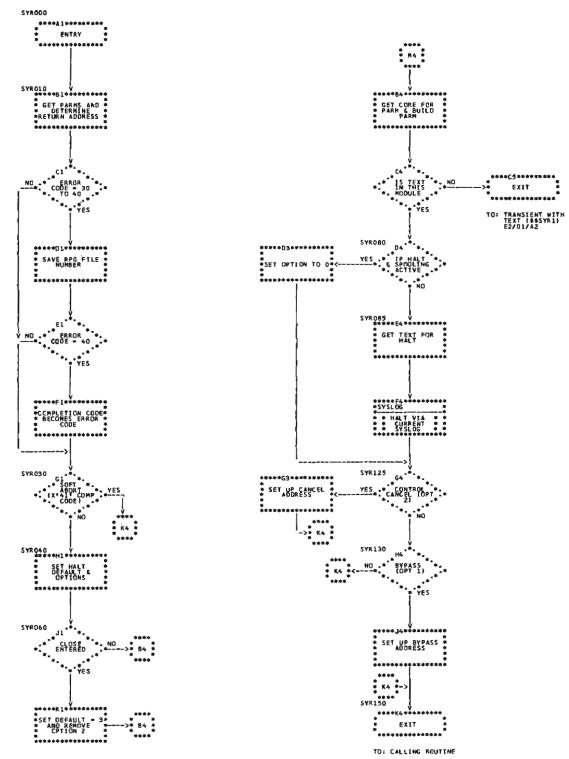


Chart E1. RPG II Halt Transient (Model 15)

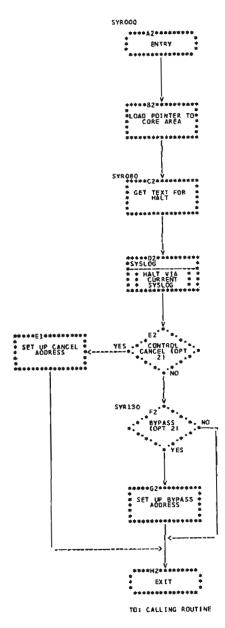


Chart E2. RPG II Halt Transient Library 1 (Model 15)

DATA AREAS		Byte	Contents		
areas used by the object	ne layout and contents of data program. Although lengths and vary with the source program,	X'90'-X'97'	Constants used by many routines: X'40FFFF0000010002'		
	ixed lengths of data areas are	X'98'-X'99'	Address of first IOCB in the IOCB chain		
Reserved Object Comm	unications Area (ROCA)	X'9A'-X'9B'	Address of IOCB of file currently being processed		
	amunications area is the first 256 ent for every RPG II program.	X'9C'-X'9D'	Address of IOCB of a forced file		
	ctually generated since most of it	X'9E'-X'9F'	Address of first table load DTF		
Prime Work Area		X'A0'-X'A1'	1P save area containing the error restart address for the First Page Output routine		
TIME HOIK ALGO			par xoulde		
	a is located in the beginning of '8F'. The following blocks of	X'A2'-X'A7'	UDATE in user-specified format		
object code use the prim	_	X'A2'-X'A3'	UMONTH (UDAY if using European method of notation)		
Object Code	Bytes Used				
TESTZ	X'00'-X03'	X'A4'-X'A5'	UDAY (UMONTH if using European method of notation)		
Z-ADD ADD	X'00'-X'0F' X'00'-X'19'	X'A6'-X'A7'	UYEAR		
Z-SUB SUB	X'00'-X'19' X'00'-X'19'	X'A8'-X'A9'	Month and day of compile		
Multiply	X'00'-X'32' (Models 6, 10, and 12) X'00'-X'3A' (Model 15)	X'AA'-X'AB'	Time of compile (HHMM Model 15		
Divide	X'00-X'3D'		only)		
SQRT	X'00'-X'3D'	X'AC'-X'AF'	Branch to controlled cancel		
LOKUP	X'00'-X'3D'	A AC · A AI	Bianch to controlled cancer		
Control fields Matching fields	Xʻ00'-Xʻ8F' Xʻ00'-Xʻ8F' Xʻ07'-Xʻ0F'	X'B0'-X'B3'	Branch to Input Mainline		
Chain packed keys MVR	X'1 B'-X'2A'	X'B4'-X'B8'	Halt parameters used by the RPG II		
Table load	X'60'-X'83'		Halt Processor		
*PRINT	X'00'-X'8F'		X'B4' = X'43'		
TIME	X'00'-X'0C' (Model 15 only)		X'B5' = X'40'		
Constant Area			X'B6' - X'B7' = Console stick-light mask (see Halt/Syslog, IBM System/3 Disk Systems System Control Pro-		
	rea follows the prime work area in ated from X'90' to X'C1'. Com-		gram Logic Manual, SY21-0502) X'B8' = Acceptable restart options		
piler phase \$RPGH builds the constant area and phases \$RPPC and \$RPPO make additions to it. The constant area contains:		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		

I

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				Hex Byte	e Mask			
from XR1	80	40	20	10	08	04	02	01
Ç2	H4	нз	H2	Н1		MR (Int.)	MR (Ex.)	1P
C3	L1	LO	LR	Н9	Н8	H7	H6	H5
C4	L9	L8	L7	L6	L5	L4	L3	L2
C5	U1	U2	U3	U 4	U5	_U6	U7	<u>U8</u>
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
СВ	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
D 0	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	OG 1st	OF 1st	OE 1st	OD 1st	OC 1st	OB 1st	OA 1st
	Int.	Int.	Int.	Int.	Int.	Int.	Int.	Int.
D8	OV 2nd	OG 2nd	OF 2nd	OE 2nd	OD 2nd	OC 2nd	OB 2nd	OA 2nd
	Int.	Int.	lnt.	Int.	Int.	Int.	Int,	Int.
D9	Total cycle switch	Control fields processed	Overflow being processed	EOF on look- ahead	Close has been entered	* * RESER	RVED	

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:

Bytes Used
X'DD'-X'EC'
X'B9'-X'D0'
X'DA'-X'EA'
X'DD'-X'E2'
X'E7'-X'FF'
X'E8'-X'FF'
X'EA'-X'ED'
X'EA'-X'FF'
X'EC'-X'FF'
X'ED'-X'FF'
X'F1'-X'FF'
X'F3'-X'FF'
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, \$Y21-0512, and/or IBM System/3 Model 15 System Data Areas and Diagnostic Aids, \$Y21-0032.

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 'M1 M3 M4'.

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 outputformat 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)

Code	Meaning
4B*	Invalid ASCII code
4D*	Invalid call by user
4E*	Programmer lost communication

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

Meaning
Normal completion
Controlled cancel requested
Updating a record not found
Conversational reply requested (BSC)
Overflow
Invalid ASCII code
Invalid call by user
Programmer lost communication
Key changed
Duplicate add
Add or load out of sequence
Key too low or too high for indexed random multivolume online file.
Sequence ADD to multivolume indexed file because high key missing from previous volume.
End of extent
Key too high for last volume of multi-volume indexed file.
mpletion codes are sent to the RPG Halt 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 1OCB. 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	8-9	File relation address ('from' or 'to' IOCB addresses for record address files or tables)
0	0	1 = End of file has occurred		ines of tables)
	1 2	1 = File not open1 = Identify look-ahead file	10	Overflow indicator mask
	3	1 = Non-input control file (not primary or secondary)	11-12	Record length
	4 6	1 = Translate file 1 = End-of-file specified on file	13-14	Address of output work area
	7	description specifications 1 = Buffer full (does not need to be	15	Sequence number (in binary)
		read from this cycle)	16	External indicator

Byte

1

2-3

4-5

6-7

Bit

0

1

2

3

7

4-6

Contents

1 = BSCA 'last' file

1 = Combined file

Record address type

110 = Record number

IOCB chain address

Translate table address

DTF address

1 = Record address file

1 = Limits file

1 = Update file

000 = Data base 010 = Key

100 = Record ID

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

Contents

Byte

١

1

Bit

Вуге	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
	5-7	000 = No select 010 = Print 4 tiers 100 = Stacker 4 101 = Stacker 1 110 = Stacker 2 111 = Stacker 3
	0-7	X'00' = No select X'01' = Stacker 1 X'02' = Stacker 2 X'03' = Stacker 3 X'04' = Stacker 4 X'05' = Stacker 5 MFCM only

Byte	Bit	Contents		
29-30		Address of Move Input Fields, for this record type	code	
31-32		Address of Control Fields Mov code for this record type	⁄e	
33-34		Address of Matching Records moves code for this record type		
35-36		Address of next numeric seque checking code for this file	ence	
37		Numeric sequence information X'01' 1 = Numerous 0 = One X'02' 1 = Mandatory record 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
- Input Records
- 4. Total Calculations
- 5. Total Output
- LR and Overflow Control
- 7. Input Fields
- 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
10	Total Calculations Mainline
01	Total Output Mainline
01	LR and Overflow Control Mainline
01	Input Processing Control routine (IPCR) Subsegment

Keyboard** (Display or Store Mode)	Address of field					Tractor 1	Tractor 2		Numeric field	Blind key mode	Display mode		Field length	Field light mask	Allowed command key mask Byte 1,Bit 0=Com. Key 16	Bit 7=Com. Key 9 Byte 2,Bit 0=Com. Key 8 Sit 7=Com. Key 1	
Bit						0	-	0 B	4	2	9	7					
Byte	0 -		,	7				ო				4	LO.	ဖ	7		
Kayboard (Manual Mode)	Tractor 1	Tractor 2		Turn off command key lights				Manual mode	Element return	Field light	mask	Allowed command key mask	Byte 1,Bit 0=Com. Key 16 {	Byte 2,Bit 7=Com. Key 9 Byte 2,Bit 0=Com. Key 8 Sit 7=Com. Key 1	Address of	tab table	
Console	Input*	Output*					Accept characters		Number of bytes to be printed	Number of bytes	WTOR*	Zone: Space before	Digit: Space after				
CRT		Operation code							Blank screen	Blank	before	Blank	after				
Ledger	Operation code								Space after; X'80'= eject							,	
Printer		Print							Skip before	Space	before	Skip	after	Space after	Start print position	Number of bytes to be printed	Position after print
Data Recorder	Read		Punch	-													
Disk	Get	Put/Add	Update					Do not clear the output buffer									
Bit	0	-	7	ო	4	വ	9	7									
Byte				0	·				-	2		~	,	4	rs	9	7

Figure 4-5. Model 6 Data Management Parameters

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

*WTOR = Write to operator with reply specified.

Byte	Bit	MFCU	Disk	Printer	Console	Special	BSCA
	0	Read	Get		Input*	Input	Input
	1	Print	Put/Add	Print	Output*	Output	Output
	2	Punch	Update				
	3	Move			"I"-type inquiry program		
0	4						
	5						
	6		• -		Accept characters only		
	7	Do not clear o	utput buffer				
1	2	Print 4	•	Skip	Number of bytes to		
	5-7	Stacker select		before	be printed		
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	
01	Record Identification processing
	Multi-file Logic processing
01	Control Fields processing
03	Alternate Collating Sequence sub-
	routine
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
	Divide subroutine
25	
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 over-

lay 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:

- Flags the segment or subsegment that has the lowest priority and has not previously been flagged as a main overlay or suboverlay.
- Determines the length of the main overlay or suboverlay areas and adds the length to the total program.
- 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.
- 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
В	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	1 = Overlay in storage now1 = 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.

- Locate the address of the Overlay Fetch routine on the core usage map of the source listing (Figure 4-7, part 1).
- 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).

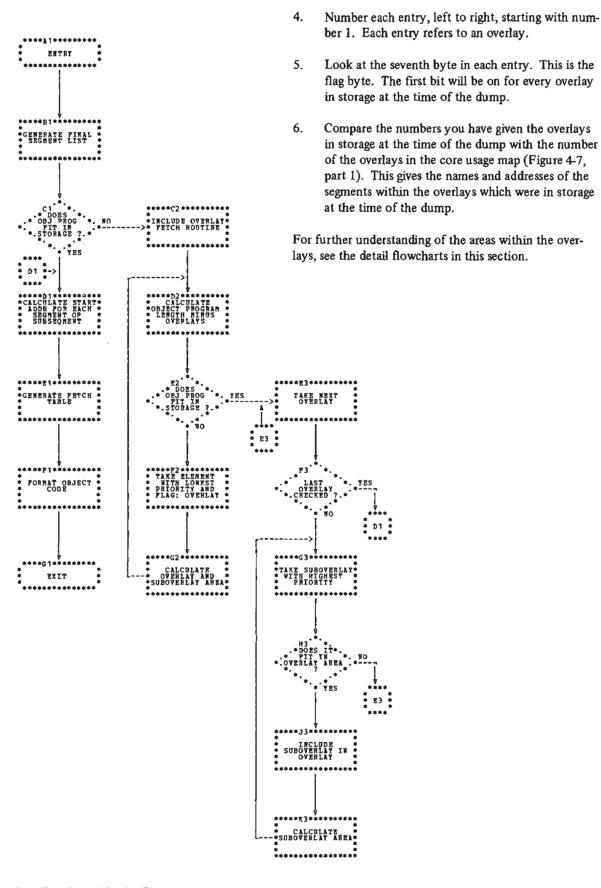


Chart FA. General Overlay Flow

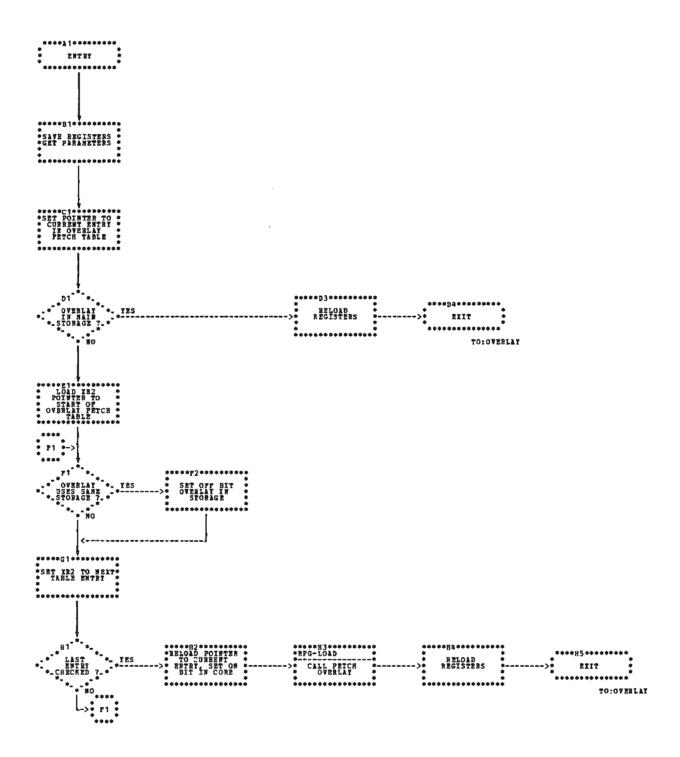


Chart FB. Overlay Fetch Routine

```
INDICATORS USED
         LR 01 02
RG 314 UNREFERENCED FIELD NAMES
 STMT# NAME DEC LGTH DISP
  0005 NODATA
                     001
                           01.00
  FIELD NAMES USED
 STMT# NAME DEC
                    LGTH
                           DISP
  0005 NODATA
                     001
                           0100
  OOOA COUNT
                     006
                           01.06
  0007 RECNER O
                     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
    ADDR
            OVERLAY
                     LENGTH
    0000
                      04F6
                              RGROOT
                                       ROOT
    10F6
                              ROSHBS
                                       OVERLAY FETCH ROUTINE
                      017E
                                       OVERLAY FETCH AREA
    1274
                      0600
                              RGSUBS
    12E6
            $$$001
                      008E
                              RGMAIN
                                       INPUT MAINLINE
    127C
            ‡$$001
                      006A
                              RGSUBS
                                       INPUT CTRL RTN
    1374
            ‡$$001
                      003A
                              RGSUBS
                                       RECORD ID
                                                        OVERLAY IN STORAGE
                                       CONTROL FIELDS
    13AE
            $$$001
                      0026
                              RGSUBS
                              RGSUBS
    1274
            $$$001
                     0008
                                       SUBSEC
                                       SYSTEM SUBR
            $$$001
                      0027
                              SSBDMC
    1282
            #$$002
                      8000
                              RGMAIN
                                       TOTAL CALCS
    1274
            #$$002
                      000E
                              RGSUBS
                                       CONSTANTS
    1274
            $$$003
                      0022
                              RGMAIN
                                       INPUT FIELDS
    133D
            $$$004
                     0048
                              RGMAIN
                                       DETAIL CALCS
            $$$004
    132A
                     0005
                              RGSUBS
                                       CONSTANTS
    132F
            $$$004
                     000E
                              RGSUBS
                                       CONSTANTS
    1280
            $$$004
                     00AA
                              RGSUBS
                                       OUTPUT CTRL RTN
            #$$004
                     0043
                              $$PGRI
                                       RESET RESULTING INDR
    1385
                              RGSUBS
                                       EXCEPTION
            #$$004
                     0031
    1274
            ~$$004
                     000C
                              RGSUBS
                                       SUBSEG
                                      DISK INDEXED OUTPUT
                     005F
                              SSIGUT
            #$$004
                                      SYSTEM SUBR
            $$$004
                     0094
                              SSSRBI
            $$$004
                     0026
                              SSSRUA
                                       SYSTEM SUBR
            $$$004
                     007£
                              $$SRBR
                                      SYSTEM SUBR
            $$$004
                     001C
                              $$SRDF
                                       SYSTEM SUBR
                              $$SRTC
            $$$004
                     001C
                                      SYSTEM SUBR
            $$$004
                     002F
                              SSSRBP
                                       SYSTEM SUBR
            $$$004
                     0019
                              $$SRCR
                                       SYSTEM SUBR
            $$$004
                     0015
                              $$SRPD
                                       SYSTEM SUBR
            #$$004
                     0081
                              SSSRMO
                                       SYSTEM SUBR
            ‡$$004
                     0043
                              $$SRSB
                                      SYSTEM SUBR
                              $$SRRD
            ÷$$004
                     0015
                                       SYSTEM SUBR
    1274
            $$$005
                              RGMAIN
                                      TOTAL OUTPUT
                     000B
                                      LR & OVERFLOW PROCESSING
    1347
            $$$006
                     0024
                              ROMAIN
    1280
            $$$006
                     00AA
                              RGSUBS
                                       DUTPUT CTRL RTN
                                      OVERFLOW SUBSEGMENT
    132A
            $$$006
                     001D
                              RGSUBS
    1274
            ÷$$006
                     0000
                              RGSUBS
                                       SUBSEG
                              SSBDMC
                                      SYSTEM SUBR
            $$$006
                     0027
    1438
            $$$007
                     0024
                              RGMAIN
                                      CLOSE
                                      OUTPUT CTRL RTN
    1280
            4$$007
                     AA00
                              RGSUBS
    1416
            $$$007
                     000E
                              RGSUBS
                                      CONSTANTS
                     OOEC
                              RGSUBS
                                      CONSTANTS
    132A
            ‡$$007
            $$$007
                     0076
                              RGSUBS
                                      LR PROCESSING
    1450
                                      SUBSEG
                              RGSUBS
    1274
            $$$007
                     000C
                                      SYSTEM SUBR
            $$$007
                     0027
                              SSBDMC
    1424
            $$$007
                     0014
                              RGSUBS
                                      LR CALCS
    1274
            $$$008
                     0072
                              RGMAIN
                                      OPEN
                              SAMPL1
                                      TOTAL CORE USAGE
                    03144
                 PROGRAM EXCEEDS CORE IN COLUMNS 12 THRU 14 OF HEADER CARD.
RG 999 W
RRABC 12
               ۸
```

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

0E60 00000000 00000000 000040	40 40404040 4040404	40404040 40404040	40404040	***********
OEBO 40404040 40404040 404040	40 40404040 4040404	40404040 40404040	40404040	*
OEAO 40404040 40404040 404040	40 40404040 4040404	40404040 40404040	40404040	* *
OECO 40404040 40404040 404040	08 40AB00C9 80000EE	00000000 00800013	OECAOB70	**
OEEO 00000004 00000000 000000	00 00000000 00000000	00000000 00000000	00000000	W
OFOO 00000000 00000000 000000	00 00000000 00040000	00000000 00000000	00000000	**
OF20 00000000 00000000 0000000	00 00000000 00000000	00000000 00000000	00000000	**
OFCO 00000000 00000000 000000	00 00000000 00000000	00000000 00000000	00000000	**
OFE0 C95C40A9 00C90000 0FF600	20 004001DA 0D700FE	OD70FFFF FFFFFFF	FFFFFFF	*I*I6
1000 FFFFFFF FFFFFFF FFFFFF	FF FFFFFFFF FFFFFFFF	FFFFFFFF FFFFFFF	FFFFFFF	**
10CO FFFFFFFF FFFFFFF FFFFFF	FF FFFFFFFF FFFFFFF	FFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFFFFFFF		**
10E0 FFFFFFF FFFFFFF FFFFFF	FF FFFFFFFF FFFFFFF			*B6*
1100 60740264 74081402 021100	6C 0270027C 0771B20	2 7376026E 5F007172	D0011FB8	*%\$
1120 8006F210 286C003A 067402	4E D20273B9 6006F210	03BB8006 E20207BD	FF00D001	*2*K2*
1140 39C20211 69BAB006 C08700	04 4075086A 75046CC	2 010C00C2 020DAD35	10116612	*.B* -OVERLAY FETCH TABLE
1160 E6070100 0012E600 01	02 127400E0 00110112	2 74006000 13011274	00400015	*#*
1180 05127400 60008301 127400	1-		01127400	**
11A0 40FF0000 00000000 000000	00 00003408 11600087	7 10F60712 74340811	60008710	**
END OF OVERLAY 11C0 F6061438 34081160 C08710		8710F601 12823408	11600087	*6*
11E0 10F60513 47340811 60C087	10 F606143F 34081160	C08710F6 04127434	081160C0	*.6*
1200 8710F603 133D3408 1160C0	B7 10F60614 47340811	60C08710 F606144B	34081160	*6*
1220 C08710F6 02127400 000000	00 00000000 00000000	00000000 00000000	00000000	*6*
1240 00000000 00000000 000000	00 00000000 00000000	00000000 00000000	0000000	**
1260 00000000 00000000 000000	00 00000000 00000000	C20113D4 3510127D	127E3408	*BM*.*
1280 12E53401 12A9E201 007502	05 BD1200F2 0106B81	03F2900E 9C000F1B	792118F2	*.V22*
1.2A0 10049C00 101CCOFF 13D46C	01 120DBD41 0EF28224	F28109BD 440EF282	0DF28112	*
12C0 C2020CB4 C0870004 8F407A	BO 00F28708 00030E0	1 12E512D1 BD400EB2	0200C201	*BV.JKB.*
12E0 0C00C087 13607B02 C97BF1	C2 789FC37B FFC47A46	C37820C3 C0101323	00011310	** I = 1B = .C = .D = CC*
1300 13717502 99B81000 F21012	OC 01135F13 751E0113	3 1097B9C1 00C01013	59B88000	**
1320 F290067A FFC47AE0 C3C087	13 AEC08711 CFC0871:	DA3C3213 58F2870D	3C301358	*2:.D:.C*
1340 B80118F2 9003RC00 1CBB01	00 E20100C2 020CB4C	8700048F 0074083C	C0871274	*2*
1360 B88000F2 1004C(87 1374BA	01 0075103C 1373127	\$ B50112B5 10143408	1381C201	*2B.*

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:

			, .	4110001	OC GIVINOW TOUGHTO
.			71	\$#ODOF	OD Overflow routine
Category	Mnemonic	-	71	\$#OEOF	OE Overflow routine
(decimal)	ID	Description	71	\$#OFOF	OF Overflow routine
			71	\$#OGOF	OG Overflow routine
00	GLOBAL	Buffers and IOBs	71	\$#OVOF	OV Overflow routine
00	COMMON	Fields, pre-execution and execu-	71	\$#FOVF	Fetch Overflow routine
		tion time table/array	93	\$#OPEN	Open Mainline
00	Object	ROCA, pre-execution and execu-	93	\$#CLOS	Close Mainline
	Program	tion time table/array DTTs	107	\$\$PGFI	Load Object Tables
	Name (from	col. 75-80 of the RPG Header Card)	107	\$\$PGFO	Dump Object Tables
00	\$#RT02	IOCBs, DTFs, DTTs and compile	107	\$#LROT	LR Output
		time table/array	107	\$#LRC	LR Calculations
00	\$#MISC	Edit codes, SETLL code, etc	126	\$#DOUT	Detail Output Mainline
00	\$#IPCR	Input Processing Control routine	126	\$#INPT	Input Mainline
00	\$#OPCR	Output Processing Control routine	126	\$#TCAL	Total Calculations Mainline
00	\$#CON0-F	Literals, constants, edit patterns	126	\$#TOUT	Total Output Mainline
		and parameters	126	\$#LROF	LR and Overflow Processing
00	\$\$PGAB	File Translate routine			Mainline
00	\$\$PGBB	BSCA logic for transmit/receive	126	\$#IFLD	Input Fields Mainline
00	\$\$PGBC	BSCA logic for conversational	126	\$#DCAL	Detail Calculations Mainline
00	\$\$PGBG	BSCA logic for receive	126	\$#RCID	Record ID routine
00	\$\$PGBP	BSCA logic for transmit	126	\$#MFLG	Multi-File Logic routine
00	\$\$PGRB	BSCA logic for transmit/receive	126	\$#CFLD	Control Fields Processing routine
		(2770/2780)		* == ==	
00	\$\$PGRG	BSCA logic for recieve (2770/	*The fire	et five colonietic	on subroutines in the source code
	,	2780)			
00	\$\$PGRP	BSCA logic for transmit (2770/			28 through 32 respectively. All
		2780)	other s	noton filles ale a	ssigned category 33.

00

10

11

15

28

28

28

29

35

39

41

41

44

45

47

48

55

55

57

57

57

71

71

71

28-33*

\$\$PGDC

\$\$PGDI

\$\$PGRI

\$\$PGAA

\$#IH01-19

\$#OH01-14

\$#SR00-FF

\$#MF01-FF

\$\$PGMC

\$#EXPT

\$\$PGLC

\$\$PGCO

\$\$PGCI

\$\$PGIC

\$\$PGDP

\$\$PGTC

\$\$PGMA

\$\$PGBI

\$\$PGBO

\$#RAFL

\$\$PGAC

\$#OAOF

\$#OBOF

\$#OCOF

\$#CHN0-F

DEBUG Routine

Array Index routine

routine

number)

number)

Alternate Collating Sequence

Set Resulting Indicator routine

Input hook (number = file

Output hook (number = file

Exception Output Segment

Move Output Field for OR lines

Calculation subroutines

Multiply subroutine

LOKUP routine

Unpack routine

Divide routine

Display routine

TESTZ routine

MOVEA routine

Convert to Binary routine

Convert to Decimal routine

RA File Processing routine

Chain Code Subsegment

Square Root routine

OA Overflow routine

OB Overflow routine

OC Overflow routine

Pack routine

source code ctively. All

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**************** SAMPLI SAMPL 1 0103 F* 0104 F# THIS PROGRAM --SARPL 1 SARPE 1 0105 FX 1. LOADS 100 RECORDS TO AN INDEXED FILE. 0106 F* SAMPL 1 0107 F* SAMPL 1 0108 F* 2. READS ONLY A BLANK RECORD AND A /* RECORD* SAMPLE 0108 F* AS INPUT DATA. SAMPLI 0109 F* SAMPL.1 0110 F* 3. CREATES THE OUTPUT DATA USING A SAMPLI 0111 F* LOOP IN THE CALCULATION SPECIFICATIONS. SAMPL 1 0112 F* SAMEL 1 0113 F* 4. USES KEYS FROM 000005 THROUGH 000500 SAMPL 1 0114 F# IN INCREMENTS OF 5. SAMPL 1 0115 F* SABPL1 0116 F* 5. SHOULD BE FOLLOWED BY SAMPLE PROGRAM 2 SAMPL 1 0117 F* TO VERIFY THAT THE FILE WAS PROPERLY SAMPL1 0118 F* LOARED. SAMPL1 0119 F* SAMEL 3 SAMPL1 0001 SAMPL1 0002 01 SAMPL 1 0003 0004 0201 IINPUT NS 01 1 C SAMPL1 0202 I 1 NODATA SAMPL1 0005 SAMPL 1 0006 0301 C Z-ADDO Z-ADDO 0007 0302 C RECNER SAMPL1 01 REPEAT SAMPL1 8000 0303 C TAG 0009 0304 C COUNT ADD COUNT SAMPLI 0010 0305 C 01 RECNER AI(I) 1 RECNBR SAMPL1 0011 0306 C 01 COUNT COMP 505 02 SAMPL1 0012 0307 C 01N02 EXCPT SAMP1.1 GOTO REPEAT 0013 030B C 01N02 SAMEL 1 RECNBR RECNER 0014 0309 CLR SUB 1 SAMPL1 SAMPL1 0015 0401 GOUTFUT T 201 0402 0 20 'SAMPLE PROGRAM 1 HAS' SAMPL 1 0016 27 'LOADED' SAMPL.1 0017 0403 0 RECNBRZ SAMPLI 0018 0404 0 0405 0 39 'RECORDS' SAMPL1 0019 0020 0406 0 61 'INTO AN INDEXED FILE.' SAMPL 1 0021 040B B T 2 SAMPL 1 0022 0409 0 21 'KEYS ARE IN ASCENDING' SAMPU1 0023 0410 0 42 'SEQUENCE STARTING AT' SAMPL 1 0024 0411 0 64 '000005 AND INCREASING' SAMPL 1 0025 0412 0 84 'IN INCREMENTS OF 5.' SABPLI 0026 0413 0 01 LR SAMPL 1 21 'SAMPLE PROGRAM 2 WILL' SAMPLI 0027 0414 D 44 'PRINT FROM THE INDEXED' SAMPLI 0028 0415 0 65 FILE TO SHOW THAT IT? SAMPI.1 0029 0416 0 86 'WAS PROPERLY LOADED.' SAMPL I 0030 0417 D 0031 0501 ODISKOUT E 01N02 SAMPL1 0502 0 СОИМТ SAMPL1 0032 0033 0503 0 94 'RECORD NUMBER' SAMPL.1 RECNER 0034 0504 B 129 SAMPL 1

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

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

				CORE USAGE OF REGII	CODE
START	NAME IF	CODE	NAME	TITLE	
ADDR	OVERLAY	LENGTH			
0000		0500	RGROOT	ROOT	
117B		008E	RGMAIN	INPUT MAINLINE	
1114		0067	ROSUBS	INPUT CTRL RIN	
1209		003A	RGSUBS	RECORD ID	
1243		0026	RGSUBS	CONTROL FIELDS	
1100		0008	RGSUBS	SUBSEG	
1269		000B	RGMAIN	TOTAL CALCS	
1274		0022	ROMAIN	INPUT FIELDS	
12A7		0048	RGMAIN	DETAIL CALCS	
12A2		0005	RGSUBS	CONSTANTS	
1320		0043	\$\$PGRI	RESET RESULTING INDR	,
12EF		0031	RGSUBS	EXCEPTION	
1296		0000	RGSUBS	SURSEG	
1363		005F	\$\$TOUT	DISK INDEXED OUTPUT	
1302		0093	\$\$SRBI	SYSTEM SUBR	
1455		0023	\$\$SRUA	SYSTEM SUBR	
147B		007B	\$\$SRBR	SYSTEM SUBR	
1486		0010	\$\$SRDF	SYSTEM SUBR	DISK DATA
1512		0.000	\$\$8RTC	SYSTEM SUBR	MANAGEMENT
152E		0015	\$\$SRPD	SYSTEM SUBR	
1543		0019	事事包投信權	SYSTEM SUBR	ROUTINES
1550		002F	\$\$SRBP	SYSTEM SUBR	
158B		0081	\$\$5RMO	SYSTEM SUBR	
1600		0043	\$\$SRSB	SYSTEM SUBR	
164F		0015	\$\$SRRD	SYSTEM SURR	
1664		000B	RGMAIN	TOTAL OUTPUT	
1680		0024	ROMAIN	LR & OVERFLOW PROCES	SING
166F		00170	RGSUBS	OVERFLOW SUBSEGMENT	NON-DISK DATA
16B0		0027	SSBDMC	SYSTEM SUBR	MANAGEMENT
189B		0024	REMAIN	CLOSE	LINKAGE ROUTINE
16E3		OQAA	RGSUBS	OUTPUT CTRL RTN	
1879		000E	RGSUBS	CONSTANTS	
178D		OOEC	RGSUBS	CONSTANTS	
18BF		0076	RGSUBS	LR PROCESSING	
1607		0000	RGSUBS	SUBSEG	
1887		0014	RGSUBS	LR CALCS	
1935		0072	ROMÁIN	OPEN	
		03495	SAMPL1	TOTAL CORE USAGE	•

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

0000		000	00000000	00000000	00000000	000000000	00000000	00000000	* GRS	*	
0020	00000000	00000000 00000000	00000000	00000000	00000000	00000000	00000011	B2000000	*	* 25.	
000	90000000	000000000	000000000	000000000	00000000	000000000	00000000	00000000	*	*	
0630		00000000 00000000 000000000	00000000	00000000	40FFFF00	00010002	оподород	00000000	*	* B(C(C *	
OCAO	OOOOOOOOO OOOO	000000000	000000000	C087189B	C087117E	43400000	00000003	n95C0C00	* 60¢	S GJ⊕C CR*< *	
0000	02000040 00000000		00000000	00000000	00000000	00000000	00000000	0000000	ж *	*	
OCEO	SECONDARY WORK AREA	WORK AREA 000000000	0000	000000000 0000	00000000	00000000	00000000	00000000	*	*	
0000	40F0F0F0 F0F0F0F0		F0F00200	00300052	0000003A	00000000	00001000	00122E0C	* 000000000 (O(K	*> <x &="">- #</x>	
0000	00009200	00009200 00806000	00000000	00000000	1008 2 (DISK)	8	00390000	80000002	· *	ADCACO : CB*	
0000	0010000D	OPOURDOO	00003401	00840000	03001200	BOSSEFFF	ODZOOSER	OOSFOESO	* & (¢(:A D(C K L##(0E# 1+ *	
0.000	40806000	40804000 000E800F	00 400E B0	22380000	DTF 2(DISK) A8F 18441	оп520пап	OSBBOOSE	000000	* + - + 58	YIDA(K(_E\$ 1 F *	
овпо	03300330	01000080	FFFFC980	CB000102 01E2	01E2D2C9 80017301	- 1	00000863	C9000010	*++- ##I . ABA	ABASKI A3AI I &*	
ONGO	00000000	000FF600 06C95C00	0926390	05E10040	470M70FF	OSBROO	BF000040	00000000	*< I 16 FI* E/ (* - #国以此〇)5	
ODEO	00018400	00018400 42000052 D40DD03C	D40DD03C	00000000	E0010000 00	00000140	овебовав	вововово	* AD B KH(&@	* -)31) & &-	
OBEO	80808080	80000000	00000000	000000000	00000000	00000000	00000000	00000000	*	*	
0E00	000000000	0000000 00000000 00000000		000000000	00000000	00000000 00000000 00000000		00000000	*	*	
OEBO	40404040	40404040	40404040	40404040	40404040	40404040	40404040	40404040	*	*	
OEEO	40084068	00086300	0EF 60000	080000000	00130EE0	DISK BUFFER 00700000 00040000	OCO40000	00000000	* H Y I +6	L+-(0 I) *	
OFOO		0000000 00000000 0000000	00000000	00000000	00000000	00000000	00000000	00000000	*	*	
0F20	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	a *	*	
0F40		0000000 00000000 0000000	00000000	000000000	00000000	00000000	0000 00000000 00000	00000000 ISK FILE	*	*	
OFEO	00000000	00000000	00000000	U00000000	00000000 00000000 INDEX OF KEYS	000000950	40A900C9	00001000	*	*>\$ I Z *I	
1.000	00200040	00200040 01DA0D70	0FF60E70	FFFFFFF		FFFFFFF	4666666	11111111	* - A1(0)6(0)44	**************************************	
1020		4444444		FFFFFFFF	PERFERENCE PERFERENTIAL ELINA CON CONCRETE CONTRE	FFFFFFFF	FFFFFFF		⁶	*************************************	
1,100	FFFFFFF	FFFFFFF	FFFFFFF	C20116RO	35101115	11163408	117A3401	1141E201	\$00 0 000000000000000000000000000000000	*************************************	
1120	00750205	BD1200F2	01068810	03F2900E	9C000F1B	792118F2	10049500	TOTECOFF	* 5BE'K 2AFB&C2&+*	* 159/0250* 5* #*	
1140	16B06C01	12008041	OEF28224	F28109RD	440EF2B2	OBF 28112	C2020CB4	R4 COB70004	*DOZAK(*A+2BU2AI*D+2B(2AKBB(4	D+2B(2AKBB(4 G D*	
1160	8F407A80	00F28708	00030E01	11791169	D20200C2	01000000	8711F57B	02C97BF1	*! : 26H C+AJ:JZKB	TKB BAK GJS4BI41*	
1180	C27B9FC3	7BFFC47A	40037820	C3C01011	BB0C0111	A5120675	02998810	00F21012	*##JC#### C8-C &J	&JB(AJUKFSERB& 2&K*	
1100	1140 000111F4	120A1E01	11A597B9	C100C010	11EEB880	00F29006	7AFFC47A	E0C3C087	* <aj4kc#ajup9a &.<="" td=""><td>&J>8 2&F:#D:-C 6*</td><td></td></aj4kc#ajup9a>	&J>8 2&F:#D:-C 6*	

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

_8AG2&C@ *#A*	6J(8 2&D GKI:*	A *CSCZGH B*	GKISBI -KC *	1: 8 R2&D G*	GOU5B\$9BQ2&<2AK*	GKIBBLTBA(05&*	*BUO5505BBI2&(D&(FKSD-(IKSBBI2&XF*	W GL-ADI 9DI*	115BB @ (#8*	B≠(I GKO GJ‡*	2855D#Z L= 2 D*	GLZSBA4B#5B**	C2&28AX¢AX2&Z@≠L() L(*	2B(@-+2A*@S+ GN*	0+ ANW GM# G*	GNCOMMC> MC85AI*)S+)HMZAMF()AM*	C2&C4A2% VW?AU\$%BCV@#S>*	-\$5A6ZAF. GNV@ >*	C4HM9%CW,>A,*	. B G A4HK15*	GNC' 18&X NO<>ARPXA-(XAUPXASN*	*!ASHS_ARS2B:ZAWR?AWS?AUWXARS GN.*	17 12AB GO(%A*	A4HNJ_AZ>2B*	*A(I)A(R+A(N_)*	CIBCODOH*	OA+ GNEAHNF*
66<@2J26(@0J_8AG2&C@	G DI 4H@	5&@KHJ<5AK5&M4HKGBA	< 6J≠' 2A.	NSBKS#A ZAKP1=	GJF8-C &RS GOU	GKXX (G	BIZ&(D&(FKSD	*&(FKTF(IQ9DU&(FGK&KW	*2&H GK? GK9 GJ\$4HL~*!!&BB<	&L*(EE(F(<)QF(B≠(I	# 6	@:LISEC			6'0+2A6 GLE'0+		S)6AS6AMK% >S+		S82-H+ SMK2-\$5A	6N*26C@O+ 6 ≠≠	A(= M GN*@	186X NOCYARE	SZB:ZAWR?AWS	GNC(ARM3)ARW2-0"	N.ZA(- G ##	9	+5A,5&I A4HNB&GC:BC@	G (0 (8AB &
*KC GKZ 6	* SA BB(4	*A 5&@KHJ	*J¢ GJJBA	*BKS8~C2&N5B<\$#A	*JB: R GJ	*HF5BK G	*01055058	*&(FKTF(*2&H GK?	*BI9DI &L	*4H≠4B°5B≠4D#2&	*@#LIZAL_B:	*5&≠4HI4A.8	*8%AL!25A(6A@>	*W GHN GH6'0+2A6	*NT4HM&_A_:2B'5A6	*%ASB> S>	*F@<	* \$>> \$85	*SAX GN*2	*PSAT9#=2&()	*AL GNC'	*IASMS_AR	* GO(GNC	*UW GH* GN.2A(-	*<9≠,2MG@0+%C,W	*ARP@ +5A	*G 4HNM
OOICEROI	871209BA	8708087	6012BDC0	90040087	00200112	JT 0E303510	F2902C06	00790409	400D7F78	C087117B	00F20004	FF7502FI	2E00138D	0EC08715	147BC087	38750109	OD2E0114	BCFF22AE	ZEBCOOZE	2BAE012B	0814F1B5	AC012215	C087158B	160CAC01	291DF282	OD152DAE	87000800	34081586
F29003BC	F21004C0	0A220AF2	A202C980	7880D9F2	MOVE 0218F210	LINK TO DISK OUTPUT C202 1363C201 0	A27802C9	20810409	02000030	C0871296	2C00133E	02017402	3CFF138D	811CBC62	15260087	2E001403	20011406	60020325	2DC08715	79AC0326 2B	00000134	AC012417	AC011922	81040087		OF198E01	7C0004C0	C0871526
EDB80118	000888000	01000090	C087120F	14740000	175029RB9 0218	1289C202 1363C201	20000912	47021012 A6C08713	BFBF90C2	027F0E09	227504FB	87132CE2	27F21029	00F2820E BC600EFZ	700EC081	3CFF1403	002E1454	AF01241B	36600106	03340B14	002AC087	AC01200D	AF012426	26F22016 BD800FF2	1000	ODOPAEO1	15427B07 037A0203	00BC410E
00303011	30008711	08121602	00F2010B	20011257	C0871664	000000087	0612A204		871289C0 87117B34 08131F5C 8F8F90C2	5D16868C	FF00F210	E20203C0	0127BB01	00F2820K	8713C2BD	C0871543	002E22BE	AC002526	20188501	OOFFFE	87155CBC	AE011917	AF012622	26F22016	OOODFFFF	00009001		02C09000 00BC410E
11EBF287	8F007408	12 B5101434	11FF7D40	OC 9REE0100	LCS C01018A2	12 A72C000E	0C04500B	A5100E	87117B34	00088000	F21000B9	3C7A1349	F29032BB	10000000	F28107C0	7BBS0136	011452AC	03B40132	221454F2	0EC08700	2A147AC0	C015160C	AC012619	F3AE0119	00200087	2B26C0B7	0001340B	000C7B01
16803032	C0870004 8	OC#50112	00000087	153502	TOTAL CALCS C67820C3 C01018A2	000087	DETAIL CALCS 78 02C9F290	09187944	8712B9C0	1C8C0505 01068C0C	FF7404FB	027A0000	08884003	ODB6013C	F6BD700E	2113AF202	B6012236	4003F290	ACCEBEOC	8703BC70	100CBF00	OFB81027	22F2823A	80011914	158BAC01	700EAC03	0BB51009	87000C7C
1269C087	C2020CB4	30120811	B7 11D1C201 0C00C0B7 11FF	87122278 2003F210	D9C08711	987	ANTS FSFOFS	A306200D 091879	- 1	C9C09013	02FD7502	01134E	OB09B401	8F32B501	55008714	1450AB01	AC012238 AE00222E	063C4C00 000000kB	002238F2	87155CF2	B9FF2AF2	15438080	FSAD0119	C0871543	- 1	F21407EC	BC 400EB501	081554C0
1100 12430087	00E20100	01007510	11CAC0	87122278	11C27AB0 D9C08711	881EB502	CONSTANTS 16E4F0FS FSF(500D0612	F29008C0 8712EFC0	02C97904 C9C09013	-F74	3C7B1349	7510FFB4 0809B401	38200113	26C08714	15233408			00222EAE	E20127C0	17E20123	01130087	8F012214	C087160C	2426C087 149CC087	ODB9FF2B	011917BC	87000034
1100	11E0	1200	1220	1240	1260	1280	1200	1.2CO	12E0	1,300	1320	1340	1360	1380	13A0	1300	13E0	1,400	1420	1440	1460	1480	1400	1400	14E0	1500	1520	1540

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

*HR4*II&BBC <ECAPK<LLP2<E1PB<B\$CI* -4HOEABN:ZAOIU@ZN(9 12* *AD@*N(5AI?ARU5AR)ARU5B(6BD.) DHO* ##4HG+" 12A#5HLXAFF5FN* BN*884X#4X247 GDIZB+84X &DD GN):* 66X4AJ5AL4AL G 4HOT#GC:AC@ N G H* 8-C AGNERF AGNESTREEDS **OUDV4HP(4BPASBE*A(+% P9U% PEU #* *67 026* *H8-(R2&A*AFSC*AFWC9#K2&F* K2BN9#* **2&F" &2DI"H+2AR2G+"H+2AH@\$F/@&P* *0000 AND INCREASINGIN INCREMENTS* 2 WILLPRINT* **OJSB*&<FWP#(M@PM GNP(E(APQ<MMPZ* GN#* GT2A*@ DH6# 8-C 30m GO(4HDF:-R(D(G, GOP) *:BBBDB2&C#BB GK42G1\$\$BDMC 03/18/* *704HDDZADJ\$ 6 D SRB GJBBBCOBA(A5* *AMPLE PROGRAM 1 HASLOADEDRECORDS* *INTO AN INDEXED FILE, KEYS ARE IN* * ASCENDINGSEQUENCE STARTING ATCO* *AT ITWAS PROPERLY LOADED. (RECORD* 28 CUS# ** SG+ FNH8-F2&E+AFNC* S2AII SNG * NUMBERAHGIS-CZ&FG-(IQ9 6.14:-C@ (P*(+2BI9A+*11+2J¢BB(4 G (3) (3) MAX 696 600 60×5AKRSBE 6 DC ĔĤ 10 8A12&F(1(1(4&80 8-A2&DZ INDEXEDFILE *AUIGBOICCNANI+ NAUIZ 5.SAMPLE PROGRAM KF 2B&6AUG6BD6%# ΑŽ i. <€ * FROM THE 5B+B6 9 O (1) A* RAL G HUNE * \$\$\$P\$\$ 3408164F BD800FF2 011E7502 136C0106 06B50215
 C0870000
 78200300
 1018AAC0
 87158034
 0816867A
 20190004
 0100168B
 C0871617

 782019C0
 87000000
 06000000
 782003C0
 1018AE78
 01170010
 166F7BFF
 1778FF18
 81098F00 221587C0 11750113 B40113C0 B7000034 0816637B 07037A01 037C0004 C0B70008 F7F0340B 16D62C01 16D11BC0 B70004B0 223802C0 B71142C2 0216H0C2 010D4135 10F21006 BD0010F2 0409BD48 OEF20119 F2870EBD 480EF281 083C3B17 613C3B17 BF407502 0EC2010C 00C0B700 00400100 00020140 00000002 01400000 010001E2 C9D5E3D4 40C1D540 C9D5C4C5 E7C5C440 C4C9D3C5 4BD2C5E8 E240C1D9 C540C9DS F0F0F0F5 40C1N5C4 40C9N5G3 N9C5C1E2 C9N5C7C9 N540C9N5 C3N9C5N4 C5N5E3E2 87000001 04002034 08160534 0215FA2C 01160924 3C6C15CC 89800FF2 10043C9C 15CC7501 09AF0119 24B60119 AE011924 B5020D36 02160B6F 00160B16 06F28210 36011607 36021607 6CFF0000 C08715BB 3N001609 F2811C3C 00160836 CF0E0015 F416096C 000000C2 020006AE 010B24B5 4FF2870E B81027C0 101616C0 87152E8A 1016E416 E5340817 8C340217 01750205 9C010B0E 2C001739 242C0017 4524C0FF CID4D7D3 C540D7D9 D6C7D9C1 D440F140 C8C1E2D3 D6C1C4C5 C4D9C5C3 D6D9C4E2 40C1E2C3 CSDSC4C9 DSC7E2C5 D8E4CSD5 C3C540E2 E3C1D9E3 C9DSC740 C1E3F0F0 40E2C8D6 E640E3C8 C30609C4 87117R7A 20033080 928C1313 17B28C05 1A17B88C 021E0f09 BAF01E9B 021C908C 062617BF 8C143C17 D4C08716 D70C050D C117988C 141417E9 4838200C D9F21041 1C011762 0A1C0117 660AB9FF 12F21006 BD0012F2 0415B9FF 6500000C D600000C D7RD4D0E F20209B9 010EBD44 0EF2110A C2020CB4 C0870004 40F240E6 C9D3D3D7 D9C9D5E3 00048434 00000B801 0FF21006 0C8F0C8F 0C90B8F0 00782001 F290046C 001C167D 000AF281 LINK TO PRINTER OUTPUT CONSTANTS 7A02C278 04C2F210 037B02C2 C0871274 F2870F5B 5BC2C4B4 C340F0F3 04830087 4BF10905 1540|9C002207 4E000615 88786006 F290054E 0106158A BD0022F2 C540E3D6 09187900 05008700 C1C4C5C4 OVERFLOW SUBSEGMENT C4C6C91G 0607200F D4D7D3C5 40D7D9D6 C7D9C1D4 E84053D6 00997502 C4C5E7CS C087155C R810278B 1027F210 1EC08716 0819345C 8F8F90C2 020C000C 050DC117 D7C5D9D3 2003F290 18BF3501 40D5E4D4 C2C5D934 08189A78 01160936 0216090C 0315F615 00000100 0000FFFF D440F3CB C540C9B5 4017119116 16640087 40D5C640 F54BE2C1 E3E6C1E2 19870087 871564C0 01130087 40C6D9B6 1.640 1027E401 CLE340C9 3.800 1,600 1820 1840 1800

*

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

15 * .ZP'(M#GK(KLQV GOP(E(AF)<M#G:<N*<br 16 ** Q& QU(LNG8 GOP GJ* G D BA(11*</th <th>75 ***X E7:AB: C5AR5BE G D.5BE G DB5*</th> <th>03 *BE8AC2&(\$ 9& 2&D@GRE(AC(R2AB5AC*</th> <th>OD * GR-26CBB(4 G DIJBA(@ (@O(I(6(*</th> <th>* #C9 I)H* 99</th> <th>*</th> <th>87 */2CW,>A,PSAT9#2&(mtS 6+T@ m 6*</th> <th>81 * A4H+(* SG+ F†18-F2&E+AF†J' S2A*</th> <th>CO *II S1+ G1, G AD -4H†W\$GC:AC@ D *</th> <th>8D *6 H G 4H1++6C:BC@ D G H G 4H1(*</th> <th>1ክ * 6 (@ (8여ፔ ኤ @q+ 6sf4Hs14ksD%As*</th> <th>02 *LU@X109 Z&D@*105AI?ARU6AR}ARU5B*</th> <th>00 *(GB\$N) \$K\$&2B&GA\$J&F\$J&# G†E* *</th> <th>00 *\$L2A*@ \$K5A\$L6¤\$L<c\$ *<="" th="" †="\$L%" †r+=""><th>82 * BB >A(U5AL G A SS##4H51_AZ)2B*</th><th>AE *(9#,2MG@O+%C,W G *A(I)A(R+A(\$()*</th><th>04 *ARP@ +5A.5&I A/¢DGDIIGDIDDEDDD*</th><th>04 *IBDBDWBDGGDD=UI<pre>04</pre></th><th>04 **DBDDDGDI DL&C=GDEE<jbdg d(ildwfd*< th=""><th>16 *GEEEE.CBB- = 0*</th><th>15 *DGDDDGCN\$DDEJ=HD&M(CDMOD(I.B(NN*</th><th>00 *NDDECNDEGNHD=GDDDRDE)U≭ *</th><th>CO *> G*04E\$SEG4E2SEA4E'@G3EE*C*GGI *</th><th>34 *G DZKB: G*22GC2GZ G\$=4H*L; 47*-4*</th><th>10 *2AP; 57SEA'HS2D¢ 6 @D>260@B>26&*</th><th>14 *@A\2GC\$O\4B2* 35BA(DBBPE(FF*G GM*</th><th>F1 */ D+III ACALL DIR SYSTEM F1R1*</th><th>40 *F2R200003980999990000405 *</th><th>D9 * * *2 *@0\$\$RBDTSD\$\$R*</th></jbdg d(ildwfd*<></th></c\$></th>	75 ***X E7:AB: C5AR5BE G D.5BE G DB5*	03 *BE8AC2&(\$ 9& 2&D@GRE(AC(R2AB5AC*	OD * GR-26CBB(4 G DIJBA(@ (@O(I(6(*	* #C9 I)H* 99	*	87 */2CW,>A,PSAT9#2&(mtS 6+T@ m 6*	81 * A4H+(* SG+ F†18-F2&E+AF†J' S2A*	CO *II S1+ G1, G AD -4H†W\$GC:AC@ D *	8D *6 H G 4H1++6C:BC@ D G H G 4H1(*	1ክ * 6 (@ (8여ፔ ኤ @q+ 6sf4Hs14ksD%As*	02 *LU@X109 Z&D@*105AI?ARU6AR}ARU5B*	00 *(GB\$N) \$K\$&2B&GA\$J&F\$J&# G†E* *	00 *\$L2A*@ \$K5A\$L6¤\$L <c\$ *<="" th="" †="\$L%" †r+=""><th>82 * BB >A(U5AL G A SS##4H51_AZ)2B*</th><th>AE *(9#,2MG@O+%C,W G *A(I)A(R+A(\$()*</th><th>04 *ARP@ +5A.5&I A/¢DGDIIGDIDDEDDD*</th><th>04 *IBDBDWBDGGDD=UI<pre>04</pre></th><th>04 **DBDDDGDI DL&C=GDEE<jbdg d(ildwfd*< th=""><th>16 *GEEEE.CBB- = 0*</th><th>15 *DGDDDGCN\$DDEJ=HD&M(CDMOD(I.B(NN*</th><th>00 *NDDECNDEGNHD=GDDDRDE)U≭ *</th><th>CO *> G*04E\$SEG4E2SEA4E'@G3EE*C*GGI *</th><th>34 *G DZKB: G*22GC2GZ G\$=4H*L; 47*-4*</th><th>10 *2AP; 57SEA'HS2D¢ 6 @D>260@B>26&*</th><th>14 *@A\2GC\$O\4B2* 35BA(DBBPE(FF*G GM*</th><th>F1 */ D+III ACALL DIR SYSTEM F1R1*</th><th>40 *F2R200003980999990000405 *</th><th>D9 * * *2 *@0\$\$RBDTSD\$\$R*</th></jbdg d(ildwfd*<></th></c\$>	82 * BB >A(U5AL G A SS##4H51_AZ)2B*	AE *(9#,2MG@O+%C,W G *A(I)A(R+A(\$()*	04 *ARP@ +5A.5&I A/¢DGDIIGDIDDEDDD*	04 *IBDBDWBDGGDD=UI <pre>04</pre>	04 **DBDDDGDI DL&C=GDEE <jbdg d(ildwfd*< th=""><th>16 *GEEEE.CBB- = 0*</th><th>15 *DGDDDGCN\$DDEJ=HD&M(CDMOD(I.B(NN*</th><th>00 *NDDECNDEGNHD=GDDDRDE)U≭ *</th><th>CO *> G*04E\$SEG4E2SEA4E'@G3EE*C*GGI *</th><th>34 *G DZKB: G*22GC2GZ G\$=4H*L; 47*-4*</th><th>10 *2AP; 57SEA'HS2D¢ 6 @D>260@B>26&*</th><th>14 *@A\2GC\$O\4B2* 35BA(DBBPE(FF*G GM*</th><th>F1 */ D+III ACALL DIR SYSTEM F1R1*</th><th>40 *F2R200003980999990000405 *</th><th>D9 * * *2 *@0\$\$RBDTSD\$\$R*</th></jbdg d(ildwfd*<>	16 *GEEEE.CBB- = 0*	15 *DGDDDGCN\$DDEJ=HD&M(CDMOD(I.B(NN*	00 *NDDECNDEGNHD=GDDDRDE)U≭ *	CO *> G*04E\$SEG4E2SEA4E'@G3EE*C*GGI *	34 *G DZKB: G*22GC2GZ G\$=4H*L; 47*-4*	10 *2AP; 57SEA'HS2D¢ 6 @D>260@B>26&*	14 *@A\2GC\$O\4B2* 35BA(DBBPE(FF*G GM*	F1 */ D+III ACALL DIR SYSTEM F1R1*	40 *F2R200003980999990000405 *	D9 * * *2 *@0\$\$RBDTSD\$\$R*
C12 531825C0 8716D70C 050DC117 9E8C1414 183ABC15 OPEN MAINLINE 518 78C08716 D7C08711 7EC08700 0400C201 0C005F1A	'	079 1000F290 043C8719 854D0103 0C99F281 07750103	4CO 8700048F 11C2010C 003C400D 003CF00D 090C070D	PL1 000 00000000 00000000 00000000 00000000	000 00000000 00000000 00000000 00000000	123 B9FF2AF2 100C8F00 2A1A22C0 871A23BC 002AC087	74E 00061A4F 786006F2 90054E01 061A51BD 0022F281	087 00000104 00203408 1A667B07 037A0103 7C0004C0	B7B 07037A02 037C0004 C0870008 C0870000 34081A8D	090 0000BC41 OECO871B 46340B1B OF34021B 042C011B	004 3C9C1AI6 750109AF 011924B6 0119AE01 1924B502	OF2 82103601 18113602 18116CFF 0000C087 1AC53D00	11B 1336021B 130C031E 001AP90E 001AFE1B 134C0000	113 COB70000 0100E2E2 FFFF3408 1B31AD01 291DF282	CO3 2826CO87 00009CO1 0T09AEO1 0T19BEO1 0T1R4TAE	009 0001210A 04070409 09070409 04040204 02040404	404 04FE2409 0C040704 041B0404 040B0707 04040704	310 0AFE0704 05050C11 0204070F 040N0913 04260604	080 80FE8080 8080B0B0 8080B0B0 80B0B016	405 11FE0804 10140D0A 04141404 0C090B0B 040C1515	408 04FE0704 04040204 021D25FF 00000000 00000000	774 0232E202 0174027D 7C0733C2 021CC39C 07078FC0	2F2 8703F287 29C08718 FE34081C 535E0034 377D6034	DO8 35F2840A C0870000 7C042EF2 87167C02 2EF28710	232 5C003335 C2010B04 C20217C5 8C06061C 87C08714	001 OBCINBDB 40C4C9D9 40E2E8E2 EBCSD440 C6F1D9F1	8FO F9F9F9 F9F9F0FO F0F0F4FO F5000040 40404040	OIC F2000000 00001CRC 065R5RD9 C2C4E3E2 065R5RD9
1900 8C132917 FD8C143F 18128C12	DCDC6C00 C5377A01	1940 02058801 03F2900D 7BC00079	1980 C087195F F2870AC2 020CB4C0	END OF SAMPL1	3950 0000000 00000000 00000000	1400 21AC0326 2BAE012B 17E20123	1A20 00000134 081A4B9C 0022074E	1840 098F0022 184EC087 182EC087	1.460 870008C0 87000034 081A7B7R	1A80 C087000C 7C000C78 0102C090	1840 13243C6C 1AD6B980 OFF21004	1AC0 0D36021B 150F001B 121B10F2	1AE0 1813F281 1C3C001B 1236011B	1800 00C20200 00AE010D 24E50113	1820 0DB9FF2B F21407BC 700EAC03	1B40 011917BC 400EB501 0RB51009	1860 09020402 04260204 07070404	1880 04020404 04070409 0F041310	18A0 47050505 05080302 02208080	1BC0 04070404 04070A15 1B040405	1.BE0 15040404 050A1504 02070408	1000 2EC0871C 7074027B E2020774	1020 87000469 D2027AC0 8710F2F2	1040 F281175E 003537E2 02017D08	1C60 7C012EF2 870A7BF0 2E740232	1080 E140040E 09090900 60000001	1040 C&F2D9F2 F0F0F0F0 F3F9F8F0	1000 40404040 40404040 40404010

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

	0101	н оев	SAMPL2
	0102	·	SAMPL2
	0103	3 F# *	SAMPL2
	0104	F* THIS PROGRAM - *	SAMPL2
	0105		SAMPL2 SAMPL2
	0107		SAMPL2
	0108		SAMPL2
	0110		SAMPL2 SAMPL2
	0111	F* 3. USES A BLOCK LENGTH FOR DISK WHICH *	SAMPL2
	0112		SAMPL2
	0113	The state of the s	SAMPL2 SAMPL2
	0115		SAMPL2
	0116		SAMPL2 SAMPL2
	0118		SAMPL2
0001		· 广东南京大大大大村村村大大大大大大大大大大大大大大大大大大大大大大大大大大大大大大	SAMPL2
0001		FDISKOUT IPE F 512 128 06AI	01 SAMPL2 SAMPL2
0002.	0111	TODITOT OF A ROSE OF TRIBUTOR	SF1111 C. Z.
0003		IDISKOUT NS 01 1 CO	SAHPL2
0004	0202		SAMPL2
0005 0006	0203		SAMPL2 SAMPL2
0007	0301	C 01 COUNT ARR 1 COUNT 30	SAMPL2
8000		00UTPUT H 204 1F	SAMPL2
0009 0010	0402		SAMPL2 SAMPL2
0011	0404		SAMPL2
0012	0405		SAMPL2
0013	0406 0407	The state of the s	SAMPL2 SAMPL2
0015	0408	0 KEY 6	SAMPL2
0016	0409		SAMPL2
0012	0410	and the state of t	SAMPL2 SAMPL2
0019	0412	0 COUNT Z 3	SAMPL2
0020	0413		
ついだす	√→.I. ·	G 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

FIELI) NAMES	USED		
STMT#	NAME	DEC	LGTH	DISF
0013	PAGE	0	004	0110
0004	KEY		006	0105
0005	DESC		013	0112
0006	RECNER	0	003	0115
0007	COUNT	0	003	0118

				CORE USAGE OF RPGII CODE
START	NAME IF	CODE	NAME	TITLE
ABBR	OVERLAY	LENGTH	Entre Entre Comme	r', I'', I'' T
0000 12FB		068C 009D	RGROOT	ROOT
			RGMAIN	INPUT MAINLINE
1294		0067	RGSUBS	INPUT CTRL RTN
1398 1302		003A 0026	RGSUBS	RECORD ID CONTROL FIELDS
128C		0008	RGSUBS	SUBSEG
1250 13F8		0008	RGSUBS \$\$ISIP	DISK IDX SEQ INPUT
1433		0019	\$\$SRCR	SYSTEM SUBR
1440		0019 007B	\$\$SRBR	SYSTEM SUBR
1407		0026	\$\$SRIF	SYSTEM SUBR
14F3		0020	\$\$\$RRC	SYSTEM SUBR
1574		0029	\$\$SKKE \$\$SKRI	SYSTEM SUBR DISK DATA
1590		0043	\$\$\$RIC	SYSTEM SUBR MANAGEMENT
15E0		0010	SSSRTC	SYSTEM SUBR ROUTINES
iSFC		1800	\$\$SRMO	SYSTEM SUBR
1670		0043	\$\$SRSB	SYSTEM SUBR
1600		0015	SSSRRD	SYSTEM SUBR
1605		002F	\$\$SRBP	SYSTEM SUBR
1704		0015	\$\$SRPD	SYSTEM SUBR
1719		0020	RGMAIN	INPUT FIELDS
1746		0010	RGMAIN	DETAIL CALCS
1745		1000	RGSUBS	CONSTANTS
175C		0033	RGMAIN	DETAIL OUTPUT
1756		0006	RGSUBS	CONSTANTS
178F		8000	RGMAIN	TOTAL OUTPUT
17BE		0024	RGMAIN	LR & OVERFLOW PROCESSING
179A		0024	RGSUBS	OVERFLOW SUBSEGMENT
17E2		0028	ROSUBS	SUBSEG
182E		008E	RGMAIN	OPEN
1816		0018	RESUBS	CONSTANTS
1806		0000	RGSUBS	SUBSEG
1993		0018	RGMAIN	CLOSE
1880		OOAA	RGSUBS	OUTPUT CTRL RTN
1988		0020	RGSUBS	CONSTANTS
19AB		0030	RGSUBS	LR PROCESSING NON-DISK DATA
19DB		0027	SSBDMC	SYSTEM SUBR MANAGEMENT
	,	03588	SAMPL2	LINKAGE ROUTINE TOTAL CORE USAGE

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

CENTRE FETTI FET		Om No string	A 3 0 4 7 0										
40404040 404	0000		F6F04040	песясаль	D9C440D5	E4B4C2C5			40404040	*000000	RECORD MUMBER	12	*
ADMINISTRA ADM	0020		40404040		40404040	40404040	- 1	- 1	40404040	*			*
	3080	40404040	46404040	40404040	40404040				00000000	*	Z X	BOO	*
MORDADO	0000		0000		C0871993	- 1	- 1		D95C0C00	*	GRL GK#C	CR*<	*
### ### ##############################	0000	02000040	ATORS 00000000	00020000					00000000	*E	'n		*
POP-0FOPE F-6F-D19FS C3668-C4 4045-C414 C2C589-FO F 172-FOFE F-2F-OFE F-2F	OUED	SECONDARY G00000000	WORK AREA	00000000	00000000	00000000	00000000	00000000	00000000	*			*
The control of the	0000		FeFon905	ು	4005E4D4			F2F0F0F0	F102040B	*0000050K	SCORD NUMBER0120	0120001EB	*
ICCOCOGIGO CACOCOCO SAZOGOGGA ACCOCOGA OFFICIAND	0000	(DISK) 430054	00003900	00300800	UIOOOEEO	13850000	- 1	- 1	C0871381			8I P2 GL1*	*
ΦΕΕΕΘΑΘΙΘΟ ΓΡΕΓΕΖΘΙΟ ΕΡΕΕΡΘΙΟ ΕΡΕΕΡΘΙΑ ΕΝΟΟΣΟΙΣ ΒΕΘΟΘΙΟΘΟ ΕΝΕΕΡΘΙΑ ΕΝΕΕΡΘΙ	6140	010000000	OCCUTOR	90000000					12CZ0D1D	0.8	图)组= #	Y1HAzz(*KB()*	*
PRINTED 100-00-00-00-00-00-00-00-00-00-00-00-00-	9940		OF606F60	0	FFFFC980			_	0000000			≠G@ 1	*
PRINTED CORREGION CEOCOGO O1840042 110152B4 ODBESEGO ESOLGGE CEOCOGO OCCOGO	0300						- 1		E1004047		36	9 /	*5
OBSTRANDA PRINT BOTTER NAME A (NUCE 0.00200000 0.00200000 0.00200000 0.00200	ойно	(TRACTR1) O/054FFFF	OSBBOOSE	00004040	00000000		11015204		55012669		AAD	BJAKK(K@ NAW-*	*
ΦΩΕΡΩΑΤΑΟ ΙΣΕΚΩΡΑΘΑΘΕ ΓΑΓΕΛΙΖΟ ΕΣΑΘΑΘΑΘΑ ΑΝΕΤΟΝΟΘΟ ΦΟΘΟΘΟΘΟ ΦΩΕΡΩ ΝΙΜΡΕΡΩ ΚΑΙ ΦΩΕΘΩΡΟΣΕΑΟ ΙΣΕΚΩΡΑΘΑΘΕ ΓΑΓΕΛΙΖΟ ΕΣΑΘΑΘΑΘΑ ΑΝΕΤΟΝΟΘΟ ΦΩΘΟΘΟΘΟΘΟ ΦΩΘΟΘΟΘΟΘΟΘΟ ΦΩΘΟΘΟΘΟΘΟ ΦΩΘΟΘΟΘΟΘΟ ΦΩΘΟΘΟΘΟΘΟ ΦΩΘΟΘΟΘΟΘΟ ΦΩΘΟΘΟΘΟΘΟ ΦΩΘΟΘΟΘΟΘΟ ΦΩΘΟΘΟΘΟΘΟ ΦΩΘΟΘΟΘΟΘΟ ΦΩΘΟΘΟΘΟ ΦΩΘΟΘΟΘΟΘΟΘΟ ΦΩΘΟΘΟΘΟ ΦΩΘΟΘΟΘΟ ΦΩΘΟΘΟΘΟΘΟΘΟΘΟΘΟΘ	CECO		0001 400B	(2)	80808000		OCE OF OF O		40D9C5C3	<\$	₩.	0000055 REC*	*
CSCACACAC 40000000 ** CSCACACACA 40040400 40000000 ** CSCACACACA 40040400 40404040 40404040 40404040 ** CSCACACAC 40404040 40404040 40404040 40404040 40404040 ** CCACACACA 40404040 40404040 40404040 40404040 40404040 ** ** CCACACACA 40404040 40404040 40404040 40404040 40404040 40404040 ** ** 40404040 40404040 40404040 40404040 40404040 40404040 40404040 ** ** ACACACACA 40404040 40404040 40404040 40404040 40404040 40404040 ** ** ACACACACA 40404040 40404040 40404040 40404040 40404040 40404040 ** ** ACACACACA 40404040 40404040 40404040 40404040 40404040 40404040 ** ** ACACACACA	ONEO		115E 4D4C2		FIFICIEZ				90999999		HER KIAGE I		*
CUENCIPO FER GADAGAGO ADADAGAGO A	OUTO	00000000	00000000	0		000000000	00000000		00000000	*			*
\$\text{C}\$	UEEO		F6F04040	4	40404040	40404040			40404040	*6600066			*
40404040 40404040	9030	40464646	40404040	4				40404040	40404040	*			*
0B 1018K Name ACCORDAGO ACCO	0F20	404040	40404040	4		40090503	0.609046		C2094040	*	RECORI	КЕСОКО МОНВЕК	*
### ### ### ### ### ### ### ### ### ##	0440			40404040	46464046	40404040	30904040	40404040	40F0F1F2	*		0.12*	*
40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 ** ** 40404040 40404040 40404040 40404040 40404040 ** ** 40404040 40404040 40404040 40404040 40404040 ** ** 40404040 40404040 40404040 40404040 40404040 ** ** 40404040 40404040 40404040 40404040 40404040 ** ** 40404040 40404040 40404040 40404040 40404040 40404040 ** <	ÚF ả0		00099801	0F766600	00001601	00540160			46404040		316 0.3(RJ(M956065	550045	ĸ
40404040 40404040 CSESB&B9 C440BEE4 B4C2CSB9 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 FIF3F0F0 FOF0F7F0 40404040 ** 40404040 40404040 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 CSCSB&B CA40BEE4 B4C2CSB9 40404040 40404040 40404040 ** 4040404040 40404040 CSCSB&B CA40BEE4 B4C2CSB9 40404040 40404040 40404040 **	0840	40404040	40404040	ব			40404040	40404040	40404040	*			対
40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 40404040 ** ** 40404040 40404040 40404040 40404040 40404040 40404040 ** ** 40404040 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 **	00JO				C440DSE4		40404040		40404040	*	КЕСОКО МИМВЕК		*
40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 ** 404004040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 **	0440	40404040	40404040	4			FIFSFOFO	FOFOF7FO	40404040	*	0130	013000070	*
40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 ** 40404040 40404040 40404040 40404040 40404040 **	1000	40404040	40404040				40404040	40404040	46404040	*			*
######################################	1010	40.10.1040		43			40404040	40404040	40404040		RECORD NUMBER		*
### ### ### ### ### ### ### ### ### ##	3,060	40404640	40404040	4.			FIF4F0F0	FOFOF7F5	40404040	*	0140	014000075	*
40404046 40404019 C5C3D619 C44015E4 14C2C5D9 40404040 40404040 40404040 * 40404046 40404040 40404040 40404040 40404040	1,050		40404040				40404040	40404040	40404040	*			*
40404040 40404040 40404040 40404040 * 40404040 40404040 40404040 40404040 40404040 * 40404040 40404040 40404040 40404040 * *	1000		40404006	٥			40404040	40404040	40404040	*	RECORD NUMBER		*
40404040 40404040 40404040 40404040 40404040 40404040 40404040 * 40404040 40404009 CSCSD6D9 C440DSE4 D4C2CSD9 40404040 40404040 40404040 *	3.950	40404040	40404040	4			F1F5F0F0	FOFOFBFO	40404040	*	015(015000080	*
CSC3D6D9 C440DSE4 D4C2CSD9 40404040 40404040 40494040 *	1100	40404040	40404040	4			40404040	40404040	40404040	*			*
	1140	40404040	40404009				40404040		40404040	*	RECORD NUMBER		*

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

1160	1160 464646 464646 464640	40404040	40404040		108 FOR INDEX OF DISK FILE 40404040 404040F0 FIF 4C900 404906C9 0000118C	108 FO F1F6C900	IOB FOR INDEX OF DISK FILE	18K FILE 0000118C	*)(1 Z 1910 *
1160	00200000	14380054	00200000 14380054 11760054		INDEX OF MEYS FOR OFFICE FOR 1 FOR 1 FOR 0 080F0F0	00F0F0F0	FOF1FOOO	00805050	* - M#(MJA(MO00005 000010 00*
1140	FOFOFIFS		000400F0 F0F0F0F2 F0000480 F0F0F0F0	F0000480		F2F50008	00505050	F0F3F000	*00015 D 000020 D 000025 H 000030 *
1100	1100 0880F0F0 F0F0F3F5 000060F0 F0F0F0F4 F0000C80 F0F0F0F0 F4F50010 00F0F0F0	F0F0F3F5	000ceare	F0F0F0F4	F0000CB0	FOFOFOFO	F4F50010	005050	*H 000035 < 000040 < 000045 A 000*
1150	FOFSF000	1080F0F0	FORSFOOO 1080F0F0 F0F0F5F5 001400F0 F0F0F0F6 F0001480 F0F0F0F0 F6F50018	001400F0	FOFOFOF6	F0001480	FOFOFOFO	F6F50018	*050 & 000055 H 000060 H 000065 Q*
1200	OGEOFOFO	F0F7F000	00F0F0F0 F0F7F000 18G0F0F0 F0F0F7F5 001C00F0 F0F0F0FR F0001C80 F0F0F0F9	FOFOF7#5	OOICOOFO	FOFOFOFR	F0001C80	FOFOFOFO	* 000070 @ 000075 * 000080 * 0000*
1220	F8F50020	OOFOFOFO	F0F9F000	2080F0F0	FOFOF9F5	002400F0	002400FG F0F0F1F0	F0002480	*85 - 000090 - 000095 U 000100 B *
1240	FOFOFOFI	FOFS002R	FOFOFOF1 FOFSOO28 OOFOFOFO F1F1F000 2880F0F0 F0F1F1F5 002C00F0 F0F0F1F2	F1F1F000	2880F0F0	FOFIFIFS	002000F0	F0F0F1F2	*000105 Y 000110 Y 000115 Z 00012*
1260	F0002EB0	FOFOF0F1	F0002UB0 F0F0F0F1 F2F50030	OOFOFOFO		3080F0F0	3080F0F0 F0F1F3F5 003400F0	00340050	*0 % 000125 0 000130 0 000135 4 0*
1280	FOFOFIFA	F0003480	44444444	C20113FB 35101295		IZ963408 12FÁ3401	12FÁ3401	12C1E201	*00140 4 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
12A0	00750205	BB1200F2		03F2900E	01068810 03F2900E 9C000F1B 792118F2 10049C00 101CC0FF	792118F2	10049600	101CC0FF	* 58E'K 2AF8&C2&+* 199/02&B* &* ##
1200	12C0 13F86C01 120fff141 OEF28224 F28109Fb 440EF282 ObF28112 C2020CB4 C0870004	120000141	OEF28224	F28109ED	440EF282	00F28112	C2020CB4	C0870004	*LBXAK('A+2BUZAI'D+2B(2AKBB<4 G D*
12E0		00F28708	BF407AB0 00F2B708 00030E01 12FA12E9 D20200C2 010C00C0 87138479 20D77920	12FA12E9	n20200C2	01000000	87138479 24	20077920	*! : 26H C+AK:KZKB BAK GLB9-P9-*
1.309		MAINLINE DBF29003 7820D67E	20187802	C97EF1C2 789FC37B	789FC37B	FFC47A40	C37820C3	C0101347	*02&C\$-0\$-0\$DI\$1B\$-C\$XD: C8-C &L6*
1320	00011334 13957502	13957502	99881000	99881000 F210120C	01138313	99160113	991E0113 3497B9C1	00001013	*(AL4LN5BR8& 2&K(ALCLR;AL4F9A &L*
1340	70888000	F290067A	70838000 F290067A FFC47AE0 C3C08713 D2C08717 8FC08717 RE3C3213	03008713	D2508717	8FC08717	RE3C3213	7CF2870D	**8 2&F:#N:-C 6LK GP! 6P=@2L@26(*
1360		R80118F2	3C30137C RE0118F2 9003EC00 1CBB0100 E20100C2 020CB4C0 8700048F	1CBB0100	E20100C2	020CB4C0	8700048F	OC74083C	*@OL@BAG2&C@ *#A SA BB<4 G DI AH®*
1.380		E88000F2	CO87128C B88000F2 1004C087 1398BA01 0075103C 1397128C	13988401	0075103E	1397128C	RECORD ID R50112R5	RECORD ID R50112F5 10143408	* GK(8 2AD GLO:A 5AGLPK(5AK5AH4#
1300	13A50201	13079006	220AF287	08008713	59008713	60C2010C	60C2010C 00C08713	BE7DF000	*LVB6LG*CSC28H GLR GL~3A< GL+'Q *
1.500	F.6108C0	87139E82	82139E82 02C98000 1732C087 13R17820 C3F21015	17320087	13R17820	C3F21015		35020C9B BB01002C	*ZA. GLIBBI PZ GLIB-CZ6N5R<54A Z*
1.3E.0	0113E61A	74020978	0113E61A 7A020978 80A9F290 04008713 517A80A9 C0871355	04008713	51748009	00871355	B4080984	01088840	*ALW1:BIB R2&D GLUS R GLN4H146.8 *
On#)	27F29007		EC420EC0 8715F4C0	87144000	87144CCO 8714C7BB 4027F210	4027F210	16850136	C0871433	*X2&G@B+ GN4 GMC GMG8 X2&USA6 GM3*
1420	C08214F3		BD602200 01157406		### ##################################	8 CR 08144400	87000076	06007801	* GH3'S AN4 GN) GN14HMD G (8 <8A*
3.440	02009014		2080410F C08715F4		SSKBR 340814C2 B50113C0 H71433BR B00FWB10 27C01516	871433Bh	800FF810	27001516	*B &M-@A+ GN44HNB5AL GH3" (B&X MO*
1460		17000120	7D4E0119 17AC0126 0DAC0124 17AC0122 158F0122 14C6AD01 1923F282	17000122	158F0122	14C6AD01	1922F282	3AAC0126	**)ARPZA-(2AUPZASNIASME.ARS2B*ZAU*
1.480	19AF0126		22AF0124 26AC0119 22C08715 FCC08714 7NC08714 338C0119 14C4AF01	22008715	FCC08716	71/0/8714	33800119	14C4AF01	*RPAUSPAUUZARS GNO GD? GM34ARND)A*
1490		1926F220 16RBBGOF	F28104C0	87167000	012426C0 87146DC0	87146DC0	8715FCAC	01002000	*KW2-0' (24) 60'ZAWW GM. 6W@ZAC- *
3.400	SZIGOFFF	FF000134	SZ140FFF FF000134 0814F1AC 012630RS 01369F01 26099C01 2506R801 02AD0226	01263085	01369F01	26099001	2506RB01	02AE0226	*GNI#% A4HM1%AWOSA6-AWI*AVFRAB.BW#
1450	2EF29004	AD022641	2EF29004 AD022641 F28203BA 4027C0B7 14130134 0B156CR5 01309C02	4027C0B7	14130134	OBISACRS	01309002	2600AE01	*>2&D_GWAZBC: X GMLA4HNZSAO*BW >6*
1500	251BAC01	24253600	24253600 0025R501 139D0124 06F28246 9F012406 8F012415	13910124	06F2B246	9F012406	8F012415	71F2821C	*USZADU@ VSAL)AUF2BF-AUF(AHN12B**
1520	8E602515	6F9B0025	07F2842E		BB702400 9015188F	01241573	01241573 00971518	BC011915	*+ UN?) VG2D>BQU ANDIAUNS GNOKARN*
1540	1540 6KAE0119	26460119	26AE0119 17AB0119 15AF0119 17F20206 BC0022F2 B70FBCFF	15AF0119	17F20206	BCC022F2	BZOFBCFF	22800025	*>> ARW > ARM ~ ARM ~ ARM ~ S26 (0 #50 U*

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

560	80011915 6EAE0119	26C08714 24FFFF01	00040020	1598	B5013036 (01159010	* (ARN) > GRUSS Gr
082	01159600	1BE50113 4C010615	96008716			\$\$\$RIC FF340815	*ANT > BNT \$561 (AFN: GO GO TO THE
0951	DAR50136 AEV13038	1	15DC9E01	1	1	19800130	*+566>408+40N-1<
0.051	15DT9E01 3009AE01	30388E01 3015DFC0	87,60500	87160000 ED	E7142F00	FFFF0003	*N) FAOI) AOR+AON- GGN GO GN? FF CA
SEO	\$\$\$RTC 90010009 AE010019	BEO10DIS FRAE0119	17BC400E	BEOLOBBS 10	10006001	54081676	**6(1)A(R*A(N#)ARP@ :56.501 A4H66*
009	3402166B 2001167A	243C5C16 3DR9BOOF	F210043C	9C163D75 0	0109AF01	19248601	*480,260.0020?9 2800.561761768064
1620	620 19AE0119 24B5020D 3602167C	36021670 0F001679	16777282	10360116 78	78360216	7890FF66	*R)ARUSB(68081 090728&04085808**
1640	640 00C08716 2C3F0016 7AF2811C	24F2811C 3C001679	36011678	3802167A O	00031467	16400E00	* 50% 0:20*0 09640 5650 500 5 *
1660	660 1665167A 907F0000 02020D54	C2020D54 AE010D24	24 B50113C0 8714BB01	- 1	COCOROFF	FF340816	*0V0*** 8B(B/A(USA) 6N\$A x#AHO*
689	680 RFEDSOOF F2011E75	02136C01 0606B502	15008716	D5883027 B	BB1027F2	101EC087	*S SACKVARAGINE GURBA GONBAXAXXXXX G*
640	1600F287 OEB81027 C0101687	C0101687 C0871704	BA1027B4		13E40113	0000000	#0 26*8&X &06 GPD:&X\$A,556,46i, 6 *
929	### 7867037A 34081614 7867037A	01037000 04008700	08008715	\$\$\$RBP 95340816 FT	FF900022	074F0006	*4HOM&GC:AC® B B BNNAHD#* SG+ F*
.6E0	6E0 17017860 06F29005 4E010617		03BD0022 F281098F 002	00221700 COS716MN COS70000	9871600	00002000	*PA8F2&E+AFPC* S2AI! SP GO) G *
1700	01040020 34081718 7807037A		INPUT FIELDS INPUT FIELDS 02037000 04C08700 08C08700 0675029E B90218F	008200 00	1NPUT FIEL 375029B	. DS 890218F2	*AR -4HPG&GC:BC@ B G H G SF\$9BGZ*
720	MAINLINE CONSTANT 10002001 1721/16R5 02120087	02120087 17320087	17462005	01050520 00	1250	24020D15	*&CZAP_JERK GP2 GPFZE(FEZ(CK)UB(N*
1740		ł I	17450087	17504000 00000	1010	DETAIL SCSFSF90	** GP)18BI2&FF-(NPE GP* AA*!!&*
760	OUTPUT C2020C00 0C050DE0 175E7802	060063	17 BEBC0505 ODC	SBCOC 14	140D128C	02180515	*BB< (E(OP\$831 &P.CEC(ECCMCKCBQ:N*
780		S712FE	TOTAL OUTPUT 78 20030010 19960087	96C087 17	OVERFLOW 7BE340B 17E	17BE340B 17BB7A20	*:005BC& CGC GK#8-C &RO SP#4HP':-*
760		BOIEIB	CO 8717E2CO 871		20090087	12FB(7820	*R*11ABB(<e(ou\$ bps="" gk#8-*<="" gqc#-r="" td=""></e(ou\$>
1700		G C010179A 78FFB77B	FFD87A02	C27804C2 F3	F210037B	02020087	*C &R*8-P &P+***P**G*8B83B2&C*BB G*
03/	17193408 18098C02 04181E8C	8 04181E8C CA1518	1EBC 04151829 8C031D18 2D1	2D16300W 10	58003	220D1CBA	*PR4HGI(BDG)(cNGZ(C)G_DO(*N(CS(***
800	F0229B03 1F90C087 18B4C202	LINK TO TRANTHIC 1884C202 1908C201		18BEA004 00000201	- 1	D2CSECC4	*05\$C-4 GG4BFR4BA(C5&G* n BAKEYD*
820	CSE2C3N9 C9D7E3C9 D6D5D7C1	6263	MAINLINE 00040002	010C005F 16	1AUCTC6C	00C5377A	*ESCRIFTIONPAGE G D BAK + + * Z E7 - +
1840	01027446 03750199 75020500	750205C0 8700048B	75020500	87000482 75	750205BB	0103F290	*AB+ CSARSHE G D.SBE G DRSREGAC2&*
980	01711000 791000F2 90043CB7	90043CB7 (B7E4D01	030C99F2	81077501 03	03008718	58F2870A	*(* 9& 2&N@GG=(AC(R2ANSAC GRRSGC*
1880	C2020CB4 C0870004 8F11C201	8F118201 08003840	ob120C11	OB110D12 30	3CF00F18	0C040D17	*FB(4 6 DidBAC @ (KCJCJCK@O(GCDCP*
840	ODIBSCBF BF90C202	OCCOCCOS ODBO181B	C08717E2	C087180A CC	C087175C	18BE3408	*(0*1)&88((E(00\$ GPS 600 6P*0=4H*
0081	19653402 18DA7502	059C010D 0E2C0019	12242000	191E24C0 FF	FF19FEBB	010FF210	*RU4BG15BF*A(+X RKBX R)) #R58A126*
BEO	060C8F0C 8F0C90B8 F0007820	F0007820 01F29004	8C001C10	70000AF2 81	81483820 (OCD9F210	*F<1<1<480 3-42402 *& C26HB-(R26*
006	411C0119 3R0A1C01 19	193F0AB9 FF12F210	06880112	F20415B9 FF	FF10F210	OKEROLIO	*A*AR#C*AR#C9#K2&F*AK2BN9#K2&F*A&*
920	F20409BD 480EF201 19F2870E	19F2870E BD480EF2	B1083C3B	193A3C3B 19	193E3A20 (00083420	*2DI*H+246K26+*H+34H04K+04K+1-(0:+*
940	940 OCD7BB4D OFF20209 B9010EBD	B9010EBD 440EF211	0AC2020E	B4C08700 048F4075 020EC201	18F4075)ZOEC201	*{P*(+2PT90+*D+2JCB8<4 B1 5B+R0*

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

D9C4E240	0001 B9C5C3D6 B9C4E240 E7C5 C440C6C9 B3C548PA	0001 B9C5C3D6 B9C4E240	D9C5C3D& D9C4E240		_ <u>ଲ</u> ଅ ର	EACSN9CS CLOSE 20C3C087	40D9C5C1	C440C6B9 0C997502	*< GP. CA ARECORDS WERE READ FR* *OWTHE INDEXED FILE.:-C GR,5A <r58*< th=""></r58*<>
05C08700 0483C087 00048434 08191A5C 8F8F90C2	8434	8434		BFBF90C2 0	i 0	2000000	OSODBO	19 6BBC0202	C G DD4HR**11&BBC
19CO OK188AFO 02980200 908C1519 19818C10 2E1992CO 8	19818C10 2E1992C0	19818C10 2E1992C0	19818C10 2E1992C0	2B1992C0	₩.	7180AC0	87180ACO 8712FEF2	870F5B5B	*(Q:OESE &(NRRA(L,RK GQC GK#26)\$5*
61F7 F034081A 012C0119	40F0F361 F1F861F7 F034081A 012C0119	61F7 F034081A 012C0119	F034081A 012C0119		3.	FC1BC087	00048052	D402C087	*BIMC 03/18/704H*AZAR@\$ G D KMB G*
IACO IBERCACO COCCOCOC COCCOCO COCCOCO COCCOCO COCCOC	DF 8AMPL2) 60060606 06606060 6000860 6000860 0	0 00000000 000000000 000000000	00000000 00000000	00000000	Q	00000000	00000000	00000000	**
1A20 00000000 0000000 00000000 0000000 0000	00000000 00000000	00000000 00000000	00000000 00000000	00000000	Q	00000000	00000000	00000000	*
1800 00026200 00AEG100 24B50113 C0870000 0100E2E2	C0870000 0100E2E2	C0870000 0100E2E2	C0870000 0100E2E2	01006262		FFFF3408	18316000	291 DF 282	* BB)A(U5AL G A 55##4H\$1AZ)2B*
1820 ODB9FF2B F21407BC 700EAC03 2826C087 00009C01	AC03 2826C087	AC03 2826C087	28260087	000009501		00099501	00009001 0009AE01 00198601	ODIBADAE	*(9#,2MB@0+%D,W G *A(I)A(R+A(\$()*
1246 C11917BC 406ER501 OBR51009 0001210A 04070409	00012108 04070409	00012108 04070409	00012108 04070409	04070409		09070409	04040204	02040404	*ARP@ +5A.5AI A/CDGHIIGDIRDREDBDD*
1B60 07020402 04260204 07070464 04FE2409 0C040764	0464 04FE2409 0C040764	0464 04FE2409 0C040764	04FE2409 GCG4G7G4	60040764		C41B0404	04080707	04040704	*IRDBDWBRGGBRRD=:UT <brdsdbrd#< td=""></brdsdbrd#<>
1880 (%020404 04070469 OF041310 OGFE0704 05050C13	OAFE0704 05050011	OAFE0704 05050011	OAFE0704 05050011	05050011		0204070F	04070913	04260604	*producing for redeferment (turben*
1360 47050505 05080502 02208080 SUFEBORO SOSOBOBO	SDFEB080 80808080	SDFEB080 80808080	SDFEB080 80808080	30808080		80308080	80868980	80808016	*GEEEE.CAB- == 0*
3800 04070404 04070415 18040405 11FE0804 10140H0A 04141604				10140100		04141604	OCO90ROB	040C1515	*ngnnngchsnnE.J=Hn&M <cnmnn<t.,,n<nw*< td=""></cnmnn<t.,,n<nw*<>
IBEC 15040404 05641564 02070408 04FE0704 04040204	0408 04FE0704	0408 04FE0704	04FE0704	04040204		021D25FF	000000000	00000000	*NDDDECNDBGDHD=GDDDBDB)U#
ACON ZEDO8715 70746278 E2020774 62325202 0174027B	0232F202	0232F202	0232F202			70:073302	02100390	07078FC0	*> G*04E&SFG4F2SB44B*@G3FB*C*GG1 *
1620 87000469 B2027460 8718F2F2 8703F287 29C08718	FZF2 8703F287 29C08718	FZF2 8703F287 29C08718	8703F287 29C0871B	29C0871B	_	FE34081D	S35E0034	37706034	*B DZKB: G*22GC2GZ G5=4H*L; 47*-4*
1040 F281175E 003537E2 02017D0G 35F2840A C0870000	7D08 35F2840A C0870000	7D08 35F2840A C0870000	35F2840A C0870000	00870000	, .	70042EF2	87167602	2EF28710	*2AP} 57SBA*H52DC G @D>2GB@B>2G&*
1060 70012EF2 870A7BF0 2E740232 50003335 C2010D04 0	50003335 C2010D04	50003335 C2010D04	50003335 C2010D04	C2010D04	_	02021705	80060610	87008714	*@A)25c\$O)4B2* 35BA(DBBPE(FF*6 GM*
1080 E140040E 05090900 60000001 03010303 40040909	03C1E3E3 40C4C9E9	03C1E3E3 40C4C9E9	03C1E3E3 40C4C9E9	40040919		40E2E8E2	E3C5D440	C6F1L9F1	*/ IIII - ACALL DIR SYSTEM FIR1*
1080 C4F219F2 F0F0F0F0 F3F9F8F0 F9F9F9F9 F9F9F0F0 F0F0F4F0 F5000040				F9F9F0FQ		FOFOF 4FO	F5000040	40404040	*F2R2000039809999990000405 *
1880 40404040 40404040 40404018 F2000000 00001888	F2000000 00001CBC	F2000000 00001CBC	F2000000 00001CBC	00001080		DESESBE	C2C4E3E2	06585809	* *@O\$\$RBDT\$O\$\$R*
1CEO CZEZESEZ FOFOFOFO FOFOFBFO FOFOFOF FOF11604	FOFOFOFO	FOFOFOFO	FOFOFOFO	FOF11504		0404240E	07071307	7080804	*BSWS00000000000010DDDD1+6GL6*HHC*
1100 F2870E5E SEU9C2D3 D7F0F761 F0F761F7 F034081E	SEN9C2D3 D7F0F761 F0F761F7	F0F761F7	F0F761F7			2C34011E	1F34021E	23C2011E	*26+\$\$RBLP07/07/704H\$%4A}~4B}TBA}*
ID20 B56C0A1C OAC2021F 005C0003 137B0F12 796012F2 1	00003 13780F12 796012F2	00003 13780F12 796012F2	137B0F12 796012F2	796012F2		0E67940	10E67940 12F2100A	50011915	*5%c*cBB+ * CL#!K9-K2&W9 K2&c*ARN*
1840 70001A7E 40125001 0619F280 0830871B 4EC0871F 6	083C871D 4BC0871E	083C871D 4BC0871E	083C871D 4BC0871E	4BC0871F		89100011	661A7C00	21C2021F	*@ †\$ K*AFK2 H@G). G;I*)W†@ /BB-*
1550 0075011C E202069C 000000C0 871E2B0E 001EB61E 1	871E2DOE 001ED61E	871E2DOE 001ED61E	871E2DOE 001ED61E	COLEDELE		D23D601E	D6F2814D	50000100	* SA*SB * GJ+ #D#K*-#02A() A *
1386 F28107D2 0101C087 ID67BC01 00E20101 0E001ED6	'BC01 00E20101 0E00JED6	'BC01 00E20101 0E00JED6	00E20101 0E00JED6	0E001ED6		16523560	1EF6F281	285100001	*246KAA G)X@A KAA+ \$0\$K"-;02AY) A*
LUNG 00F20119 RE00001E D2BDZF00 C0011DBD 0E001ED6	сосітвя	сосітвя	сосітвы			15023060	1ED6F281	0BC0871E	* 24R+ JK'# A)(+ JOFK'-1024H G;*
THALL ZECOSZIE GECOSZIE ZEBCOGOG S40ZIEBB OCOGIECE	n SSCOSTIE ZHBDOOOG 340Z1EB8 OCOOLECF	ZHBDOGGG 340Z1EB8 OCGG1FCF	34021EDB OCOONECF	OCOGAECE		JEDSZDFE	JEDSZDFE JEDSFZOJ	250E001E	*_ 6)C 63@ 4R\$Q< \$1\$Q?=\$Q2AU+ 3*
ibro chiebssa doiecers 900doed1 ifceiebs obollece leccrsq1 ocacolo1				ore11ECE		IECCF281	00800101	1ECE3000	*+\$1_8~\$+2&F+6\$+\$N(A5+\$<24< <a45+0 *<="" td=""></a45+0>

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

0101 H 008 SAMPI 1 0102 F***************************** SAMPL 1 0103 F* SAMPL1 0104 F* THIS PROGRAM -SAMPL 1 0105 F* SAMPL1 1. LOADS 100 RECORDS TO AN INDEXED FILE. 0106 F# SAMPI 1 0107 F* SAMPL1 2. READS ONE RECORD FROM FILE \$SOURCE FOR 0108 F* SAMPL 1 0109 F* INPUT. THE FILE \$SOURCE IS BUILT WHEN SAMPL 1 01091F* SAMPLE PROGRAM SAMPLE IS COMPILED BY SAMPL 1 01092F* GIVING A RETAIN-T PARAMETER TO THE SAMPL 1 01093F* FILE \$SOURCE. SAMPL 1 01094F* SAMPL 1 3. CREATES THE DUTPUT DATA USING A 0110 F* SAMPL 1 0111 F* LOOP IN THE CALCULATION SPECIFICATIONS. SAMPL 1 0112 F# SAMPL1 0113 F# 4. USES KEYS FROM 000005 THROUGH 000500 SAMPL1 0114 F* IN INCREMENTS OF 5. SAMPLI 0115 F* SAMPL1 5. SHOULD BE FOLLOWED BY SAMPLE PROGRAM 2 0116 F* SAMPL 1 0117 F* TO VERIFY THAT THE FILE WAS PROPERLY SAMPL1 0118 F* LOADED. SAMPL1 0119 F* SAMPL 1 0120 F********************************* SAMPL1 0121 F\$SOURCE IP F 96 96 0001 DISK SAMPL1 0122 FDISKOUT 0 F 256 128 06AI 0002 1 DISK 01 SAMPL1 0123 FPRINTER O 96 96 PRINTER SAMPL 1 0003 0004 0201 ISSOURCE NS 01 SAMPL1 0005 1 2020 1 NODATA SAMPL 1 0006 Z-ADD0 COUNT SAMPL1 0301 C 01 60 0007 0302 C Z-ADDO SAMPL 1 01 RECNBR 30 SAMPL 1 8000 0303 C REPEAT TAG 01 ADD 5 COUNT SAMPL 1 0009 0304 C COUNT SAMPL 1 0010 0305 C 01 RECNBR ADD 1 RECNBR COMP 505 SAMPL1 0011 0306 C 01 Truos 02 EXCPT SAMPL 1 0012 0307 C 01N02 0013 0308 C 01N02 **GOTO REPEAT** SAMPL 1 SETON LR SAMPLE 0014 030810 0309 CLR RECNBR SUB 1 RECNBR SAMPL 1 0015 SAMPL1 0016 0401 OPRINTER T 1.8 204 20 *SAMPLE PROGRAM 1 HAS* SAMPL1 0017 0402 0 27 'LDADED' SAMPI 1 0018 0403 0 0019 0404 U RECNBRZ 31 SAMPL1 39 'RECORDS' SAMPL1 0020 0405 0 0021 0406 D 61 "INTO AN INDEXED FILE." SAMPL1 0022 SAMPL 1 0408 0 LR SAMPLI 0023 0409 0 21 'KEYS ARE IN ASCENDING' 42 *SEQUENCE STARTING AT* 0024 0410 0 SAMPL 1 64 *000005 AND INCREASING* 0025 0411 0 SAMPL1 0026 0412 0 84 'IN INCREMENTS OF 5.4 SAMPL1 0027 01 LR SAMPL 1 0028 0414 0 21 "SAMPLE PROGRAM 2 WILL" SAMPL1 0029 44 *PRINT FROM THE INDEXED* SAMPL1 0415 D 65 'FILE TO SHOW THAT IT' SAMPL1 0030 0416 0 86 'WAS PROPERLY LOADED." SAMPL 1 0031 0417 D 0032 0501 ODISKOUT E 01N02 SAMPL 1 0033 COUNT SAMPL1 0502 0 94 *RECORD NUMBER* SAMPL 1 0034 0503 D RECNBR 0035 0504 0 128 SAMPL 1

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 NDDATA 001 0100
0006 CDUNT 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		8000	RGSU8\$	SUBSEG
	177F		0027	\$\$CSIP	5444 CONSEC INPUT
	1746		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	\$\$SRDE	SYSTEM SUBR
	195D		002F	\$\$SRBP	SYSTEM SUBR
	198C		000B	RGMAIN	TOTAL CALCS
	1997		0022	RGMAIN	INPUT FIELDS
	19CA		004B	RGMAIN	DETAIL CALCS
	1905		0005	RGSUBS	CONSTANTS
	1 A 4 5		0043	\$\$PGRI	RESET RESULTING INDR
	1415		0030	RGSUBS	EXCEPTION
	1989		0000	RGSJBS	SUBSEG
	1488		0059	\$\$10UT	5444 INDEXED DUTPUT
	1875		001C	SSRDF	SYSTEM SUBR
	1AE1		0094	\$\$ SRBI	SYSTEM SUBR
	1B91		000B	RGMAIN	TOTAL DUTPUT
	IRC5		0024	RGMAIN	LR & OVERFLOW PROCESSING
	1848		0010	RGSUBS	OVERFLOW SUBSEGMENT
	1890		G00¢	RGSUBS	SUBSEG
	1869		00F8	\$\$LPRT	5203 PRINT
	1CF4		0072	RGMAIN	DPEN
	1EFE		002D	RGMAIN	CLOSE
	1FDC		3000	RGSUBS	CONSTANTS
	1D56		009D	RGSUBS	DUTPUT CTRL RTN
	1DF3		00E9	RGSUBS	CONSTANTS
	LEEA		0014	RGSUBS	LR CALCS
	1F2B		0076	RGSUBS	LR PROCESSING
			04001	SAMPL 1	TOTAL CORE USAGE REQUIRED TO EXECUTE
EL6	EDP 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)

	**************************************	***************************************	**		***************************************	*	* · · · · · · · · · · · · · · · · · · ·	***************************************	**************************************	**************************************	T*	**************************************	# (Co o o o o o o o o o o o o o o o o o o		***************************************	*	•	***************************************	
_	*	*	 ;	*	; T =	*	*	•	;	*	<u>;</u>	•	*	;	*	•	*	*	*
	00000000	00000000	00000000	00013560	06000000 SECONDARY	OOOOOOOO	00174410	80100002	04861200	00008C3C	80BC0002	0005E000	12661270	00000000	00000000	40404040	40404040	00000000	00000000
	00000000	00000000	110A110A	00000003	000000000	00000000	00010012	003A0000	118107E8 (33008000	BOFFFFBA	00068A5C (00001100	000000000	00000000	000000000 00000000 40404040	40404040	18640000	000000000
	00000000	00000000	00010002	43400000	00000000	00000000	01090000	11810000	4801FFFF	E409C301	16010000	BA00152C	04126000	00000000	00000000		40404040	11520087	00000000
	00000000	00000000	CONSTANTS	C0871694	00000000	00000000	0000003A	10CB2 00041141	0300A0E3	01586206	40141614	15420000	0000E000	00000000	00000000	00000000	40404040	11521300	00000000
	00000000	00000000	00000000	CO871EFE	00000000	00000000	11301152	00100000	00010900	BC 00C 100	B60000C5	00840000	40000000	00000000	00000000	00000000	40404040	000002A5	00000000
ŀ	00000000	00000000	000000000	00000000	00000000	00000000	00000CE00	00000000	00003401	0060FFFF	BE07E804	01848000	04860000	00000000	00000000	00126600	40404040	13160020	00000000
4	00000000	00000000	000000000	000000000	00000000 ATORS	00000000 00000000 00000000 00000000	00000000	00800000	0A11BE00 00003A01	13000100	41115211	BA800173	FFFF07E8	5F3C6000 00000000 0000000 0000000	00000000	E000127C	40404040	00000000	00000000
PRIME WORK AREA	1003 COB71CE4 00000000 00000000	00000000 00000000 00000000	1.080 000 <u>0</u> 0000 00 <u>000000</u> 0000000 00000000	1040 0000000 00000000 0000000 000000	1000 02000140 00000000 00000000 00000000	00000000	1100 00000000 0000000 0000C200 11301152	1120 0000000 0000000 0000000 0011	1000000 00000011	1160 44801300	DTF2 1180 60AUE 384	11AU 0201E2D2 BA800173 01BA8000 00BA	11CO 40C31181 FFFF07E8	5F3C6000	1200 00000000 00000000 00000000 00000000	1260 127C0000 E000127C 00126600 00000000	1280 40404040 40404040 40404040 40404040	1300 BC00F0A1 00BC0000 13160020 0000	1320 00000000 00000000 00000000 0000
PRIME	1000	1020	1080	1040	1 00 0	1050	1100	1120	1140	1160	1180	1140	1100	1160	1200	1260	1280	1300	1320

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

-	**************************************	***************************************	***	***************************************	***	***************************************	***	***	***************************************	***	***************************************	**	# .p	**************************************	*. K B	*. CCM*	*A	*E2	*B*	*	*B
00000000	00000000	00000000	80001420	00000000	00030000	00000000	0000+000	00000000	00000000	11811520	FFFFFFF	FFFFFFF	00 000001	8F407A80	C378FFC4	1000001E	1759C087	00E20100	01007510	16E6C087	L FIELDS F2101535
110 11100000 00000000 00000000 00000000	00000000 00000000	00000000 00000000 00000000		00000000 000000000 00000000	000000000	00000000 00000000	00000000	00000000	00000000	004001DA	FFFFFFF (FFFFFFF	OD12COFF (C0870004			E0C3C087	00108801	871725BA	87080087	367820C3 F2101535
00000000	10000200	00000000	00004408 40A000BA	00000000	00000000	000000000	00000000	00000000	00000000	15420020	FFFFFFF	FFFFFF		C2021084	INPUT MAINLINE	75029988 1000F210 150C0117		F29003BC	F21004C0	0A2 20AF 2	BOCO8717
11100000	00630950	00900000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	000840000	FFFFFFF	FFFFFFF	02059C00 0F189C01	00F2810D	DISK INPUT 35101643	75029988	00F29006 7AFFC47A	09880118	42B88000	01000090	61000863
00021110	00000000	00000000	00000000			0000000	00000000	00000000			FFFFFFF	FFFFFFF	E2010075	440EF282	LINK TO C201177F	168E1722	170AB880	00363017	1716	081732C2	17288202
FF000001	16FE4340	00000000	00000000	14161181	00000000 00000000	00005000	00000000	00009000	00000000	00000000	FFFFFFF		ROUTINE 34011650	F281098D	00001800	16040001	80010010	1709F287	8F007408	85101434	17180087
000040FF	0000C087 1E6EC087 16FE4340 00000000	00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000	00000000 0000000 14161181 00000003	000000000	0000000 00005000 0005000 0000000	00000000 00000000 00000000 00000000	00000000 00009000 00000000 00000000	00000000 00000000 00000000 00000000	00000000	FFFFFFF	FFFFFF FFFFFF	1 CONTROL 34081688	0EF 28219	00020200 C2011000 C0870000 C201	20030010	89C100B9	18053032	C0870004	8CB50112 85101434 08173	1000001
13A0 00000000 000040FF FF000001 00021	0 00 0 C 0 8 7	00000000		00000000	00000000 00000000	00000000	00000000	00000000	00000000	1520 0C000000 00000000 0000000 BA5C40A1	1540 0000FFF FFFFFF FFFFFF	FFFFFFF	1640 FFFF 1644 3408168B 340116	1660 12008D41 0EF28219 F281098D	00020200	16A0 7A40C378 20C3C010 16D40C01 16BE	16C0 01168E97 89C100B9 8001C010 170AB880	16E0 198CCO87 1BC53C32 1709F287 0D3C3	1700 C2021084 C0870004 8F007408	30172416	16EDC201 1000C087 171BC087
13A0	1300	1360	1400	1420	1440	1460	1480	1440	1400	1520	1540	1560	1640	1660	1680	1640	1600	1650	1700	1720	1740

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

TBN 7802 C9 JC F2 90 0C ZAZ 04 50 11 06 19 05 ZAZ 04 20 11 89 19C5 + TBN 78 \$2 C9 JC F2 90 2C AZ 06 50 1106 19 C6 AZ Ø6 20 1109 1EDC ZAZ 44 A5 10,11 06, SZ 47 DZ 18 1909 BC CØ 87 1A45 DATA 8184 C9 00 TBF 79 04 69 JC F2 9008 BC C087 1A15 BC C0 87 190C SBN 7A 20 C3 BC C 87 16 94

**************************************	***************************************	# · · · · · · · · · · · · · · · · · · ·	**********************************	*******************************	*2SS	**************************************	*	***************************************	***************************************	***************************************	#	***************************************	*2	***************************************	*.2*	***************************************	*2*	*BB	******05505**12***E****E***	*	*I2	***I2******************************	***************************************
1760 02109888 01002C01 176D1A7A 00007880 D9F29004 C08716DE 7A80D9C0 8716E284	FACO02A 28AD022A 2E8C002A F20407BC 420EC087	1740 1746C087 18563408 181C8501 13C08719 25B0800F 881027C0 1518E2AE 011917AC	17CO 01200DAC 012417AC 0122158E 0122181E AD011922 F2823AAC 012619AF 012622AF	17E0 012426AC 011922CO 871861CO 8718E2CO 8719258C 0119181E AE011926 F22016BD	\$\$\$\$BUA 1800 800FF281 04C08718 E2AC0124 26C08717 C7C08718 61AC010D 20C08700 00FFFF34	012817E2 012389FF 2AF2100C 8F002A18 44C08719	\$\$\$RTC 1840 C0870000 019C010D 09AE010D 19800AE01 1917BC40 0EB5010B B5100900	\$\$\$RMO 1860 01340818 DB340218 D02C0118 E1243C6C 18A28980 OFF21004 3C9C18A2 750109AF	1880 01192486 0119AE01 19248502 0D360218 DF0F0018 E018DCF2 82103601 18DD3602	18A0 18DD6CFF 0000C087 18913D00 18E1F281 1C3C0018 E0360118 E1360218 E10C0318	18C0 CC18A50E 0018CA18 E16C0000 00C20200 00AE010D 24850113 C0870000 0100FFFF	\$\$\$R\$B 18E0 0000 3408 1924BD80 OFF2011E 7502136C 010606B5 0215C087 195DB810 27B81027	1900 F2101EC0 87193DF2 870E8810 27C01018 ECC08719 448A1027 84011175 01138401	1920 13C08700 00340819 5CC08700 0C7C000C 780102F2 90238C41 0EC08718 593C0119	1940 53F28704 3C021953 3408195C 7C000478 07037A00 03C08700 08C08700 00340819	1960 879C0022 074E0006 19897860 06F29005 4E010619 8BBD0022 F281098F 00221988	1980 C0871965 C0870000 01040020 7820C3C0 101F05C0 871B9175 029BB902 18F2100C	1940 2C0119AB 1EB50212 C0870000 C08719CA 2C001100 00C C8719 ACC2021A 88C20111	19CO 3035101D 57F0F5F5 F0F57802 C9F2900C 04501106 19C50420 110919C5 7802C9F2	19E0 902C0650 110619C6 06201109 1EDC44A5 10110647 D21019C9 C0871A45 8104C900	1A00 7904C9F2 9008C087 1A15C087 19DC7A20 C3C08716 9484081A 445C8F8F 90C20210	1A20 003C4011 907904C9 7802C9F2 90138C05 0511068C 0C5D1EE9 8C027F11 09C08719	1840 89C08700 007408FF 7402FD75 02FF7404 FBF21000 89FF00F2 10227504 FB2C001A

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

	*******************************	*******************************	***	***************************************		******************************	* 2 2	***************************************	*2EBZ*	*B*	*	# * * * S + * * * * * * * * * * * * * * *	#	#	**SZ*	*	#	*2	*7.2N	*BB	***********************************	*22	*B 2	*. 2	*S7.2*	
A00 003C7A1A 6EE20203 C0871A51	B4080984 01088840 03F29032 88012788 0127F210 293CFF1A	1AB432B5 010DB601 3C1D0000 0000F282 0DBC600E F28103BC	181FC087 18758070 0EC08118 59C0871A E1C08717 A6C08718	3AF2027E 850136C0 8719253C FF18222E 00182238 750109AC	01223601 1872AC00 2E228E00 2E1B742C 011B250D 2E011B25	03F29003 B40132AC 002526AF 01241B6C 020325BC FF22AE00	088E0022 1874F220 15850136 6C01062D C0871944 BC002EE2		TOTAL OUTPUT DEAC032B 26C08700 00/820C3 C0101F16 C08718C5 C20218E9	OVERFLOW SEGMENT 3408188F 7A20D90C 0411D118 C4C08718 9C7B20D9 C0870000	LR & OVERFLOW PROCESSING 007820C3 C0101F1A 780107C0 1018A878 FFD778FF D87A02C2 7804C2F2	\$\$LPRT 9.7840809 8401088C 400EE201 108A0414 C0871C4E E20111C0	1CDEF1E2 00C0871C 872C001C 38220C02 1C371C38 3C001C36	71E601B5 011870E6 0185010D 1C000000 00E20112 8A0614C0	E20113F2 87098408 217D0000 F2815A9C 001500BB 58162C00	0000C087 1C87B0E0 18850118 AC00241A B80814F2 9004AC00	00006C01 0515C1E0 1CB75C01 0305B802 14AE0024 15B80414	002423F2 04038C48 0EBB0514 B5102134 081CDDB0 E31BB9E2	011CD51D BA1002CO 87000480 000002BD 410EC001 00008501	00C20110 005F18DA DA6C00C5 377A01C2 7A40C375 01997502	05C08700 04827502 05880103 F2900D7B C0007910 00F29004	99F28107 750103C0 871D0EF2 800AC202 1084C087 00048F11	3CF (1109 0C071108 1109C087 1694)D58 34081DF2 34021D74	AD002423 F204038C 480E8805 14851021 34081D9E B0E318R9	1AF21010 2C011D96 1DBA1002 C0870004 80000002 BD410EC0 011D2FB5	
	02 FD7510FF	11A B2382C01	18590087	1BI 11B 69AD012D	3AE 00222E86	000 000 08840	500 2238F2A0	195DC 087	3F2 14078C70	rER OUTPUT 141 35101057		302 C2C08719	EBB 400FC090 1CDEF1E2	:15 01103701	BC 01211C09	+3A 041C68F1	:01 1C8915F3	4AC 002415AD	•	309 C0870004	700 04887502	034 40010310	000 30401100	304 AC 002415		
	1480 02FF7502	1AA0 822E001A	1AC0 620EC087	\$\$\$RBI 1AE0 5634081B	1800 012238AE	1820 3640000	1840 222EAE00	1860 0127C087	1880 89FF28F2	LINK TO PRINTER OUTPUT 1840 C2011141 351010	1800 00060000	18E0 10037802	1000 87104688	1C20 85011F1E	1640 87164686	1C60 1C68143A	1080 24182001	1CA0 F29004AC	1CC0 1889F71A	1CE0 08851009	1000 05008 100	1020 36871034	1040 C2011000	1060 14F29004	1080 E21BB9F7	

[|] Figure 4-13 (Part 4 of 5), Storage Dump of Sample Program SAMPL1 (Models 10 and 12)

*.222222	*60P22B.*	# · · · · · · · · · · · · · · · · · · ·	*SAMPLE PROGRAM 1 HASLOADEDRECO*	*RDSINTO AN INDEXED FILE.KEYS ARE*	* IN ASCENDINGSEQUENCE STARTING A*	*T000005 AND INCREASINGIN INCREME*	*NTS OF 5.SAMPLE PROGRAM 2 WILLPR*	*INT FROM THE INDEXEDFILE TO SHOW*	* THAT ITWAS PROPERLY LOADED.IREC*	*DRD NUMBER	*C*	**** T = L = = = = = B = = = = = = = = = = = =	# = · · · · · · · · · · · · · · · · · ·	**************************************	**************************************	***	*	*** C. * *Q. *** *** ** * * * * * * * * * * *	**************************************	**************************************
1DA0 12F20415 89FF10F2 1006BD00 10F20409 BD480EF2 0119F287 0EBD480E F281083C	110AC202	4000000 02400000	C1C4C5C4 D9C5C3D6	D2C5E8E2 40C1D9C5	C1D9E3C9 D5C740C1	3 D9C5D4C5	9 03030709		CONSTANTS	CLOSE 7 00007A20	10997502	LR PROCESSING LR PROCESSING LR PROCESSING	37008718	3 871B9C0C	DBC08718 9CC08700	1FA0 00000000 00000000 00000000 00000000 0000	2000 10150C01 17890000 1E011737 9789C100 898001C0 10178388 8000F290 067AFFC4	1 18F29003	0 00F21004	501 12851014 340817A8 C2010000 9C0A220A
0EBD480	8D440EF2	4000000	C1C4C5C	D2C 5E 8E	C109E3C	40C9D5C	F240E6C	40E3D64(C4C5C441	1EDCC 08	1F2B3501	04110118	8C143C1E	53168800	DBC0871	0000000	8000F29(17828801	16ACB88	C201000
0119F287	09B9010E	87000040 04000002	1D4 40F140C8 C1E2D3D6	C5C440C6 C9D3C548	C540E2E3	CSC1E2C9 DSC7C9D5 40C9D5C3	D5E3E240 D6C640F5 48E2C1D4 D7D3C540 D7D9D6C7 D9C1D440 F240E6C9	1EA0 C9D5E340 C6D9D6D4 40E3C8C5 40C9D5C4 C5E7C5C4 C6C9D3C5 40E3D640	400306C1	C3F29006 07201109 1EDCC087	1F00 C33C801F 13780809 7A0809F2 100CC087 1EEAC087 1891C087	02100000	06261E22	16758C12	8C13551E	00000000	10178388	87003630	BC001CBB 0100E201 00C20210 B4C08700 048F0074 083CC087 16ACB880	340817AB
8D480EF2	400EF202		40F140CB	C5C440C6	40090540 C1E2C3C5 D5C4C9D5 C7E2C5D8 E4C5D5C3 C540E2E3	C5C1E2C9	D70906C7	CSE7CSC4	C50903E8	C3F29006	1EEAC087	3 8F8F90C2	021C908C	1F60 9C0C0411 D110FC8C 14141E4C 8C13291E 608C143F	1F80 0411011E 018C1414 1E9D8C15 2B1EB3BC 13401EC7	00000000	00100868	321782F2	048F0074	12851014
10F20409	10C0 3810C73C 3810C800 00100600 00100780 400EF202	01100000	C709C1D4	D9C4E2C9 D5E3D640 C1D540C9 D5C4C5E7	C7E2C508	60825063	07030540	40090504	70906070	ALCS 1EFD7820	100CC087	PROCESSING 081FA05C	BAFO1E98	8C13291E	281EB38C	00000000	9789C100	871708CO 87187ECO 8718AD3C	84008700	16F68501
10068000	00100600	75020EC2	40070906	C1D540C9	05C4C905	C1D5C440	4BE2C 1D4	40E3C8C5	E6C1E240	LR C4 C5D93408	7A08D9F2	LR 00048434	021E1109	14141E4C	1E9D8C15	00000000	15011737	87187EC0	00020210	103C179D
89FF10F2	38100800	00048F40	04070305	05630640	C1E2C3C5	FOF OF 540	06C640F5	C6090604	E340C9E3	05E4D4C2	13780809	04830087	1A1E188C	D110FC8C	01801414	AMPLE 4 000000000	17890000	8717D8C0	01005201	BA010075
2F20415	18100730	1084C087 00048F40 75020EC2	1E00 0100E2C1 04D7D3C5 40D7D9D6 C7D9C	09C4E2C9	0000000	E3F0F0F0 F0F0F540 C1D5C440 C9D5C	35E3E240	3905E340	10£3C8C1	LR CALCS 1EE0 D609C440 D5E4D4C2 C5D93408 1EFD7	33CB01F	0928005	1F40 1E158C05 1A1E188C 021E1109 BAF01	3000411	34110116	000000000	10150001	2020 7AE0C3C0	3C001CBB	2060 C087179E BA010075 103C179D 16F6B
10A0 1	1000	1050 1	1E00 (1E20 (1E40 4	1E60 6	1E80	1 E A 0 (1500 4	1EEO C	1F00 (1F20 (1F40	1F60 c	1F80 (1FAO (2000	2020	2040	2060 (

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

	0101	н	800											SAMPL2
	0102	F****			****									C. 1 110 1 2
	0102		****	****	****	****	*****		****		****			SAMPL 2
		F* TH		0004								*		SAMPL2
	0105		13 PK	UGRAM	_							*		SAMPL2 Sampl2
	0106		. Mil	ST RE	DD EC	EDEN	BY SAM	101 6		SPAM 1		-		SAMPL2
	0107						DEXED			SKM11 I		*		SAMPL2
	0108		W ri	ICH L	UADS	474 114	DEXED							SAMPL2
	0109		2. RE	ADS A	א זאח	EXED	FILE S	FOL	IENTIA	ALLY.		*		SAMPL 2
	0110													SAMPL2
	0111		. US	ÉS A	BLOCK	L ENG	TH FOR	DI	SK W	HICH				SAMPL 2
	0112						THAT					*		SAMPL2
	0113	F*	LO	ADING	THE	ILE	IN SAM	PLE	PRO	GRAM 1	. •	*		SAMPLZ
	0114	F*										*		SAMPL2
	0115	F* 4	4. CO	UNTS	THE N	JMBER	OF RE	COF	RDS RE	EAD SO)	*		SAMPLZ
	0116	F*	TH	AT TH	E USEI	R CAN	QUICK	LY	VERIF	FY THA	T	*		SAMPL2
	0117		10	D REC	ORDS 1	IERE .	LOADED					*		SAMPL2
	0118											*		SAMPL2
		F****								*****	****	**		SAMPL2
0001		FDISK						1	DISK				01	SAMPL 2
0002	0121	FPRIN	rer o	F	96	96	0F		PRIN1	ER				SAMPL 2
0003		IDISK	IN N	5 01	1 (.0								SAMPL2
0004	0202									1	6 K			SAMPL2
0005	0203	_									94 D			SAMPL2
9006	0204	1								126 1	280R	ECNBR		SAMPL2
0007	0301	C 01	l	cou	NT	ADD	1		C	JUNT	30			SAMPLZ
		•			•••		-		•					
8000		OPRINT		204	I P									SAMPL2
0009	0402	_	OR		OF				_					SAMPL2
0010 0011	0403									KEY.		taus		SAMPLZ
0012	0405	_								DESC		LUN		SAMPL 2
0012	0406	_					DACE	,		"PAGE	•			SAMPL2
0014	0407	-		1	01		PAGE	Z	35					SAMPL2 Sampl2
0015	0407	-	U	1	UI		KEY		6					SAMPL2
0015	0409	-					DESC		21					SAMPL2
0017	0410						RECNB	Q 7	25					SAMPL2
8100	0411	_	Ŧ	3	01 LR				23					SAMPL2
0019	0412		•	,	-1 -K		COUNT	7	3					SAMPL2
0020	0413	_					550111	-	_	* RECO	RDS	WERE READ (FROM!	SAMPL2
0021	0414	_										XED 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

				4.50 F 110.105 A5 50.011 -0.05
START	NAME IF	CODE		CORE USAGE OF RPGII CODE
ADDR	OVERLAY		NAME	TITLE
1000	UVERLAT	LENGTH 06AC	BCBGGT	2007
16FE		ODAD	RGROOT	ROOT
179E -			RGMAIN	INPUT MAINLINE
1708		003A	RGSUBS	RECORD ID
		0026	RGSUBS	CONTROL FIELDS
16AC		004A	RGSUBS	INPUT CTRL ATN
16F6		8000	RGSUBS	SUBSEG
17FE		003B	\$\$ISIP	5444 IDX SEQ INPUT
1839		0079	\$\$SRBR	SYSTEM SUBR
1882		0038	\$\$SRDI	SYSTEM SUBR
18EA		006D	SSSREC	SYSTEM SUBR
1957		007B	\$ \$ SRRC	SYSTEM SUBR
1902		0029	5\$SRRI	SYSTEM SUBR
19FB		001C	SSSRTC	SYSTEM SUBR
1A17		0081	\$\$SRMO	SYSTEM SUBR
1 A 9 8		0043	\$\$SRSB	SYSTEM SUBR
1 ADB		002 F	\$\$SRBP	SYSTEM SUBR
1 BOA		002C	RGPAIN	INPUT FIELDS
1837		0010	RGMAIN	DETAIL CALCS
1B36		0001	RGSUBS	CONSTANTS
184C		0032	RGMAIN	DETAIL OUTPUT
1847		0005	RGSUBS	CUNSTANTS
187£		8000	RGMAIN	TOTAL OUTPUT
18AD		0024	RGMAIN	LR & OVERFLOW PROCESSING
1B89		0024	RGSUBS	OVERFLOW SUBSEGMENT
1 BF 4		008E	RGMAIN	OPEN
1BDD		0017	RGSUBS	CONSTANTS
1BD1		000C	RGSUBS	SUBSEG
1CAA		OOFB	\$\$LPRT	5203 PRINT
1082		002B	RGSUBS	SUBSEG
1E6E		0021	RGMAIN	CLOSE
1E42		Q02C	RGSUBS	CONSTANTS
10A5		009 D	RGSUBS	DUTPUT CTRL RTN
1E8F		0030	RGSUBS	LR PROCESSING
		03775	SAMPL2	TOTAL CORE USAGE REQUIRED TO EXECUTE
ELGEDP 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)

#000065 RECORD NUMBER 13 *	•	**	CR**			*000065RECORD NUMBERG130130001*	**************************************	* " H	*0DFD	*D	**************************************	10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	**************************************	#•aS.S.s.a	*65 RECORD NUMBER 13 *	*	*000065	*	RECORD NUMBER *	*610	*DD	•
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	 *	-	_	*		_
40404040	40404040	00000000	D95C1000	SEC WORK 06000000	00000000	10CB1 F 1020411	C0871787	16C81110	04800000	PRINTER DTF E00040C3	109F5F3C	00000000	00000000	00000000 60606060	40404040	40404040	40404040	40404040	C5094040	40F0F1F3	40404040	40404040
F3404040	4040404	11101110	00000003	00800000	00000000	F3F0F0F0	80001623	FFFF119C	10080086	00000000	11001266	000000000	00000000		40404040	40404040	40404040	43404040	D5E4D4C.2	40404040	DISK PHYSICAL BUFFER FOFO FOFOF6F5 4040	40404040
D94040F1	4040400 40404040	00010002	43400000	00000000	00000000	FIF3F0F1	00820269	AOE38801	AB0100C4	00000000	60001200	00000000	00000000 00000000	000000000	F3404040	40404040	40404040	40404040	06090440	40404040	1154F0F0	40404040
E404C2C5	40404040	CONSTANT 40FFFF00	C08716FE	00000000	00000000	C2C5D9F0	300 17031000		C6000216	05000000	E2010412	00000000		00000000	D94040F1	40404040	40404040	40404040	40090503	40404040	11541380	40404040
09044005	40404040	40404040	E6E	00000000	00000000	4005E4D4	01001	34200060 10000200	007FC480 C6000216	06648000	100000000	00000000		E2U1127C 00126600 00000000	E404C2C5	40404040	40404040	40404040	4040404	40404040	00001884	
09C5C3D6	40404040	40404040	00000000	0002000	00000000	C30609C4	00801000	90000000	02000080	00159600	10004040	00000000	00000000	00126600	D9C5C3D6 D9C440D5	40404040	40404040	40404040	40404040	4040400 40404040	13960020	40404040
WORK AREA FOFOFOFO F6F54040 D9C5C3D6	4040400 4040400 40404040 404040040	404040 40404040	1040 00000000 00000000 00000000 0001	02000040 00000000	00000000	F6F 509C5	43115400 00003A00 00801000	1140 10000000 00111011 9C000000	13801380	320000C4	1154FFFF 1DC41143 10004040	11Cg 12 0000000 00000000 00000000	00000000 00000000 00000000 00000000	E201127C		\$0\$0\$ 0\$0\$0\$0\$ 0\$0\$0\$0\$ 0\$0\$0\$0\$	F0F0F0F0 F6F54040 40404040 40404040	04040404 04040404 0404040	0000000 0000000000000000000000000000000		00049801	0,00,000 0,00,000 0,00,000 0,00,000
1	40404040	40404040	000000000		00000000	FIELDS FOFOFOFO		10000010	1160 13004080 13801380	C40C9016	1154FFF	12 0000000 C	00000000	127CE201	F6F54040		FOFOFOFO	40404040	40404040	40404040	DISK IOB C49C40A1	40404040
PRIME 1000	1020	1080	1040	1 00 0	AREA 10E0	1100	1120	1140	1160	1180	1140	1100	1160	1260	1280	1240	1300	1320	1340	1360	1380	1340

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

* RECORD NUMBER	*	* RECORD NUMBER *	* 014000075 *		* RECORD NUMBER *	* 015000080 *	*	* RECORD NUMBER *	* 0160b*	***************************************	*0015000020000025000030.*	*000035000040000045000*	*050***000055***000060***000065***	*.00007000007500008000000*	*85000090000095000100*	*00010500011000011500012*	*00001250001300001350*	*00140 565	************************************	**************************************	*P Q. 2 # . D# . Q. # . I#18# . C# . D. C C #
13E0 40404040 404040D9 C5C3D&D9 C440D5E4 D4C2C5D9 40404040 40404040 40404040 1400 40404040 40404040 40404040 40404040	1420 404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040	1460 48404848 48484889 C5C3B6B9 C448B5E4 B4C2C5B9 48484848 48484848 48484848	1480 48484848 48484848 4848488 48484848 484848F0 F1F4F8F8 F8F8FF5 48484848	1480 404040 4040404 40404040 40404040 40404040 40404040 40404040 40404040	14E0 40404040 404040D9 C5C3D&D9 C440D5E4 D4C2C5D9 40404040 40404040 40404040	1500 4040400 40404040 40404040 40404040 404040F0 F1F5F0F0 F0F0F8F0 40404040	1520 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040	1560 4040404 404040D9 C5C3D6D9 C440D5E4 D4C2C5D9 40404040 4040404 40404040	1580 40404040 40404040 40404040 40404040 404040F0 F1F6C400 40A100C4 000015AC	15A0 002000000 188A1154 159&1300 FOFOFOFO FOF50000 00FOFOFO FOF1FOOO 0080FOFO	15CO FOFOFIF5 000400FO FOFOFOFZ F0000480 FOFOFOFO FZF50008 00F0F0FO F0F3F000	15E0 0880F0F0 F0F0F3F5 000C00F0 F0F0F64 F0000C80 F0F0F0F0 F4F50010 00F0F0F0	1600 FOFSF000 1080FOFO FOFOFSF5 001400FO FOFOFOF6 FOO01480 FOFOFOFO F6F50018	1620 00F0F0F0 F0F7F000 1880F0F0 F0F0F7F5 001C00F0 F0F0F0F8 F0001C80 F0F0F0F0	1640 F8F50020 00F0F0F0 F0F9F000 2080F0F0 F0F0F9F5 002400F0 F0F0F1F0 F0002480	1660 FOFOFOFI FOF50028 00FOFOFO FIFIFO00 2880FOFO FOFIFIF5 002C00FO FOFIF2	1680 F0002C80 F0F0F0F1 F2F50030 00F0F0F0 F1F3F000 3080F0F0 F0F1F3F5 003400F0	16A0 F0F0F1F4 F0003480 FFFFFFFF 16AE3408 16F53401 16C7E201 00750205 9C000F18	F2820DF2 810D	16E0 1084C087 00048F40 7A8000D2 0200C201 1000C087 178AC201 17FE3510 16AD7920	MAINLINE 1700 D77920D8 F290037B 20D67B20 D87B02C9 7BFIC27B 9FC37BFF C47A40C3 7820C3C0

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

A.	*2588	**BBBB* **0.2I	*	*2*		*2	*23*		*2*
1720 10174DOC 01173717 98750299 881000F2 10150C01 1789179F 1E011737 9789C100 1740 898001C0 10178388 8000F290 067AFFC4 7AE0C3C0 8717D8C0 87187EC0 8718AD3C	1760 321782F2 870D3C30 1782B801 18F29003 BC001CB8 0100E201 00C20210 B4C08700 FECORD 1D 1780 048F0074 083CC087 16F6B80 00F21004 C087179E BA010075 103C179D 16F6B501	17A0 12B51014 340817AB C20117CD 9C0A220A F28708CO 87175FCO 871766C2 011000CO 17CO 8717947D F000F201 08C08717 A48202C9 80001823 C0871787 7820C3F2 10153502 AND MOVE 17EO 1098BB01 002C0117 EC1A7A02 C978B0D9 F29004CO 8717577A 80D9C087 175BB408	1800 09840108 884027F2 90078C42 0EC0871A 0FC08718 39C08719 27884027 F2101685 1820 0136C087 1882C087 19578DFF 22C08119 02C08718 EACO871A 0C340818 AF850113 1840 C0871882 BD800F88 1027C015 1A98AE01 1917AC01 20DDAC01 2417AC01 22158E01	1860 2218B1AD 011922F2 823AAC01 2619AF01 2622AF01 2426AC01 1922C087 1A17C087 1880 1A98C087 18828C01 1918B1AF 011926F2 2016BD80 OFF28104 C0871A98 AC012426 18A0 C087185A C0871A17 AC010D20 C0871815 FFFF3408 18E9C087 000C7C00 0C780102	F29023BC 410EC087 1A0F3C01 18E0F287 043C0218 E0340B18 E97C0004 \$\$\$RIC 0103C087 0008C087 18263408 1951B501 36AE0130 388E0130 1956BC01	1900 012009AD 012030F2 02448C01 3019549E 013009AE 013038BE 01301956 C0871ADB 1920 C08718CA F2872734 081951B5 0136AC01 26309F01 26099C01 2506B801 02AD0226 1940 2EF29004 AD022641 F28203BA 4027C087 183500FF FF000334 0819CAB5 01309C02	290 058F0124 19D1C087 19708C01	1940 011926AE 011917AD 011915F2 020AAF01 1917BC00 22F2870F BUFFESEL 0023BC01 1950 1919CCAE 011926C0 87182AFF FF010004 00203408 19F68501 30360119 FA1C0119 1950 F8002E01 19F81BB5 01134C01 0619F8C0 8718CAC0 87000000 00FFFF9C 010009AE 185SRMO 1800 010D198E 010D1816 AE011917 BC400E85 0108B510 09000134 08189134 0218862C	1A20 011A9724 3C6C1A58 B9800FF2 10043C9C 1A587501 09AF0119 24860119 AE011924 1A40 B5020D36 021A950F 001A961A 92F28210 36011A93 36021A93 9CFF0000 C0871A47 1A60 3D001A97 F2811C3C 001A9636 011A9736 021A970C 031A821A 580E001A 801A979C

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

* B	************************************	************************	**2*	*.T*	**	**************************************	**************************************	*C * * * * * * * * * * * * * * * * * *	*************************************	*QBB#.BBBB	*KEYDESCRIPTIONPAGEB*	*** E B. C	*• = = 2 = 0# • • • = = 2 • • • = = = = 0 0 0 0 0 0 0 0 0 0 0 0 0	*2BB0#	# · · D · · · · · · · · · · · · · · · ·	**************************************	*0*	*8.98	#7S	*********************************	*********************************	#	********************************	*.'2R2H2*
1A80 7F0000C2 021154AE 010D2485 0113C087 18A80100 FFFF0080 34081ADA BD800FF2	1AA0 011E7502 136C0106 06850215 C0871ADB 88102788 1027F210 1EC08718 CAF2870E	188RBP 11750113 840113C0 8718018A 10278401 11750113 840113C0 87188234 0818059C	1AE0 0022074E 00061B07 786006F2 90054E01 061B09BD 0022F281 098F0022 1B06C087	Ξ.	112		1860 8C050511 058C0C14 11128C02 1811158A F0189B02 1690C087 18D1C087 16FE7820 PUT	1880 C3C0101E 71C0871B AD34081B AC7A20D9 5C8F8F90 C2021000 0C0411AF 18E1C087	18A0 1C82C087 18D17B20 D9C08700 007820C3 C0101E7E 7820D7C0 101B897B FFD77BFF	18C0 D87A02C2 7804C2F2 10037802 C2C08718 04C2021C AAC20111 4335101D A6400400	18E0 0002D2C5 E8C4C5E2 C3D9C9D7 E3C9D6D5 D7C1C7C5 C0870004 00C20110 005F18DA	1C00 DA6C00C5 377A01C2 7A40C375 01997502 05C08700 048B7502 05C08700 04827502	1C20 05880103 F2900D78 C0007910 00F29004 3C871C44 4D010310 99F28107 750103C0	1C40 B71C1EF2 B70AC202 1084C087 00048F11 C2011000 3C401112 0C111111 11123CF0	1C60 111C0C08 111B111C 5C8F8F90 C2021000 0C0411AF 1BE1C087 1C82C087 1BD1C087	SUBSEGMENT	1CAO F0229803 1F90C087 1C7AB408 09840108 BC400EE2 0110BAO4 14C0B71D 0FE20111	1CCO CO87100F 88400FCO 90109FF1 E200CO87 1D782C00 1CF922OC 021CF81C F93C001C	ICEO F785011F 1E011CF8 0171E601 85011870 E6018501 0DIC5F12 085FE201 128A0614	1000 C0871D0F 8C01211C CAE20113 F2870984 08217000 00F2815A 9CC01500 BB58162C	1020 001D2914 3A041D29 F1E600C0 871D78B0 E01BB501 18AC0024 1AB80814 F29004AC	1040 00241B2C 011D4A15 F3E2016C 010515C1 E01D785C 010305BB 0214AE00 2415B804	1D60 7502059C 01000E2C 001D9F24 2C001DAB 24C0FF00 0088010F F210060C 8F108F10	1080 90700000A F2814838 201009F2 10411C01 10C80A1C 011DCC0A B9FF12F2 10068D00

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

3	********************************	***************************************	? 22	**************************************	RECORDS WERE READ FROMTHE*	INDEXED FILECRR2*		* · · · · · · · · · · · · · · · · · · ·		***************************************	\$ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	46	46	***************************************	000000000000000000000000000000000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	••••••••A•••••••••••••••••	· · · · · · · · · · · · · · · · · · ·	S S B	\$ · · · · · · · · · · · · · · · · · · ·	**************************************	
	*		*2	*-2	:	*		*	*	* B	*	*	*	*	•		*	*	*	. <u>.</u>	*	*2*	.
001DFA24	411C011E	F20409BD	10078040	1000001	D4E3C8C5	1E8F3501	0411AF1E	8 70 00 000	00000000	291DF282	OD1F4DAE	02040404	040404040	04260604	80808016	040C1515	00000000	067AFFC4	18F29003	00F21004	9C0A220A	A48202C9	00788009
10EE242C	1009F210	06801210	10063A20	020EC201	40060906	10040087	02 100 00C	871BD1C0	00000000	1F31AD01	0D198E01	04040204	04080707	04000913	80808080	90808090	00000000	8000F290	17828801	16ACB880	C2010000	08C08717	EC1A7A00
000E2C00	81483820	FF10F210	1E1A3A20	048F4075	D9C5C1C4	7A08D9F2	8F8F90C2	281E60C0	00000000	FFFF3408	0009AE01	09010409	04180404	0204070F	80808080	04141604	021025FF	10178388	87003630	083CC087	34081748	F000F201	00200117
02059001	7D000AF2	F2041589	1E163C3B	B4C 08 700	C509C540	CLOSE MAINLINE	PROCESSING 081EBE5C	1E5C8C10	0000000	01006262	10060000	04010409	00040704	05050011	80808080	10140D0A	04040204	89800100	321782F2	048F0074	12851014	87179470	10988601
O210C375	108F1090	06801212	81083C38	0AC 202 10	C4E240E6	C5487A20	1R 00048434	908C1519	00000000	00001800	2826C087	0001210A	04FE2409	0AFE0704	80FE8080	11FE0804	04FE0704	9789C100	871BAD3C	84008700	16F68501	01100000	10153502
CONTROL R	10060C8F	FF12F210	BD480EF2	440EF211	62630609	40C6C9D3	04830087		00000000			08851009 00012	07070404	OF041310	02208080	18040405	02070408	1E011737	87187EC0	00020210	BA010075 103C179D 16F6B	871766C2	7820C3F2
OUTPUT CONTROL ROUTINE	88010FF2 10060C8F 108F1	170A1C01 1E180A89 FF12F210 068D1	19F2870E	89010EBD 440EF211 0AC20	CONSTANTS 187A4000 03010009 C5C30609 C4E24	40C905C4 C5E7C5C4 40C6C9D3 C54B	05008700 04830087	11188AF0 02980200	00000000	00C20200 00AE010D 24850113	F21407BC 700EAC03	011917BC 400EB501	09020402 04260204 07070404 04FE2	04020404 04070409 0F041310 0AFE0	05080302 02208080	04070404 04070A15 1B040405 11FE0	05041504 02070408	17890000	7AE0C3C0 871708C0 87187EC0	0100E201 00C20210 84C08		87175FC0	C0871787 7820C3F2 10153
01088510	COFF1CAA		480EF201 19F2870E BD480EF2	1E20 0EF 20209		4006004	10997502	9C 0202		00020200	0089FF28	011917BC		04020404	47050505	04070404	15040404	2000 10150C01 17890000 1E011737	7 A E OC 3 C O	80001088	C087179E	F28708C0 87175FC0 871766C2 01100	80001823
1040	0001	1060	1E00	1E20	1E40	1660	1E80	1E A O	SAMPLE 2 1600 000	1 F 00	1F20	1540	1F60	1F80	1FA0	IFCO	1FE0	2000	2020	2040	2060	2080	20A0

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 PRO	GRAM -	*	SAMPLI
	0105 F#		*	SAMPL 1
	_	DS 100 RECORDS TO AN I		SAMPL 1
	0107 F#	ac aur accona caou E.I.	*	SAMPLI
		DS ONE RECORD FROM FILE		SAMPL 1
		UT. THE FILE \$SOURCE IS PLE PROGRAM SAMPL2 IS (SAMPL1 Sampl1
		ING A RETAIN-T PARAMET		SAMPL1
		E \$SOURCE.	*	SAMPL1
	01094F*		*	SAMPL 1
	0110 F# 3. CRE	ATES THE DUTPUT DATA U	SING A *	SAMPLI
	0111 F* LOO	P IN THE CALCULATION ST		SAMPL1
	0112 F*		*	SAMPLI
		S KEYS FROM 000005 THRE		SAMPLI
		INCREMENTS OF 5.	*	SAMPL1
	0115 F#	W D 35 5311 DJED 84 54M	* PLF PRIGRAM 2 *	SAMPL1 Sampl1
		ULD BE FOLLOWED BY SAM! VERIFY THAT THE FILE W	EE THOUSENESS	SAMPL1
		DED.	*	SAMPL1
	0119 F*	323.	*	SAMPL1
		*******	********	SAMPL 1
2001	0121 FSSOURCE IP	F 256 256	DISK	SAMPL1
0002	0122 FDISKOUT O	F 256 128 06AI 1	DISK 31	SAMPL 1
0003	0123 FPRINTER O	F 96 96	PRINTER	SAMPLI
0004 0005	0201 I\$SDURCE NS 0202 I	01	1 1 NODATA	SAMPL1 SAMPL1
		7 4000	50007 (3	CAMOL 1
0006 0007	0301 CL0 0302 CL0	Z-ADD0 Z-ADD0	COUNT 50 RECNBR 30	SAMPL1 Sampl1
0000	0302 CL0	REPEAT TAG	NECTOR 55	SAMPLI
0009	0304 CL0	COUNT ADD 5	COUNT	SAMPL 1
0010	0305 CL0	RECNBR ADD 1	RECHBR	SAMPL1
0011	0306 CL0	COUNT COMP 505	02	SAMPL1
0012	0307 CLO NO2	EXCPT		SAMPL 1
0013	0308 CLO NO2	GOTO REPEAT		SAMPL 1 Sampl 1
0014	03081 CL0	SETON	LR RECNBR	SAMPLI SAMPLI
0015 0016	0309 CLR 0401 OPRINTER T	RECNBR SUB 1 204 LR	RECOOR	SAMPLI
0017	0402 D	204 CN	20 *SAMPLE PROGRAM 1 HAS*	SAMPLI
0018	0403 0		27 'LOADED'	SAMPLI
0019	0404 0	RECNBRZ	31	SAMPL 1
0020	0405 O		39 "RECORDS"	SAMPL1
3021	0406 D	•	61 'INTO AN INDEXED FILE."	SAMPL1 SAMPL1
0022	0408 D T 0409 D	2 LR	21 "KEYS ARE IN ASCENDING"	SAMPL I
0023 0024	0410 B		42 SEQUENCE STARTING AT	SAMPL1
0025	0411 0		64 1000005 AND INCREASING!	SAMPL1
0026	0412 0		84 "IN INCREMENTS OF 5."	SAMPL 1
0027	0413 D T	LR		SAMPL1
0028	0414 B		21 'SAMPLE PROGRAM 2 WILL'	SAMPL1
3029	0415 0		44 PRINT FROM THE INDEXED	SAMPL1
0030	0416 0		65 "FILE TO SHOW THAT IT" 86 "WAS PROPERLY LOADED."	SAMPL1 Sampli
0031	0417 0 0501 ODISKOUT E	NO 2	00 .MA2 SKALEKTI FRANCA.	SAMPL I
0032 0033	0501 0D13K001 E	COUNT	5	SAMPL 1
0034	0503 0	00011	94 "RECORD NUMBER"	SAMPL 1
		OFCHEO	128	SAMPLI
0035	0504 O	RECNBR	128	SAMPLI

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
RG 314 UNREFERENCED FIELD NAMES
 STMT# NAME
A TAGGO CCCC
  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
RG 305 W
RG 314 W
                 DUTPUT INDICATORS IN COL 23-31 MISSING OR ALL NEGATIVE.
                 INDICATOR ASSIGNED BUT NOT USED TO CONDITION DPERATIONS.
                 FIELD. TABLE OR ARRAY NAME DEFINED BUT NEVER USED.
```

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

START ADDRESS	CATEGORY	NAME AND	CODE LI		
4000		GLOSAL Common	0768	1992 9	
47C8 4800	0	S AMPL1	0009 0100	256	
4900	0	\$#RT02	0123	291	
4AZ3	ō	\$#IPCR	004F	79	
4A24	_	\$80AC9			
4A72	0	\$#OPCR	0090	156	
4A73		\$00B18			
480E	0	\$#CONO	00E9	233	
48F7	0	S#CONL	0005	. 5	
4 B F C 4 C O A	0 2	\$#CON2 \$\$CSIP	000E 0027	14 39	
4031	2	\$\$[DUT	005F	95	
4090	2	\$ \$ SRBR	0082	130	
4012	2	\$\$SRUA	0026	38	
4D38	2	\$\$SRTC	0010	28 .	
4038		DMSRLO			
4D49		DMSRTC			
4D4C	_	DMSRER	01.05	2	
4054	2	\$\$ SRB1	0105	261	
4E59 4E75	2 2	\$\$SRDF \$\$SRMO	001C 00A4	29 164	
4F19	2	\$ \$ \$ \$ \$ \$ \$ B	0044	70	
4F5F	2	SSRDI	003E	62	
4F84	-	DMSRPD	4002		
4F7D		DMSRRO			
4F9D	2	\$\$SRBP	002F	47	
4F33	6	\$\$LPRT	0001	209	
5090	93	\$#0PEN	0060	107	
5104	126	\$#INPT	0090	144	
515C 5163		\$@0025 \$@0020			
5190		\$20059			
5154		\$90010			
5158		\$00021			
51 94	28	\$#IH01	0008	8	
51A2	126	S#TCAL	0052	82	
51F4	29	\$ #EXPT	3020	45	
5221	28	\$#0H02	2000	12	
522D 5246	126	\$#IFLD \$adces	0010	29	
524A	93	\$#CLOS	0028	43	
5251	, ,	\$ a D E C 2	20,	• •	
5262		\$30E03			
5266		\$40ED7			
5275	107	\$#LROT	0076	118	
5253	126	\$ # TOUT	8000	11	
52F6 5314	126 71	\$#LRDF \$#DAOF	0024 0010	36 29	
5337	28	\$#0H03	0010 000C	12	
5343	126	\$#RCID	0046	70	
537A		\$30C9A	00.0		
5389	126	S#CFLD	0026	38	
53AF	11	\$\$PGR1	0043	67	
START ADDRESS	CATEGORY	NAME AND ENTRY	CODE LEM		
53F2	107	\$#LRC	0014	20	
OL100 I OL101 I OL104 I	THE START	CONTROL ADD ER OF LIBRA	Y SAMPLI IS RESS OF THIS RY SECTORS RE EM,UNIT-RI,RE	MODULE IS	4800. 13

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

	# (B	***************************************	**************************************	*B+2*	*		**************************************	******************************	*48	* . K	*>	***	***************************************	**** B**********************	*/e*	*.D* </th <th>¥ * * * * * * * * * * * * * * * * * * *</th> <th>***************************************</th> <th>********************************</th> <th>*222</th> <th>本。。 * * * * * * * * * * * * * * * * * *</th> <th>*</th> <th>*2*</th> <th>*************************</th> <th>**************************************</th> <th>***************************************</th> <th>格</th>	¥ * * * * * * * * * * * * * * * * * * *	***************************************	********************************	*222	本。。 * * * * * * * * * * * * * * * * * *	*	*2*	*************************	**************************************	***************************************	格
3PTI34-PL1 ID-\$ IAR-5021 PMR-0072 PSR-0004 XR2-4402 XR1-4000 ARR-4FDF	GLOBAL (BUFFERS AND IOBS) 80124500 00000050 300E407C 08010000 129D0C14 AE420040 19000040 00000000	E0000040 7C000300 30300004 00000000 40008820 93C39043 DB381353 FFF29014	0C014555 46EE8880 82F29008 3C484550 3C874060 C20245A4 F4100088 3C444550	C20245A4 F4100082 350246EA C08043AB B8408EF2 90960CO5 47015457 40404340	0+0+0+0+0 +0+0+0+0+0 +0+0+0+0+0+0+0+0+0	DUPLICATE LINES	459F0C01 484C484E 30874092 32024704 8C0100F4 10008633 304111BD 5033C091	410C8060 00C38141 3C388048 5CF28745 880200F2 100E8801 00F21040 C2024750	F4100085 \$2024735 380846FA F2101EC2 32473EC2 0154C930 4846FAF2 8203D201	12D20109 797006F2 90038A04 06F41000 85CO8740 92B53134 C232454E 3802485B	F290128C 100FC087 4260BD41 0EC08145 463B0248 5B2C0141 A30D0E01 41A3484C	1C5F0000 5FJE0148 4C48442D 01484C15 C004410C 34024858 8C400FC0 874250BD	700EC081 453EBD44 0EC08145 3E8E014F 459F0C31 4845484E C087410C 350246EA	83208FC2 02454E38 02485BF2 930EBC10 0FC08742 638D410E C0814546 3883485C	40004000 0000000A2 0000421E 00000000 00000000 00000000 0004200 49487061	307C5C01 0001484C 4541F284 088C0115 4541F287 0E050148 4C45A3F2 84058C31	1545433A	84080984 01088501 11C08743 549C013D 388C430E B9C33FF2 106EAC31 2E4FC087	42EE6G02 1634F204 168C002D AC002E14 8F002E43 746C0017 2EF20209 8C01148C	440EF287 43AE012E 4FC08742 EE=28418 B8400FF2 9006BC70 0EF2872C BC420E8F	00144374 C087427A B8800F7B 0707F290 067A0107 F287037A 02077200 08F41000	0288100F C0104354 B5010BB5 10093408 435388C0 00F21018 AC01341C 986034F2	10088F01 2E4374F2 82388E01 344376C0 8742FCAC 01331CBC 0134BD14 34F28411	8F012E43 74F28224 40904000 00C804A2 00004364 00200300 12904000 00C80430	00004346 4996C09D \$00003C8 03AZB3D3 44640020 0300129D 400000C8 00000000	43284996 FOFUFJF4 F9F54040 40404040 40404040 40404040 40404340 43404340	0+0+0+0+ 0+C+0+0+ 0+0+0+0+ 0+0+0+0 +0+0+0+0
gC	000%	4020	4040	4353	4080		4100	4120	4140	4160	4180	41A3	4120	41E0	4200	4220	4240	4260	4280	4240	4200	42E0	4300	4320	4340	4360	4380

Figure 4-17 (Part 1 of 6). Storage Dump of Sample Program SAMPL1 (Model 15)

343	04040404	00000000 00000000	40404040	404	04040 40404040 40090503		0609C440 D3E4D4C2	03640402	**************************************
300	55394040	25394040 40404040 40404040	40404040	404	04040 40404040	40404040	40404040	40404040	#ER.
3E0	40F0F9F9	FOFOFOF5	F0F04040	40404040	40404040	40404040	40404040	40404040	**003000500**
00 %	40404040	40404040	05050505	40404040		40404040	40404040	40404040	
0.74	04040404	4848488 48484848 4848484848484848	0+0+0+0+	04040404	0 + 0 + 0 + 0 +	40096563	04406000	U2 E4 U4 C C	**************************************
0 4 4 0	05036060	40404040	04040404	04040404	40404040	40404040	0 5 0 5 0 5 0 5	0404040	* ER
1460	40F1F9F0	F0F3F3F4	F8F54040	40404040	69696969	67670707 07070707	40404343	43494340	*.130300485*
4480	40404040	05050505 05050505 05050505	05050505	43404343	40404040	43434343	43434349	49494049	# · · · · · · · · · · · · · · · · · · ·
044	40404040	40404040 40404040 40404D4O	40404040	69690969	04040404	43090553	06096443	95 E\$ D4 C2	**************************************
0094	05094040	40404040	00000000	4040404	40404040	43404343	43404043	60606060	# E D
64E0	40F0F9F7	F3F3F3F4	F9F04040	40404040	40404040	¢0404040	4040404	40404040	*.097000493*
15 00	40404040	04040404	¢0¢0¢0¢0 ¢0¢0¢0¢0	43434343	404040	40404040	40404040	6904040	*
1520	40404040	¢0¢0¢0¢0 ¢0¢0¢0¢0 ¢0¢0¢0¢0	69690404	63636363	05050505	43090563	0903643	05640402	*NUM8*
1540	25394043	09090909	40404040 40404040	43404040	60404040	43434343	43434343	40404040	# · · · · · · · · · · · · · · · · · · ·
1560	40F0F9F8	40904300	30058442	00000000000000000000000000000000000000	50200035	12904000	00088000	00004582	*.058F
4580	06049664	400000008	38A2 3333	45 A00323	0000000	43000006	88000000	4564496	*******************************
5 A O	F0F0F0F4	F2F500C8	0000000	F 4F 3F 000	C 5 3 0 F 0 F 0	F 3 F 4 F 3 E 5	000CC00F0	F0=0F4F4	*003425.H.003430.H.00043500044*
1500	FOOOCCBO	F000CC80 F0F3F0F4 F4F500D0 00F	F4F500D0	0F 0F 0	F4F5F000	00905050	F0F4F5F5	000400F0	*0000445000450000455.M.0*
45E0	F 0F 0F 4F6	F0000480	F00005480 F0F0F64	F6F500D8	0000000	F4F7F330	0880F0F0	F0F4F7F5	*00460.M.000465.Q.000470.Q.000475*
0094	000000000	F3F3F4FB	F3000C80	FOFOF0F4	F8F50100	30F0F0F0	F 4F9=001	0 080F 0F 0	*00048000048500049000*
1620	FOF4F9F5	01040160	FOFOF5FO	F3310480	F 0F 0F 0F 2	F2F50058	00F 0F 0F 0	F2F3F000	*0495000500003225000230.*
0494	5880F0F3	5880F0F3 F3F2F3F5	005C00F0	F0F0F2F4	F0005C80	FOFOFOF2	F4F50080	0000000	*000235.*.000240.*.000245000*
0994	F2F5F000	8080F0F0	8080F0F0 F0F2F5F5 008	008400F0	F0F0F2F6	F3308483	FOFOF0F2	F6F50088	*250000255000260000265*
0894	00F0F0F0	F2F7=300	8880F0F0	F0F2F7F5	008C00F3	FJFJF2F3	F3008083	00081500	*.000270000275000280*
5 A 0	FOF3F0F2	F8F53090	00F0F0F0	F2F9F000	9080F3F0	F0F2F9=5	009400F0	F0F0F3F0	*30028530029300029500030*
0094	F3009480	F0F0F0F3	F 0F 5 0 3 9 8	03505050	F3F1F000	9880FJF0	F0F3F1F5	00900000	*00303350003100003150*
•6E0	F0F0F3F2		F0009C80 F0F0F0F3	F2F	500A0 00F0F0	F3F3F333	AOBOFOFO	F3F3F3F5	*00320000325000330000335*

Figure 4-17 (Part 2 of 6). Storage Dump of Sample Program SAMPL1 (Model 15)

***** 000340*** 000345*** 000350*** 00*	*0355000360000365000370.*	***000375***000380***300385***	*390,000395000400000405*	*.300410000415.D.000420.D*	************	***************************************	*	*SAMPLE.PROSRAM.1.HAS.LOADED.100.*	*RECORDS.!NTO.AN.INDEXED.FILE*	*		***************************************	**************************************	*	**************************************	*	*	*	*H\$SOURCE	**************************************	*	*	**************************************	**************************************	***************************************	**<.822284*
0 00A400F0 F0F0F3F4 F000A480 F3F0F3F3 F4F53JA8 33F0F3F3 F3F5F330 A880F0F0) FOF3F5F5 00ACOOF3 F3F3F6 F330AC80 F0F0F3F5 F6F50380 00F0F0F3 F3=7F000) 8093F3F7 F3F7F5 008430F0 F0F0F3F8 F0008480 F0F0F3F3 F8F50098 03F0F0F3) F3F9F000 B880F3F0 F0F3F9F5 008C00F0 F0F0F4F0 F000BC80 F0F3F0F4 F0F500C0) 00F0F0F0 F4F1=000 \$080F0F0 F0F4F1F5 00C400F3 F3F0F4F2 F333C483 48484848	\$65 83638483 83838383 8383838383838383838383838	COMMMON (FIELDS) 48484848 48484849 FOFOFOF5 FOF5F1FO FO484848 48484848 4848484 43434848]	PRIME WORK AREA) E2C10407 D3C540D7 09D6C709 C10440F1 40C8C1E2 40D3D6C1 C4C5C440 F1F0F040) 09C5C3D6 D9C4E240 C9D5E3D6 4OC1D540 C9D5C4C5 E7C5C440 C6C9D3C5 484O4O40	0+C+0+C+C+C+0+0+ 0+0+0+0+0 +0+0+0+0 +0+0+0+0	DUPLICATE LINES	CONSTANT AREA (CONSTANT AREA (CONSTANT AREA (CONSTANT AREA (CONSTANT AREA (CONSTANT AREA	3 00000000 00000000 00000000 C087524A C087510A 43400000 0000000 D9554800	ļ	SECONDARY WORK AREA)00003000 000000000 000000000 00000000	83004925 49483000 333A3031 00480001 30413053 744830	1 00000048 000001004 49374996 0000003A 0000048 00020010 0049004A 0200000	3A010084 480 <u>00300</u> \$0004831 FFFF4995 4A454900 41004280 42004200 01000100	FFF70C8 8300C833 3158E2D& E4D9C3C5 40000000 0008	000000000 0300C884 300000000 00000000 003033C8 830034333 84114948 44024491) 49264800 40404328 43460100 0080FFFF 00C70030 C88302C4 C9E2D2D6 E4E34040) 000000048 0000FF00 00008001 04000001 00010000 C8080000 00000000 00000000	9200		INPUT CONTROL ROUTINE) 12COFF4C 0A6CO112 0DBD410E F28218F2 81098D44 0EF2823C F2810CC2 0248B4F4
4700	4723	4740	4763	4780	47A0	677.0	47E0	4830	4820	4840		4880	48A0	4800	48E0	4900	4920	4940	4960	4980	4940	4900	49E0	4400	4420	6445

Figure 4-17 (Part 3 of 6). Storage Dump of Sample Program SAMPL1 (Model 15)

Figure 4-17 (Part 4 of 6). Storage Dump of Sample Program SAMPL1 (Model 15)

*2	**************************************	*./2	***************************************	*.22	*	*.	*1.2	***! ***! *** +7+,,,+5 ,%**.B*	***************************************	*2	***>****	*.!24B2	*.2	*G+ I,2+ 2 H*	*S*3**********************************	*· ·3···3····*	*52	*	*** - 10 - 10 - 10 - 10 - 10 - 10 - 10 -	***************************************	***************************************	*.B	*4	*2802B4B0*	**	********************************	**************************************
F2900384 0153AC00 3738AF01 361C6C02 03378CFF 1UAE0333 40AF3033 5AF2A308	8EJJ304E 56F2203C 8501696C 00176A8E 006A4E58 3C034JE2 3136CJ87 4F9DADJJ	6451F282 1F8C006A 850169C0 874F848B 01388804 02F2930C 85016984 01587501	18840169 C0874C88 AC034138 BC700EC0 87404CFF FF000300 0134084E 74A)013F	\$\$\$RMO 1FF2820D 89FF41F2 14078C70 0EAC0341 38C0874C 7034084F 1234024E F92C014F	18343C6C 4EC)8980 OFF21004 3C9C4ECD 75010888 40398043 OFF2963E AF317734	860177AE 017734F2 8708AF31 19348601 19AE0119 34850200 36024F16 0F334F17	4F13F282 1036914F 1436024F 146CFF00 00C0874E 3333004F 18F2811C 3C004F17	36014F18 36024F18 0C034EF7 4ED00E00 4EF54F18 6C7F3333 C2324996 AE013034	85011388 40398020 3FF29603 853111C0 874CD801 33FFFF30 8034084F 5E8D830F	F2011E75 02136:01 16168502 10C0874F 90880139 880139F2 1021C387 4F7DF287	11880139 C0104F23 30874F84 84013988 02388401 11750118 840113C0 874CDH34	278 4303F210 04F41030 037C000F 780102F2 90238541 0EC0374D 45	93F28704 3C024F93 34084F9C 7C000878 07377432 37F41000 02C0874C F334084F	C79C0030 174E0016 4FC97860 16F29005 4E01164F C8B00030 F281098F 00304FC8	C0874FA5 C0874D30 01040020 84080984 31388C40 3EE23110 8A0414C0 875306E2	01115087 50058840 0FC09050 97E20112 BA06148A 0102C087 50060C01 50934FEC	E20113F2 870A3408 50937D00 00F28L7D 9C001500 85011778 1000F210 04F41333	03750118 8401177D 4002F201 67880102 F290222C 00504E18 0002504D 504E3000	504C1E01 504D0885 310D1C30 00000085 01178831 029C031E 166C0108 15F41000	02750118 9E001E08 780407F2 90049C00 1E088832 14AE031E 15880414 F29904AC	001E15AD 001E1CF2 04038348 0EBB0414 CO874FDF 83410EB5 31088513 09350230	2EC20148 005F18DA DA6C00C5 827A01C2 7A40C375 01997502 05F41000 8A750205	F4100082 1C0050CA 10380048 C5F2900D 78C00079 1000F290 043C8750 E0450103	4899F281 07750103 C08750C4 F28709C2 024884F4 10008F11 C2014800 35F047D3	0C0747CF 4700C087 51047802 C978F1C2 789FC378 FFC47A43 C37823C3 C010514A	OC015134 51977502 99881000 F210150C 01518551 981E0151 349789C1 00898301	CO10517F 888000F2 900674FF C474EOC3 C0875389 C08751A2 C08752F6 3C32517E
40E0	4E00	4E20	4E 40	4E60	4	4 E A O	4 EC 0	4EE0	4F 00	4F20	4F40	4F60	4F80	4FA0	4F C 0	4FE0	9300	5020	5040	5360	5380	5040	9000	5060	9100	5120	5140

Figure 4-17 (Part 5 of 6). Storage Dump of Sample Program SAMPL1 (Model 15)

*************************************	2B.<	7752.		***************************************	I 2 1 <	.<.B*	*******************************	**************************************	***************************************	· · · · · · · · · · · · · · · · · · ·	***	*d66	***P#*Q: 8 * 82 * # B * * * * * * * * * * * * * * * * *	**************************************	*************************************	***** 2****** B********	*******************************			**************************************	*10-**	3.WA.MODULE.IN.A.GROUP.HAS.C*	*ATEGJRY.VALUE.0-7*	+0-************************************	*LO16.WENTRY.POINT.NAME.ON.OPTI*
5	*	*	*	*	*	* S B	*	*	*	*	*	***	*	*	*	*	*	*R2	*.2	*	*	*03	# A T	*	*L0
F287303C 30517EB8 0118F290 038C001C 880109E2 0100C202 48B4F410 008F0074	75343 BA310075 103C5199 5194C231	470048F7 7843C3F2 932F0650	48F344A5 1047C047 D2104BFB C08753AF 8104C933 7904C9F2 EXCEPTION OUTPUT	51F4C087 51847A20 C37820C3 C0105251 C08752E8 34085220 5C8F8F90 C2024800	0547CD8C	EZC2024C 31C20149 2635104A 73750298 890218F2 100C2C01 52411E85 0212C087	0000C087 510AC087 52427A20 C33C8052 5F7808D9 7A08D9F2 100CC087 53F2C087	52EBC087 52753501 48997502 05F41000 83F41000 84340852 EA5C8F8F 90C20248	330C044A 1548L28C 13134B30 8C051A4B 368C021E 47D09AF0 1E99021C 908C0626	483D8C14 3C4852C0 8753370C 044A154B 178C1414 48678C13 2949738C 143F4890	80125348 43008753 37000444 15481080 14144888 80152848 CE801340 48E28013	5548F6C0 875337C0 87526678 20C3C010 5262C087 52F57823 C3C31352 66780107	FD87402 C27804C2 F2100378 02C2C0	UNCHELOW HOUTINE D90C044A 155336C0 87533778 2009C087 00000006 0000000C2 024FCCC2 01493735	RECORDID 10447385 01128510 14340853 50C20100 009C0A22 04F28714 880118F2 90038307	1008751 50880118 F290038B 0710087 51630201 49000087 51900087 53498202	C9803052 46C08753 657820C3 F2101555 024898BB 01032C31 53931A7A 303978B3	09F29004 C3875154 748009C0 87515874 08FF7402 FD7502FF 7404F3F2 1030R9FF	00F21022 7504F82C 3053CD00 F281043C 7853082C 01533402 7404CP3C 7453D8E2	7502F075 10FF	4BFCC087 525 <u>6</u> FF40 404040 40404040 40404040 40404343 4343434 434310603	F0F3F340 E64040C1 40D4D6C4 E4D3C540 C9D543C1 43C7D9D6 E4D74OC8 C1E24OC3	C153C5C7 0609E840 E5C1D3E4 C54OF050 F7404040 43404343 40434343 43404040	\$009C+0+ 0+0+0+0+ 0+0+0+0+0+0+0+0+0+0+0+0+0+	03F0F1F6 40E64040 C505E3D9 E840D706 C905E340 D5C1D4C5 40D6D540 D6D7E3C9
5160	5180	5140	5100	51 E0	5233	5220	5240	5260	5280	52A0	5200	52E0	5300	5320	5340	5360	5380	53A0	5300	53E0	2400	5420	2440	5460	5480

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
                                                                                 SAMPL 2
      0104 F* THIS PROGRAM -
      0105 F#
                                                                                 SAMPL2
      0106 F*
                1. MUST BE PRECEDED BY SAMPLE PROGRAM I
                                                                                 SAMPL 2
      0107 F*
                   WHICH LOADS AN INDEXED FILE.
                                                                                 SAMPL 2
      0108 F*
                                                                                 SAMPL2
      0109 F*
                2. READS AN INDEXED FILE SEQUENTIALLY.
                                                                                 SAMPL 2
      0110 F*
                                                                                 SAMPL2
      0111 F*
                3. USES A BLOCK LENGTH FOR DISK WHICH
                                                                                 SAMPL 2
      0112 F*
                   IS DIFFERENT FROM THAT USED FOR
                                                                                 SAMPL2
      0113 F*
                   LOADING THE FILE IN SAMPLE PROGRAM 1.
                                                                                 SAMPL2
      0114 F*
                                                                                 SAMPL 2
      3115 F*
                4. COUNTS THE NUMBER OF RECORDS READ SO
                                                                                 SAMPL 2
      3116 F*
                   THAT THE USER CAN QUICKLY VERIFY THAT
                                                                                 SAMPL2
      0117 F#
                   100 RECORDS WERE LOADED.
                                                                                 SAMPL 2
      0118 F*
                                                                                 SAMPL2
      OI19 F****************************
                                                                                 SAMPL 2
     0120 FDISKIN 1PE F 512 128 06AI 1 DISK
1000
                                                                          91
                                                                                 SAMPL 2
0002 0121 FPRINTER 0 F 96 95 OF
                                             PRINTER
                                                                                 SAMPL 2
0003 0201 IDISKIN NS 01
                                                                                 SAMPL 2
                             1 C0
0004
      0202 [
                                                        6 KEY
                                                                                 SAMPL2
                                                    1
                                                    82 94 DESC
0005
      0203 1
                                                                                 SAMPL2
                                                   126 1280RECNBR
                                                                                 SAMPL2
0005
     0204 1
0007 0301 C
               10
                       COUNT
                                 ADD 1
                                                COUNT
                                                         33
                                                                                 SAMPL 2
0.008
     0401 OPRINTER H 204
                             1 P
                                                                                 SAMPL2
                             αF
                                                                                 SAMPL 2
0009
     0402 0
                   ЭR
0010
     0403 0
                                                5 "KEY"
                                                                                 SAMPL 2
                                                22 'DESCRIPTION'
                                                                                 SAMPL2
0011
      0404 0
                                                30 *PAGE*
                                                                                 SAMPLZ
2012
      3405 O
                                                                                 SAMPL 2
0013
     0406 0
                                     PAGE Z
                                                35
                                                                                 SAMPL 2
      0407 0
                             01
0014
                    D = 1
0015
     0408 0
                                     KEY
                                                                                 SAMPL2
                                                                                 SAMPL 2
0015
     0409 0
                                     DESC
                                                21
                                     RECNBRZ
                                                                                 SAMPLZ
0017
     0410 0
                                                25
8100
     0411 0
                    T 3
                                                                                 SAMPL2
                          01 LR
                                     COUNT Z
                                                                                 SAMPL2
0019
     3412 O
                                                26 *RECORDS WERE READ FROM*
                                                                                 SAMPL2
0023
     0413 0
                                                44 THE INDEXED FILE.
120¢
     0414 0
                                                                                 SAMPL2
```

```
INDICATORS USED
LR OF 1P 01
```

```
FIELD NAMES USED
STMT# NAME DEC LGTH
                        OISP
0013 PAGE
              0
                  004
                        0010
0004 KEY
                  006
                        3005
0005 DESC
                  013
                        3012
0006 RECNSR 0
                  003
                        0015
0007 COUNT
             0
                  003
                        3018
```

Figure 4-18 (Part 1 of 2). Source Listing of Sample Program SAMPL2 (Model 15)

START	CATEGORY	NAME AND	CODE LE	исти
ADDRESS	CATEGORY	ENTRY	HEXADECIMAL	
ADDRESS		CHIKI	HEXADECTHAL	DECIMAL
4000		GLOBAL	04D2	1234
4402		COMMON	0010	29
4500	0	SAMPLZ	0100	256
4600	0	\$#RT02	0004	196
4604	0	\$#IPCR	004F	79
46C5		\$@06D3		
4713	0	S#BPCR	0090	156
4714	_	\$20722		_
47AF	0	\$#CONO	0005	5
4784	0	\$#CONL	002C	44
47E0 47E1	0	\$#CON2 \$#CON3	0001 0017	1 23
47E1 47F8	2	\$#CU\3	0017 004F	79
4847	2	\$\$SRMD	0044	164
4 BEB	2	\$\$SRDI	003E	62
4910		DMSRPD	0032	O.L
4909		DMSRRD		
4929	2	\$\$SRIC	0003	211
4981		DMSRIF		
49F5	2	SSSRRC	0088	187
4AB7	2	SSSRRI	0032	50
4AE9	2	\$\$ SR TC	001C	28
4AE9		DMSRLO		
4AFA		DMSRTC		
4AFD		DMSRER		_
4B05	2	\$\$SRBP	002F	47
4834	6	SSLPRT	0001	209
4005	93	\$#DPEN	080	141
4092 4063	126	\$#INPT \$@0832	009F	159
4CFA		\$@0832 \$@0839		
4027		\$30866		
4CEB		\$208ZA		
4CEF		\$ 2 08 2E		
4031	28	S#IHOL	0008	8
4039	126	S#IFLD	002C	44
4052		\$008FB		
4D65	126	\$#DCAL	0010	16
4D 75	93	\$#CLOS	001F	31
4078		\$@09F7		
4D85		\$ 80 A 0 4		_
4094	107	\$#LROT	0030	48
4DC4	126	S#DOUT	0032	50
4DF6	126	S#TOUT	000B	11
4E01	126	\$#LROF	0024	36
4E25	71 20	\$#0F0F \$#0H02	0024	36
4E49 4E55	28	\$#UHUZ \$#MF01	000C 0028	12 40
4E33 4E 7 0	35 126	\$#MFUI \$#RCID	0028 0040	7 6
4EB4	120	\$208A7	0070	10
4EC9	126	S#CFLD	0026	3.8

OLIOO I THE TOTAL CORE USED BY SAMPL2 IS 3823 DECIMAL.

JLIOI I THE START CONTROL ADDRESS OF THIS MODULE IS 4500.

JLIO4 I TOTAL NUMBER OF LIBRARY SECTORS REQUIRED IS 10

NAME-SAMPLZ, PACK-SYSTEM, UNIT-RI, RETAIN-T, LIBRARY-D

Figure 4-18 (Part 2 of 2). Source Listing of Sample Program SAMPL2 (Model 15)

	***************************************	*S************************************	**************************************	* K	*553ECORD.NJM9E911*	*		*************************************	*	************************************	*	*00*******************************	**************************************	*	*	*00600 · · · · · · · · · · · · · · · · ·	*0050*	*	*	*	#00055	# 	**************************************	***************************************	***************************************
OPTION-PL1 10-\$ IAR-4889 PMR-0072 PSR-0001 XR2-46A3 XR1-4019 ARR-48FC	GLOBAL (BUFFERS & IOBS) 0 40004000 00030062 3136437C 1303030 12903C14 AF421340 198312A5 00303003	0 E2010E40 70.080100 00129004 14AE000F 400083Z0 90C09043 08381053 FFF29014	0 0C014555 46EE3880 82F29008 3C484550 3C474369 C23245A4 F4133399 3C444553	0 C2024544 F4100082 350246E4 C0804348 88408EF2 90960C05 47018457 F0F040F0	3 F5F54040 39050306 09044035 F4040205 394040F1 F1404043 43434943 43434347	0 - 40404040 60606040 40604040 60606040 60606060	DUPLICATE LINES	0 FOFOFOFO F5F04343 40404043 40404043 40404040 4040404 41434141 4341411 4341411	0 40404040 40404040 40404040 4040400 40404040 40404040 4040403 40404040	3 - 40404040 43404040 40404040 40404040 40 090503 J5N40443 95E47402 05094040	0 - 40404040 40404340 40404040 43404340 40434343 434343 4347+34) 43E3EFE2	0 C0994000 005714A1 0000419E 00200000 129D4900 00671131 33334183 4537F3F0	3 FOFOF4F5 40404040 40404043 40404040 40494340 40404040 40404343 49434343	0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0	0	0 4040 4040 40404040 40404040 40404040 40404040 40404040 404040F0 F0F9F0F0	0 FOFOF5FO ¢0¢0¢0¢0 ¢0¢0¢0¢0 ¢3¢3¢0¢0 ¢0¢0¢0¢0 ¢0¢0¢3¢3 ¢0¢0¢3¢3 ¢3¢3¢3¢3	0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0	0	0	O FOFOF5F5 4040404 40404040 40404049 40404949 49494949	0+0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0+0+0+0 +0	3	0	0 FOFOF6FO 40404040 4,0404040 40404040 40404340 43434343 40404340 40404340
ď	000%	4020	4040	4363	4383	4040		4100	4120	4143	4160	4180	4143	4100	41E0	4200	4220	4240	4260	4280	42A0	42C0	42E3	4300	4320

Figure 4-19 (Part 1 of 5). Storage Dump of Sample Program SAMPL2 (Model 15)

*	**	**************************************	**************************************	*05000010000015000020*	*000025 000030 000035 00004*	*00000450000500000550*	*00060000065000070000075*	*****000080***000085***000090***	*0095000100000105000110.*	*303115000120000125000*	*1300001350001402*	*	*UMBER0120120001.4*	*J0J060RECORD.NUMBER12*	#		**	**************************************	*	* · · · · · · · · · · · · · · · · · · ·	**	***************************************	*W	# · · · · · · · · · · · · · · · · · · ·	*FFF
04040	0,40,40	F1F2C09D	0F0F0F0	F0000480	F0F0F0F4	001400F0	F0F0F7F5	2080F 0F 0	F1F1F000	00F0F0	97F29006	09C44005	F213038A	40404040	40404040		0000000	095C4500	01000000	0000000	00405200	FF46A346	09054040	cocococo	6900000
\$C\$0\$C\$ 0\$C\$0\$C\$	04C2C5D9 4040434	404040F0 F1F2	439E4637 FOF	FOFOFOF2 FOOG	000C00F0 F0F0	FOF3F5F5 3014	1880F0F0 F0F (F 0F 9F 000 208	00F0F0F0 F1F1	F2F50030 00F0	48484848 97F2) D9C5C3D6 D9C4	350246EA F210	F2404040 4040	\$0\$0\$0\$0\$0 \$9\$C		46334603 3903	00000003 0950	00800000 0100	000000000	2020980	08801FF	C4C9E2D2 C9D!	000000000000000000000000000000000000000	01000680 0000
40404343 43	C44005E4 04	05050505	83033333 43	000400F0 F	F0F0F3F5 0(1080F0F0 F(F0F7F330 18	00F0F0F0 F	F0F50028 00	F0F0F0F1 F2	F0003480 48	COMMON (FIELDS)	388053FE 3	094043F1 F2	40404040		30013332 46	43400000 00	00000000	00000000	34450000 83	00020340 0	00088001 C	0007000	26020305
40404040	05030609	04040404	40300006	FOFOF1F5	0880F0F0	F0F5F000	30F0F0F0	F8F50323	F 0F 0F 0F 1	F0002C80		COMN 0146F0F0	F4100083	E404C2C5	40404340		CONSTANT 40FFFF00	C0874C92	20000000	00000000	0041004E	20008445	FF00C700	00800000	00009000
40404040	40404000	43404340	09210000	0080F0F0	F0F3F000	03F OF OF O	F6F50018	FOFOFOFD	F0002480	F 0F 0F 1F 2	003400F0	BB80900C	F0F0F182	09044035	40404040		40404040	0.0874075	00000000	00000000	80450001	0000003A	10080000	C7100180	C680439E
40404040	40404040	40404040	43BC0020	F0F1F000	00F0F0F0	F4F50010	FOFOFOFO	F3301C80	FOFOF1FO	002500F0	FOF1F3F5	90F29009	F 0F1F2F0	09050306	4040404 40404040		40404040	00000000	30020000	00000000	00340000	460046A3	80418002	00000000	00000000
C+0+0+0+ 0+0+0+0+ 0+0+0+0+	40404040	40404040	80AL3333	00505050	F2F50008	FOFOFOFO	F0001480	FOFOFOFS	002400F0	FOFIFIF5	3080FJF0	460F3880	39F0F1F2	PRIME WORK AREA F0F0F0F0 F6F04040		LINES	40404040	00000000	000000040 000000000	SECONDARY WORK AREA 00000000 0000000 00000000		10CB2 000011330	00428341	41000000	90442700
40404040	40404040	40404040	40000005	F0F5000	FOFOFOFO	F0000C80	F OF OF OF 6	00100000	F3 F0 F9 F5	2880F0F0	F1F3F000	00014694	E404C2C5	FOFOFOFO F6F040	40404040	DUPL ICATE	40404040	00000000	00000000000000000000000000000000000000	SECONDA 000000000	10CB1 32344626	874E9F45	E6460041	00000000	33000 680
4340	4360	4380	4340	4300	43E0	4400	4450	0555	0555	6875	4440	4400	44E0	4500	4520		4580	45A0	4500	45E0	4500	4623	4540	4660	4580

Figure 4-19 (Part 2 of 5). Storage Dump of Sample Program SAMPL2 (Model 15)

•	**************************************	*8%2228*	*¢*	***************************************	*.2R2	*222	JP.1.2		**************************************	* · · · · · · · · · · · · · · · · · · ·	*.2	**************************************	*2*	**************************************	**************************************	*6"9*	**************************************	* *	*.4	***************************************	*********************************	*	**.28	*i6ppf2*
DIF2	42110642 46C53408 47123401 46E5E201 00750205 9C000F18 BD1200F2	0012COFF 47F85C01 123DBD41 OFF28218 F28109	00 F410008F 407A8000 020200C2 014500C0 87401047 15340847 AE340247 31750205	13 9C01000E 2C03475C 242C3347 6824C3FF 48348801 0FF21906 0C8F458F 45907300	+O OAF28148 382045D9 F210411C 0147850A 1C014789 JAB9FF12 F2100689 0812F204	50 1589FF10 F210068D 0810F204 098D483E F20119F2 873E8D48 3EF28108 3C394784	3C384788 34204506 3A204507 B0400EF2 02098901 0E8040E F21109C2	10000140 1000011 000010 0 0 0 0 0 0 0 0	CONSTANTS CONSTANTS)3	30F28104	\$	60 489F7501 08884039 80400FF2 960EAF01 77348601 77AE3177 34F28708 AF311934	33 860119AE 01193485 02003602 48E80F00 48E949F5 F2821036 0148E536 3248F59C	0 FF0000C0 87488E3D 0048EAF2 811C3C00 48E93601 48E43602 48F40C03 48C948A2	0E0048C7 48EA9C7F 0000C2	30 11C08748 3 0100FF FF008d34 08492878 4000F213 04F41333 337C330F 733132F2	90238C41 0EC08744 FD		+O 2E084E01 2E48AD01 2E4FF232 A18C014F 49F49E01 4F38AE31 4F548F01 4F49F48F	60 - 004749F8 AE014F48 40004761 F2927FC0 974805C3 9749394F 314F48FC 3347F287	30 6D340849 F1850158 4C01384F 9F01390B AE003737 AF003737 AC00369E 01371544	(3 6037F290 058E0137 49F83801 02F21034 4002384D F2023488 1029F293 31C38748	:0 EBB5014F 360149F6 3CFF4907 2E004907 5A2C0149 DA534300 000300F2 04108A08
6.6.4.4	4650	4550	4700	4723	4740	4760	4780	A A	4.7E0	4900	4820	4840	4860	4883	4840	4820	48E0	4903	4920	4940	096%	4980	4943	0365

Figure 4-19 (Part 3 of 5). Storage Dump of Sample Program SAMPL2 (Model 15)

4960	29F28707	AD02387C	F282038A	013AC087	483600FF	PFFFF000	20000301	\$\$SRRC 34.084AAF	* 2 **
4400	85014F90	023700AE	01361CAC	00383780	00378501	13884339	F2933385	10061110	***!
4423	3616F282	41F28120	8E003744	92990037	17F28459	8-01354A	84887036	F293058F	*22*
4440	01364486	C0874A1E	8C01194A	81 AE0119	38AE0119	17AD0119	15F20241	AF011917	***************************************
4460	BC0030F2	8745AE01	3817E201	2089FF37	F210268F	33374AB2	C0874805	BC OF 30 A F	*2S*
4A80	00381785	01138840	39F29003	05111058	013616F2	320AF287	04AF0138	17AC FF30	*
4AA0	BC00378C	01194481	4E011938	C087481E	FFFF0100	\$\$\$ 04002034	\$\$\$##I 34 084AE4B5	014F3631	* *
44 CO	4AE81C01	4AE6002E	0144E61C	85011388	4039F290	03850111	4C01164A	E6C08749	*.YMW2
44E0	092 08 748		SSRTC FF9C0100	08AE013D	198 50 100	4834AE31	19178040	0E950109	***************************************
4800	85100900	\$\$\$PBP 85100900 01340848	2F9C0030 174E	174E0016	4831 7860	15F29335	4E011648	33800030	**
4820	F281098F		00304830 C0874800 C087	C 08 7 00 00	01040020	\$\$LPRT 84080984	01088640	0EE20110	***************************************
6843	BA3414C0	87486EE2	01110087	486EB840	0FC0994B	FFE20112	8 A O 6 1 4 R A	01020087	***************************************
4B60	4B6E0C01	48F84854	£23113F2	87043408	4BFB7033	00F28170	90001500	35011778	*.>
4360	1000F210	04541000	03750118	84011770	4032F291	67883132	F29322C	00488618	*24*
48A0	00024885	48863000	48841601	48850885	01001083	40FF83B5	31178831	029C031E	*
4800	16600108	15641000	02753118	9E001E08	780407F2	90049006	1E088802	14AE001E	*.6
48E0	15880414	27	331E15AD	001E1CF2	34038648	0E880414	C08748FF	B1410EB5	*******
\$C 00	01088510	09350200	2EC20145	005F13DA	DA6C03C5	82740162	78436375	01997502	*
4C 20	05F41000	05F41000 83750205	F4100082	10004032	10380045	C5F2933D	78033379	10005290	*.4
6643	04308740	55400103	4599F281	07750103	C0874C2C	F28709C2	024584F4	113808611	*<
4,060	C2014500	30404564	JC1144E3	44E43CF0	44EE0C08	44E044EE	5C8F8F90	52024500	*B**
4080	00044686	47E5C087	4E55C087	4E49C087	400 479 20		F2900378	20067820	*V+++(DPQ2#.O#.*
\$C40	08780259	78F1C278	D87802C9 78F1C278 9FC378FF	C47A40C3	78200300	13406100	01400840	2E750299	*Q*.[#18#.C#.D:.CC<*
4cc0	881000F2	10153001	40104032	1E014CCB	97890100	89830100	10401688	8000F293	*2(.[<a(2.*< td=""></a(2.*<>
43E0	0674FFC4	0674FEC4 7AE003C0	87460900	8740F5C0	874E013C	324015F2	87003633	40158801	*::.):.C+I(6+[.2(*
4000	18F29003	80001088	3100E201	00020245	B4F41000	9F037408	3CC0874D	31898330	*.2SB4
4020	F21004C0	874E7DBA	874ETDBA 31307510	30 403040	31020147		C5750298	B90218F2	*2+*(.(.B8E2*
4040	10002001	40401E85	02120087	4052C087	40652005	44D7052C	06446450	24024E7	******(0*****)*************************

Figure 4-19 (Part 4 of 5). Storage Dump of Sample Program SAMPL2 (Model 15)

	DETAIL CALCS CLOSE	
4050	606_2044EA47_E0C0874D	***(**[2:(D:.CR:.R2.*
4080	04C0874D 94350145 99750205 F4100083 F4100084 340840C3 5C8F8F90 C2024500	*(C*B*
4040	OC044686 47888CO2 0244EABA F0029802 00908C15 1947CE8C 102847DF C0874E49	***************************************
4000	C0870000 5C8F8F90 C2024500 0C044686 47837802 C9F2901A 8C050544 078C0C14	**B
4DE0	44E48C02 1844E7BA F0189B02 1690C087 4E49C087 4C927820 C3C0104D 78C0874E	**)5+0.XV.*
4E00	LR AND OVERFLOW PROCESSING 01/7820C3 C0104085 782007C0 104E257B FF0778FF 087A02C2 7804C2F2 10037802	*C[
4620	-8F90 C2024500 OCC	*8[+
4E 40	4E497B20 D9C08700 0GC20248 34C20146 26351047 1434084E 7C8C0204 47E88C0A	*+. #.R B B +0
4E 60	1547F38C 031047F7 163044EE 958C0322 44EEBAF0 229B031F 90C0874C 8AB50112	*370*
4E80	RECOMDID 85101434 084E8AC2 014EBE9C 0A220AF2 871AB801 18F29333 B8071CC0 874CF3B8	**+.B.+22<3,*
4EA0	0118F290 03BB071C COB74CFA C2014500 CO874D27 70F000F2 010BC087 4E838202	*2
4EC0	C9803040 52C0874E 9F7820C3 F2101535 02459988 01002C01 4EDD1A7A 02C97880	*1(+C2+:.1*
4EE0	END SAMPLZ 09F29304 C3874CEB 7A80D9C0 874CEHFF 05C3C540 04C12935 E3C105C3 C540D4C1	*R2 <.: R < NCE . MAINT ANCE . MA*
4F00	62626262 62626262 62626262 62626262 62626262 62626262 62626262 62526262	*\$
	DUPLICATE LINES	
5300	E2E2E2E2 E2E2E2E2 E2258C25 B02594FF 25943F26 38265226 38002638 J0258426	株************************************
53E0	142584FF 25842126 18263426 18002618 30258825 84038A10 0A4500EF 45034500	将
2400	00000000 00000000 00000000 00000000 0000	4
	DUPLICATE LINES	
5440	00000000 00000000 10000000 00000000 1000000	***
5460	14400000 00000000 00000000 00000000 000000	*
5480	00000000 00000000 00000000 00000000 0000	******************
	DUPLICATE LINES	
2400	00000000 00000000 33E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2	#*************************************
54E0	62626262 62626262 62626262 62626262 62626262 62626262 62626262 62626262	#SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

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.

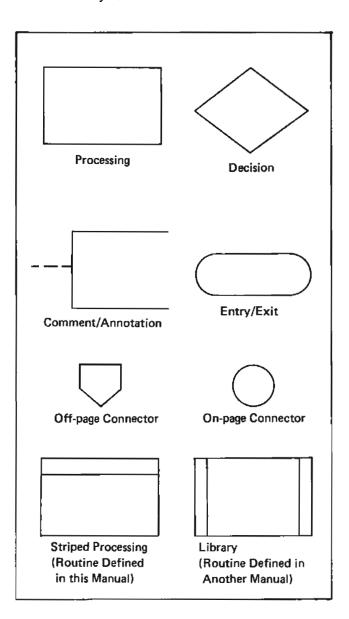
Example: First page, CA-01

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

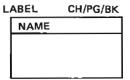
The flowchart symbols used in this PLM are:



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:

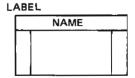


CH/PG/BK indicates the flowchart, page, and block identification where the module or routine is flowcharted. If it is not flowcharted, see the index for the location of the description of that routine.

Library Block

A library block indicates a function or module is documented in another manual.

Example:

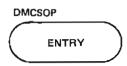


The NAME of the function/module is listed in the *Preface* of this manual under the name of the manual that contains the description of this function/module.

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.

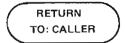
Example:



Exit Block

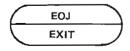
This block indicates that control is leaving this chart.

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

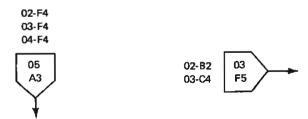
Off-page connectors are used to reference between different page of the same chart ID. Off-page connectors leaving a page contain the page number and block number of their destination.

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 pro-

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 'l	D' to identify it as a
	dump control ca	rd.
7	Dump function i	dentification
	character	
	C - Main stora	ge dump
	D - Disk dump	-
	H - Halt on ph	ase load
	1 - Compression	
	T - Print text 1	blocks
9-10	Start phase ID	Different meanings
12-13	End phase ID	depending on dump
15-18	Lower limit	function identifica-
20-23	Upper limit	tion character in
25	Compression/	column 7.
	file ID	See the specific
	/	function for detail.
		function for detail.

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 = 1 (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.

Page of LY21-0501-5 Issued 24 September 1976 By TNL: LN21-5423

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
                         BRANCH TO SET ARR TO PARM LIST
         \mathbf{B}
                         PARM FOR TYPE OF DUMP
             XL1'XX'
         DC
                         LOWER CORE OR COMPRESSION UNIT
         DC
             XL2'LLLL'
         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
             DUMP FILE $SOURCE
         01
         00
             'F' COMPRESSIONS
         02
             'E' COMPRESSIONS
         04
             'L' COMPRESSIONS
             DATA MANAGEMENT NAME COMPRESSIONS
         06
         80
             'I' COMPRESSIONS
         OA
             'C' COMPRESSIONS
         00
             DEBUG COMPRESSIONS
             SPARE COMPRESSIONS
         0E
             'O' COMPRESSIONS
         10
             'T' COMPRESSIONS
         12
         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.

*PLACE, address of 4-5	\$RPGY - build compile-time tables phase 2-7
	\$RPGZ - compile-time table/array code phase 2-7
	\$RPHA - build symbol table phase 2-6
	\$RPHC - check symbol table phase 2-6
\$\$PGAA – array index subroutine 4-41	\$RPHD - print symbol table phase 2-7
\$\$PGAB – file translate subroutine 4-41	\$RPHQ - assign alternate collating sequence and file
\$\$PGAC - square root subroutine 4-41	translate tables phase 2-8
\$\$PGBB - BSCA logic for transmit and receive subroutine 4-42	\$RPHS - check input specifications 2 phase 2-8
\$\$PGBC - BSCA logic for conversational subroutine 4-42	\$RPHT - assign control and match field hold areas and
\$\$PGBG - BSCA logic for receive subroutine 4-42	edit code patterns phase 2-9
\$\$PGBI - convert to binary subroutine 4-42	\$RPHU - assign limits file hold areas phase 2-9
\$\$PGBO - convert to decimal subroutine 4-43	\$RPIC - post compression initialization phase 2-5
\$\$PGBP - BSCA logic for transmit subroutine 4-43	\$RPIG - check file description continuation cards phase 2-5
\$\$PGCB - command key indicator set routine 4-43	\$RPJA - check input specifications 1 phase 2-6
\$\$PGCl – unpack subroutine 4-43	\$RPJE - check calculations specifications 1 phase 2-5
\$\$PGCO pack subroutine 4-44	\$RPIG - check calculations specifications 3 phase
\$\$PGDC - DEBUG subroutine 4-44	(Model 6 only) 2-9
\$\$PGDI - alternate collating sequence subroutine 4-44	\$RPJJ - control field and match field move optimization
\$\$PGDP - display subroutine (Model 15 only) 4-45	phase 2-9
\$\$PGFI - load object tables subroutine 4-45	\$RPJK - check file and table/array phase 2-6
\$\$PGFO - dump object tables subroutines 4-46	\$RPJL - check calculation specifications 4 phase 2-9
\$\$PGIC – divide subroutine 4-46	\$RPJM - check output-format specifications 1 phase 2-10
\$\$PGLC - LOKUP subroutine 4-47	\$RPJN - check output-format specifications 2 phase 2-10
\$\$PGLG - command key indicator light restorer routine 4-47	\$RPJO - check output-format specifications 3 phase 2-10
\$\$PGMA – move array subroutine 4-48	\$RPJP - check input specifications 3 phase 2-9
\$\$PGMC - multiply subroutine 4-48	\$RPJS - header/trailer assign and diagnostic phase 2-9
\$\$PGRI – set resulting indicators subroutine 4-49	\$RPJU - check compile-time tables/array phase 2-7
\$\$PGTC - test zone subroutine 4-49	\$RPJW - check file description specifications 1 phase 2-5
\$\$\$YRP - RPG II halt processor 4-49, 4-70, 4-71	\$RPJX - check file description specifications 2 phase 2-5
\$\$SYR1 - RPG II halt library 1 4-49, 4-72	\$RPJY - check extension and line counter phase 2-10
\$RPEA - file description compression phase 2-3	\$RPJZ - check file description specifications 3 phase 2-5
\$RPEB - extension compression phase 2-3	\$RPKA - error sort and print phase 2-10
\$RPEC - line counter compression phase 2-3	\$RPKB - error message list phase 2-10
\$RPEE - telecommunications compression phase 2-3	\$RPLB - output indicator optimization phase 2-13
\$RPEI - input compression phase 2-3	\$RPLG - DTF parameter assign 1 phase 2-13
\$RPEK - calculation compression phase 2-3	\$RPLJ - pre-assemble calculations 3 phase
\$RPEO - output compression phase 2-4	(Model 6 only) 2-13
\$RPEW - post compression and initialization phase 2-5	\$RPLN - constant, literal, edit word assign phase 2-13
\$RPFA - copy tables into compression phase 2-4	\$RPLR - constant, literal edit word, DTF parameter assign
\$RPG - compiler initialization phase 2-3	11 phase 2-13
\$RPGB - process MFCM print specifications phase 2-6	\$RPLV - output fields move optimization phase 2-13
\$RPGF - assign and check indicators phase 2-5	\$RPLZ - output indicator testing phase 2-14
\$RPGG - check calculation specifications 2 phase 2-5	\$RPMA - output object code block length phase 2-14
\$RPGH - assign filename and IOCB phase 2-7	\$RPMB - pre-assemble calculations 1 phase 2-13
\$RPG1 - assign filename and DTF phase 2-7	\$RPMG - pre-assemble calculations 2 phase 2-13
\$RPGJ - remote terminal/device support 2-8	\$RPMH - pre-assemble calculations 4 phase 2-14
\$RPGK - check telecommunications specifications 1 phase 2-7	\$RPMI - pre-assemble indicator optimization phase 2-14
\$RPGL - check telecommunications specifications 2 phase 2-7	\$RPMK - build segment list phase 2-14
\$RPGN - build DTFs phase 2-8	\$RPMM - output segment list entries build phase 2-14
\$RPGR - assign I/O areas MFCU1, MFCU2, READ42, SPECIAL	\$RPMP - build calculation segment list phase 2-15
MFCM1, MFCM2, READ01, DISKET (Models 6 and 10)	\$RPMQ - completes calculation segment list phase 2-15
phase 2-8	\$RPPA - generate IPCR phase 2-11
	\$RPPB - generate OPCR phase 2-11
\$RPGT – assign I/O areas phase CONSOLE, DISK, PRINTER,	\$RPPC - input mainline code phase 2-11
PRINTR2, PRINT84 BSCA 2-8 SPPCS assign shared I/O and other devices areas phase 2.8	\$RPPE - build input record recognition 1 code phase 2-11
\$RPGS - assign shared I/O and other device areas phase 2-8	\$RPPF - build input record recognition 2 code phase 2-11
\$RPGU - build name table phase 2-5	\$RPPG - file selection and match field extraction code phase 2-11
\$RPGV - check table/array phase 2-6	\$RPPJ - control field extraction code phase 2-12
\$RPGW - check name table phase 2-6	

\$RPGX - print name table phase 2-6

\$RPPL - look-ahead field extraction code phase 2-12	alignment 4-27
\$RPPM - input field extraction code phase 2-12	alphameric fields, blank after 4-5
\$RPPN - chain and read files codes phase 2-15	alternate collating sequence subroutine (\$PGDI) 4-44
\$RPPO - LR and overflow control mainline code phase 2-15	alternate collating sequence, file translate, and compile-time
\$RPPS - output record code phase 2-15	table/array compressions 3-51
\$RPPU - output fields 1 code phase 2-16	alternate collating sequence and translate tables 4-76
\$RPPV - output fields 2 code phase 2-16	alternate collating sequence table
\$RPPW - output fields 3 code phase 2-16	compile-time 3-51
\$RPQA - calculations 1 code phase 2-16	object time 4-76
\$RPQB - calculations 2 code phase 2-16	аггау
\$RPQD - calculations 3 code phase 2-16	array addressing (see array addressing)
\$RPQE - calculations 4 code phase 2-16	array index subroutine (\$\$PGAA) 4-41
\$RPQF - calculations 5 code phase 2-17	entire, use of 4-26
\$RPQG - calculations 6 code phase 2-17	array addressing
\$RPQH - Calculations 1.5 code phase 2-16	calculation, use of
\$RPQK - calculations 7 code phase 2-17	array with integer index 4-27
\$RPQL – calculations 8 code phase 2-17	array with variable index 4-27
\$RPQT - calculations 9 code phase 2-17	output fields, use of
\$RPQU - calculations 10 code phase 2-17	array with numeric index 4-5
\$RPQV - calculations 11 code phase 2-17	array with variable index 4-5
\$RPRA – initialization and close code phase 2-18	array with no index 4-5
\$RPRC - table load/table dump code phase 2-18	array index subroutine (\$\$PGAA) 4-41
\$RPRW - chain and read files move field code phase 2-15	array with integer index, calculation addressing 4-27
\$RPRX – initialize segment list phase (Models 6 and 10) 2-19	array with no index, output fields addressing 4-5
\$RPRX - reformat segment list phase (Model 15) 2-21	array with numeric index, output fields addressing 4-5
\$RPRY - library of subroutines code phase (Models 6	array with variable index
and 10) 2-19	calculation addressing 4-27
\$RPRY - sort object code phase (Model 15) 2-21	output fields, addressing 4-5 assemble 1 phases 2-11
\$RPRZ - generate R-module output phase 2-21	build input record recognition 1 code (\$RPPE) 2-11
\$RPSA - resolve EXTRN phase 2-19	build input record recognition 2 code (\$RPPF) 2-11
\$RPSB – overlay editor 1 phase 2-19	control field extraction code (\$RPPJ) 2-12
\$RPSC calculate start address phase 2-19 \$RPSD print segment list phase 2-19	file selection and match field extraction code (\$RPPG) 2-11
\$RPSE – sort object code phase 2-19	input field extraction code (\$RPPM) 2-12
\$RPSF – overlay editor 2 phase 2-19	input mainline code (\$RPPC) 2-11
\$RPSG – overlay fetch and transfer vector code phase 2-20	generate IPCR (\$RPPA) 2-11
\$RPSI – final output generation 1 phase 2-20	generate OPCR (\$RPPB) 2-11
\$RPSK - final output generation 2 phase 2-20	look-ahead field extraction code (\$RPPL) 2-12
\$RPSN – sort segment list phase 2-19	assemble II phases 2-15
\$RPSP - eliminate duplicate segment list entries phase 2-19	build calculation segment list (\$RPMP) 2-15
\$RPXA - build DTF and I/O area for DISK45 phase 2-8	calculations 1 code (\$RPQA) 2-16
\$RPXB - build DTF and I/O area for tape phase 2-9	calculations 1.5 code (\$RPQH) 2-16
\$SOURCE, use of 3-61	calculations 2 code (\$RPQB) 2-16
\$WORK, use of 3-61	calculations 3 code (\$RPQD) 2-16
	calculations 4 code (\$RPQE) 2-16
	calculations 5 code (\$RPQF) 2-17
	calculations 6 code (\$RPQG) 2-17
abbreviations used in calculations 4-26	calculations 7 code (\$RPQK) 2-17
ADD object code 4-28	calculations 8 code (\$RPQL) 2-17
addresses used	calculations 9 code (\$RPQT) 2-17
in calculations 4-26	calculations 10 code (\$RPQU) 2-17
аттау with integer index 4-27	calculations 11 code (\$RPQV) 2-17
array with variable index 4-27	chain and read files code (\$RPPN) 2-15
entire array 4-27	chain and read files move field code (\$RPRW) 2-15
table elements 4-27	initialization and close code (\$RPRA) 2-18
with output fields 4-5	LR and overflow control mainline code (\$RPPO) 2-15
*PLACE 4-5	output fields 1 code (\$RPPU) 2-16
array with numeric index 4-5	output fields 2 code (\$RPPV) 2-16
array with variable index 4-5	output fields 3 code (\$RPPW) 2-16
array with no index 4-5	output record code (\$RPPS) 2-15
Editing 4-5	table load/table dump code (\$RPRC) 2-18
PAGE 4-5	assign alternate collating sequence and file translate
Punch 4-5	tables phase (\$RPHQ) 2-8
table element 4-5	assign and check indicators phase (\$RPGF) 2-5
Update FILES 4-5	

assign and diagnostic phases 2-5	BSCA logic for conversational subroutine (\$\$PGBC) 4-42
assign alternate collating sequence and file translate	BSCA logic for receive subroutine (\$\$RGBG) 4-42
tables (\$RPHQ) 2-8	BSCA logic for transmit subroutine (\$\$RPGP) 4-43
assign and check indicators (\$RPGF) 2-5	BSCA logic for transmit and receive subroutine (\$\$PGBB) 4-42
assign control and match field hold areas and edit	buffer, input and output 4-77
code patterns (\$RPHT) 2-9	build calculation segment list phase (\$RPMP) 2-15
assign filename and IOCB (\$RPGH) 2-7	build compile-time tables phase (\$RPGY) 2-7
assign filename and DTF (\$RPGI) 2-7	build DTF's phase (\$RPGN) 2-8
assign I/O areas (\$RPGR) 2-8	build DTF and I/O area for DISK45 phase (\$RPXA) 2-8
assign limits file hold areas (\$RPHU) 2-9	build DTF and I/O area for tape phase (\$RPXB) 2-9
assign shared I/O and other device areas (RPGS) 2-8	build input record recognition 1 code phase (\$RPPE) 2-11
build compile-time tables (\$RPGY) 2-7	build input record recognition 2 code phase (\$RPPF) 2-11
build DTFs (\$RPGN) 2-8	build name table phase (\$RPGU) 2-5
build DTF and I/O area for DISK45 (\$RPXA) 2-8	build segment list phase (\$RPMK) 2-14
build DTF and I/O area for tape (\$RPXB) 2-9	build symbol table phase (\$RPHA) 2-6
build name table (\$RPGU) 2-5	
check calculation specification 1 (\$RPJE) 2-5	calculation addressing 4-26
check calculation specification 2 (\$RPGG) 2-5	array with integer index 4-27
check calculation specification 3 (\$RPJG) 2-9	array with variable index 4-27
check calculation specification 4 (\$RPJM) 2-10	entrie array 4-27
check compile-time tables/arrays (\$RPJU) 2-7	table elements 4-27
check file and table/array (\$RPJK) 2-6	calculate start address phase (\$RPSC) 2-19
check file description continuation cards (\$RPIG) 2-5	calculation compression 3-32
check extension and line counter (\$RPJY) 2-10	calculation compression phase (\$RPEK) 2-3
check file description specifications I (\$RPJW) 2-5	calculations object code 4-26
check file description specifications 2 (\$RPJX) 2-5	ADD 4-28
check file description specifications 3 (\$RPJZ) 2-5	address of array with integer index 4-27
check input specifications 1 (\$RPJA) 2-6 check input specifications 2 (\$RPHS) 2-8	address of array with variable index 4-27
	address of entire arrays 4-27
check input specifications 3 (\$RPJP) 2-9 check name table (\$RPGW) 2-6	address of table elements 4-27
	BEGSR 4-28
check output-format specification I (\$RPJM) 2-10 check output-format specifications 2 (\$RPJN) 2-10	BITON 4-28 BITOF 4-28
check output-format specifications 2 (\$RPJO) 2-10	CHAIN 4-28
check symbol table (\$RPHC) 2-6	COMP 4-28
check table/array (\$RPGV) 2-6	DEBUG 4-29
check telecommunications specifications 1 (\$RPGK) 2-7	DIV 4-29
check telecommunications specifications 2 (\$RPGL) 2-7	DSPLY 4-30
compile-time table/array code (\$RPGZ) 2-7	ENDSR 4-30
control field and match field move optimization (\$RPJJ) 2-9	EXCPT 4-30
error message list (\$RPKB) 2-10	EXIT 4-30
error sort and print (\$RPKA) 2-10	EXSR 4-30
header/trailer assign and diagnostics (\$RPJS) 2-9	FORCE 4-30
print name table (\$RPGX) 2-6	GOTO 4-31
print symbol table (\$RPHD) 2-7	KEY 4-31
post compression and initialization (\$RPEW) 2-5	LOKUP 4-31
post compression initialization (\$RPIC) 2-5	MHHZO 4-32
remote terminal/device support (\$RPGJ) 2-8	MHLZO 4-32
assign control and match field hold areas and edit code	MLHZO 4-32
patterns phase (\$RPHT) 2-9	MLLZO 4-32
assign filename and DTF phase (\$RPGI) 2-7	MOVE 4-32
assign filename and IOCB phase (\$RPGH) 2-7	MOVEA 4-33
assign I/O areas-CONSOL, DISK, PRINTER, PRINTR2,	MOVEL 4-33
BSCA (\$RPGT) 2-8	MULT 4-33
assign I/O areas-MFCU1, MFCU2, READ42, SPECIAL,	MVR 4-34
phase (\$RPGR) 2-8	obtaining addresses 4-26
assign limits file hold areas (\$RPHU) 2-9	preserving decimal integrity 4-27
assign phases (see assign and diagnostic phases)	READ 4-34
assign shared I/O and other device areas phase (\$RPGS) 2-8	RLABL 4-34
	see also detail calculations and total calculations
begin subroutine 4-28	SET 4-34
BEGSR subroutine 4-28	SETLL operation code 4-34
BITOF object code 4-28	•
BITON object code 4-28	

calculations object code (continued)	command key indicator light restorer routine (\$\$PGLG) 4-47
SETOF 4-35	command key indicator set routine (\$\$PGCB) 4-43
SETON 4-35	COMMON 3-1
specification descriptions 4-28	COMP object code 4-28
SQRT 4-35	compare object code 4-28
SUB 4-35	compile-time data areas 3-1
TAG 4-35	compile-time symbol table 3-54
TESTB 4-35	compile-time table/array code phase (\$RPGZ) 2-7
TESTZ 4-36	compile-time table/array compression 3-54
use of conditioning indicators 4-26	compiler control region 3-1
use of entire arrays 4-26	introduction to 1-3
use of RPG II subroutines 4-26	compiler initialization phase (\$RPG) 2-3
use of tables 4-26	compiler operation 1-1
use of XR1 4-26	compiler overview 1-2
XFOOT 4-36	completes calculation segment list phase (\$RPMQ) 2-15
Z-ADD 4-36	completion codes from data management 4-77
Z-SUB 4-36 calculation segment list phase (\$RPMQ) 2-15	compressions 3-18 alternate collating sequence 3-54
calculation specifications descriptions 4-28	calculation 3-34
calculations 1 code phase (\$RPQA) 2-16	compile-time table/array 3-54
calculations 1.5 code phase (\$RPQH) 2-16	control statement 3-17
calculations 2 code phase (\$RPQB) 2-16	data management entry point and module names 3-54
calculations 3 code phase (\$RPQD) 2-16	extension 3-18
calculations 4 code phase (\$RPQE) 2-16	file description 3-17
calculations 5 code phase (\$RPQF) 2-17	file translate 3-54
calculations 6 code phase (\$RPQG) 2-17	input 3-28
calculations 7 code phase (\$RPQK) 2-17	line counter 3-18
calculations 8 code phase (\$RPQL) 2-17	output-format 3-44
calculations 9 code phase (\$RPQT) 2-17	phase load 3-60
calculations 10 code phase (\$RPQU) 2-17	telecommunications 3-50
calculations 11 code phase (\$RPQV) 2-17	compression block table (CZATAB) 3-17
CHAIN object code 4-28	compression phases (see input and compression phases) 3-17, 3-18
chain and read 4-6	compression work area 3-17
chain and read files codes phase (\$RPPN) 2-15	conditioning indicators, use of by calculations 4-26
chain and read files move field code phase (\$RPRW) 2-15	constant area 4-73
chain table 3-54	constant, literal, edit word assign phase (\$RPLN) 2-13
check calculation specifications 1 phase (\$RPJE) 2-5	constant, literal, edit work assign II phase (\$RPLR) 2-13
check calculation specifications 2 phase (\$RPGG) 2-5	constants 4-76
check calculation specifications 3 phase (Model 6 only)	control field and match field move optimization phases
(\$RPJG) 2-9	(\$RPJJ) 2-9
check calculation specifications 4 phase (\$RPJL) 2-9	control field extraction code control phase (\$RPPJ) 2-12
check compile-time tables/arrays phase (\$RPJU) 2-7	control fields logic and move 4-6
check extension and line counter phase (\$RPJY) 2-10	control fields move (see control fields logic and move)
check file and table/array phase (\$RPJK) 2-6	control fields save areas 4-76
check file description continuation cards phase (\$RPIG) 2-5	control routine save area 3-16
check file description specifications 1 phase (\$RPJW) 2-5	control statement compression 3-17
check file description specifications 2 phase (\$RPJX) 2-5	convert to binary subroutine (\$\$PGBI) 4-42
check file description specifications 3 phase (\$RPJZ) 2-5	convert to decimal subroutine (\$\$PGBO) 4-43
check input specifications 1 phase (\$RPJA) 2-6	copy tables into compression phase (\$RPFA) 2-4
check input specifications 2 phase (\$RPHS) 2-8	crossfoot (see XFOOT)
check input specifications 3 phase (\$RPJP) 2-9 check name table phase (\$RPGW) 2-6	CZATAB 3-17
check output-format specifications 1 phase (\$RPJM) 2-10	
check output-format specifications 2 phase (\$RPJN) 2-10	
check output-format specifications 3 phase (\$RPJO) 2-10	
check symbol table phase (\$RPHC) 2-6	
check table/array phase (\$RPGV) 2-6	
check telecommunications specifications 1 phase (\$RPGK) 2-7	data areas
check telecommunications specifications 2 phase (\$RPGL) 2-7	compile-time
close a compression area routine (DRGCZE) 2-24	alternate collating sequence, file translate, and
close a compression block routine (DRGCZK) 2-25	compile-time table/array compressions 3-54
close mainline 4-7	chain table 3-54
detailed object program flow 4-25	COMMON 3-1
intermediate object program flow 4-4	compile-time symbol table 3-54
overall object program flow 4-2	compiler control region 3-1

data areas (continued)	detail output
compile-time (continued)	intermediate object program flow 4-3
compression block table (CZATAB) 3-17	overall object program flow 4-2
compression formats 3-18	(see also output)
compression work area 3-17	diagnostic phases (see assign and diagnostic phases)
control routine save area 3-16	disk control routine (PIOCS) 2-24
data management entry points and module names	disk input and output control phases (see interphase control
compression 3-54	routines)
error file 3-55	display object code 4-30
final segment list 3-57	display subroutine, Model 15 (\$\$PGDP) 4-45
file input/output table 3-56	DIV (divide) object code 4-29
filename table 3-57	divide subroutine (\$\$PGIC) 4-46
general storage table 3-58	DRGCG1-find item in compression routine 2-25
I/O parameters 3-16 internal symbol table 3-58	DRGCZA—open a compression area routine 2-24
IOB 3-16	DRGCZC—write a compression routine 2-24
name table 3-59	DRGCZE—close a compression area routine 2-24
object code block 3-59	DRGCZG-open a compression block routine 2-25
parameters to RPG II halt processor 3-16	DRGCZH-get next compression routine 2-25
phase load compression 3-60	DRGCZK-close a compression block routine 2-25
segment list 3-60	DRGCZM-write an object code block routine 2-25
symbol table 3-60	DRGCZN-get next source record routine 2-24
telecommunications table 3-60	DRGCZP—printer control routine 2-24
text-RLD record 3-61	DRGCZZ-call next compiler phase routine 2-24
object-time	DSPLY object code 4-30
alternate collating sequence and translate tables 4-76	DTF 4-75
constant area 4-73	DTF parameter assigning (\$RPLG) 2-13
constants, edit words, and edit codes 4-76	DTT (define the table) 4-75
control field save areas 4-76	dump analysis, sample 4-90 dump control card format B-1
data management completion codes 4-77	dump control card format B-1 dump control compressions (see dump facility)
data management parameters 4-81	dump control phase 2-1
define the files (DTF) 4-75	dump facility B-1
define the table (DTT) 4-75	dump object tables subroutine (\$\$PGFO) 4-46
error recovery procedure (ERP) area 4-77	dump object tables suprodume (\$\$1 Gr O) 4-40
indicator table 4-74	
input and output buffers 4-77	
input/output control block (IOCB) 4-78	edit codes and edit words
match field save areas 4-76	data area 4-76
prime work area 4-73	editing 4-5
reserved object communications area (ROCA) 4-73	editing, edit codes and edit words 4-5
secondary work area 4-75	editor (see overlay editor)
trailer table 4-75	elements
data management	table elements
completion codes from 4-77	calculation addressing 4-26
parameters to 4-81, 4-82	output fields addressing 4-1
data management entry points and module names	eliminate duplicate segment list entries phase (\$RPSP) 2-19
compression 3-54	end of job (see program close mainline)
DEBUG object code 4-29	ENDSR (end subroutine) 4-30
DEBUG subroutine (\$\$PGDC) 4-44	ENDSR object code 4-29
decimal	EOJ routine (see program close mainline)
convert to decimal subroutine (\$\$PGBO) 4-43	ERP (error recovery procedure area) 4-77
pack subroutine (\$\$PGCO) 4-44	error
unpack subroutine (\$\$PGCI) 4-43	file 3-55
decimal halt adjusting 4-27	message list phase (\$RPKB) 2-10
decimal integrity, methods of preserving 4-27	sort and print phase (\$RPKA) 2-10
decimal point alignment 4-27	error file 3-55
define the file (DTF) 4-75	error message list phase (\$RPKB) 2-10
define the table (DTT) 4-75	error sort and print phase (\$RPKA) 2-10
letail calculations	exception output object code 4-30
intermediate object program flow 4-4	EXCPT object code 4-30
overall object program flow 4-2	EXIT object code 4-30
(see also calculations)	explanation of program organization flowcharts 2-22
letailed object program flow 4-1	EXSR execute subroutine object code 4-30
	EXSR object code 4-30

extension compressions 3-18	index register 1, use of 4-26 indicator
extension compression phase (\$RPEB) 2-3	command key indicator light restorer routine (\$\$PGLG) 4-47
	command key indicator set routine (\$\$PGCB) 4-43
fatali avantlavi. A 6	pre-assemble indicator optimization phase (\$RPMI) 2-14
fetch overflow 4-6 fields (see matching fields; input field; output field;	set resulting indicator subroutine (\$\$PGRI) 4-49
control field)	initialization and close code phase (\$RPRA) 2-18
file	initialization of the compiler 2-3
error file 3-55	initialize segment list (\$RPRX) 2-19
file description compressions 3-17	input/output control block (IOCB) 4-78
file description compressions phase (\$RPEA) 2-3	input and compression phases 2-3
file input/output table 3-56	calculation compression (\$RPEK) 2-3
filename table 3-57	compiler initialization (\$RPG) 2-3
files (see multifiles)	copy tables into compression (\$RPFA) 2-4
file selection and match field extraction code phase	extension compression (\$RPEB) 2-3
(\$RPPG) 2-11	file description compression (\$RPEA) 2-3
file translate compression 3-54	input compression (\$RPEI) 2-3
file translate subroutine (\$\$PGAB) 4-41	line counter compression (\$RPEC) 2-3
final output generation 1 phase (\$RPSI) 2-20	output-format compression (\$RPEO) 2-4 telecommunications compressions (\$RPEE) 2-3
final output generation 2 phase (\$RPSK) 2-20	input and compression control routines (see interphase control
final segment list (Models 6, 10, and 12) 3-57	routines)
final segment list (Model 15) 3-57	input and output buffers 4-77
find item in compression routine (DRGCGI) 2-25	input compression 3-18
finding an overlay 4-84	input compression phase (\$RPEI) 2-3
FIOT 3-56	input fields
flowchart techniques compile-time 2-22	extraction code phase (\$RPPM) 2-12
general A-1	mainline object code 4-6
object program 4-1	input fields extraction code phase (\$RPPM) 2-12
FORCE, object code 4-30	input fields mainline
10002,00,000	intermediate object program flow 4-4
	overall object program flow 4-2
	input mainline
general object program flow 4-2	detailed object program flow 4-5
general overlay flow 4-85	intermediate object program flow 4-3
general storage table 3-58	overall object program flow 4-2
generate IPCR phase (\$RPPA) 2-11	input mainline code phase (\$RPPC) 2-11
generate OPCR phase (\$RPPB) 2-11	input phases (see input and compression phases)
generate R-module output phase (\$RPRZ) 2-21	input processing control 4-1 input processing control routine (IPCR)
get next compression routine (DRGCZH) 2-25	detailed object program flow 4-1
get next source record routine (DRGCZN) 2-24 GOTO object code 4-31	generation phase 2-11
GOTO object code +51	integer index, calculation addressing 4-27
	integrity, decimal 4-27
	intermediate object program flow 4-3
half adjusting, decimal 4-27	internal symbol table 3-58
halt (see error)	interphase control routines 2-22
halt processor 4-49, 4-70, 4-71, 4-72	call next compiler phase routine (DRGCZZ) 2-24
parameters to 3-16	close a compression area routine (DRGCZE) 2-24
header/trailer assign and diagnostic phase (\$RPJS) 2-9	close a compression block routine (DRGCZK) 2-25
hold area, control field (see control field save area)	find item in compression routine (DRGCGI) 2-25
how this publication is organized ii	get next compression routine (DRGCZH) 2-25
	get next source record routine (DRGCZN) 2-24
	open a compression area routine (DRGCZA) 2-24
	open a compression block routine (DRGCZG) 2-25
I/O parameter 3-16	printer control routine (DRGCZP) 2-24 write a compression routine (DRGCZC) 2-24
index	write a compression routine (DRGCZC) 2-24 write an object code block routine (DRGCZM) 2-25
array index subroutine (\$\$PGAA) 4-41	Write all object code block foutflie (DROCEM) 2-23
calculation addressing 4-26	
array with integer index 4-27 array with variable index 4-27	
output fields addressing 4-1	
array with numeric index 4-5	
array with numeric index 4-5	
array with no index 4-5	

introduction 1-1	look up (see LOKUP)
compiler control region 1-3	LR and overflow control mainline
compiler operation 1-1	detailed object program flow 4-6
compiler overview 1-2	intermediate object program flow 4-3
linkage between phases 1-3	overall object program flow 4-2
main storage map 1-3	LR and overflow control mainline code phase (\$RPPO) 2-15
IOB (input/output block) 3-16	•
IOCB (input/output control block) 4-78	
IPCR (input processing control routine) 4-1	
	main storage map 1-3
	mainline segments 4-80
	mainlines
KEY object code 4-31	detail calculations 4-4
ALL I DOJON OCCUPANT	detail output 4-80
	input fields 4-4
	input records 4-3
låst record	LR and overflow control 4-3
output discussion 4-5	open 4-3
control mainline 4-6	program close 4-4
control mainline code phase (\$RPPO) 2-15	total calculations 4-3
library of subroutines 4-40	total output 4-3
alternate collating sequence (\$\$PGDI) 4-44	match field save areas 4-76
array index (\$\$PGAA) 4-41	matching fields 4-18
BSCA logic for conversational (\$\$PGBC) 4-42	multiple files with 4-19
BSCA logic for receive (\$\$PGBG) 4-42	one file with 4-18
BSCA logic for transmit (\$\$PGBP) 4-43	matching records logic 4-16
BSCA logic for transmit and receive (\$\$PGBB) 4-42	multiple files with matching fields 4-19
command key indicator light restorer (\$\$PGLG) 4-47	multiple files with no match fields 4-18
command key indicator set routine (\$\$PGCB) 4-43	one file with match fields 4-18
convert to binary (\$\$PGBI) 4-42	methods of preserving decimal integrity 4-27
convert to decimal (\$\$PGBO) 4-43	MHHZO (move high-high zone) object code 4-32
DEBUG (\$\$PGDC) 4-44	MHLZO (move high-low zone) object code 4-32
divide (\$\$PGIC) 4-46	MLHZO (move low-high zone) object code 4-32
dump object tables (\$\$PGFO) 4-46	MLLZO (move low-low zone) object code 4-32
file translate (\$\$PGAB) 4-41	move array object code 4-33
load object tables (\$\$PGFO) 4-46	move field code phase, chain and read files (\$RPRW) 2-15
LOKUP (\$\$PGLC) 4-47	move high-high zone object code 4-32
move array (\$\$PGMA) 4-48	move high-low zone object code 4-32
multiply (\$\$PGMC) 4-48	move input flelds mainline 4-6
pack (\$\$PGCO) 4-44	move left object code 4-33
RPG II halt processor (\$\$SYRP) 4-49	move low-high zone object code 4-32
set resulting indicator (\$\$PGRI) 4-49	move low-low zone object code 4-32
square root (\$\$PGAC) 4-41	MOVE object code 4-32
test zone (\$\$PGTC) 4-49	move remainder (MVR) object code 4-34
unpack (\$\$PGCI) 4-43	MOVEA object code 4-33
library of subroutines code phase (\$RPRY) 2-19	MOVEL object code 4-33
line counter compression 3-18	MULT (multiply) object code 4-33
line counter compression phase (\$RPEC) 2-3	MULT subroutine (\$\$PGMC) 4-48
linkage between phases 1-3	multifile logic 4-6
linkage editor	with matching fields 4-19
entry control statement, build phase 2-20	with no matching fields 4-18
external symbol list, build phase 2-20	multiple files (see multifiles)
	multiply (MULT) object code 4-33
phase control statement, build phase 2-20	multiply subroutine (\$\$PGMC) 4-48
skip records, build phase 2-20	* * *
text, RLD, build phase 2-20	MVR object code 4-34
load object tables subroutine (\$\$PGFI) 4-45	
LOKUP object code 4-31	
LOKUP subroutine (\$\$PGLC) 4-47	
look-ahead field extraction code phase (\$RPPL) 2-12	name table 3-59
	numeric index, output fields addressing 4-5

object code-block 3-59 write an object block routine (DRGCZM) 2-25 object program 4-1 calculations object code 4-26 data areas 4-73 detailed object program flow 4-1 flowchart techniques 4-1 object code-block 3-59 editor 1 phase (\$RPSB) 2-19 editor 2 phase (\$RPSF) 2-19 fetch and transfer vector code phase (\$RPSG) fetch routine 4-84 fetch table 4-84 how to find an overlay 4-84	2-20
write an object block routine (DRGCZM) 2-25 object program 4-1 calculations object code 4-26 data areas 4-73 detailed object program flow 4-1 editor 1 phase (\$RPSB) 2-19 editor 2 phase (\$RPSF) 2-19 fetch and transfer vector code phase (\$RPSG) fetch routine 4-84 fetch table 4-84	2-20
object program 4-1 calculations object code 4-26 data areas 4-73 detailed object program flow 4-1 editor 2 phase (\$RPSF) 2-19 fetch and transfer vector code phase (\$RPSG) fetch routine 4-84 fetch table 4-84	2-20
calculations object code 4-26 fetch and transfer vector code phase (\$RPSG) data areas 4-73 fetch routine 4-84 detailed object program flow 4-1 fetch table 4-84	2-20
data areas 4-73 fetch routine 4-84 detailed object program flow 4-1 fetch table 4-84	2-20
detailed object program flow 4-1 fetch table 4-84	
flowchart techniques 4-1 how to find an overlay 4-84	
intermediate object program flow 4-3 phase description 2-19	
library of subroutines 4-40 priority 4-80	
overall flow 4-2 segments 4-79	
overlays 4-79 suboverlays 4-83	
object program flow 4-2 subsegments 4-80	
intermediate 4-3 technique 4-83	
overall 4-2 overlay (Model 15) 4-89	
object-time data areas (see data areas) category 4-89	
one file with match fields logic 4-18 overlay editor 1 phase (\$RPSB) 2-19	
OPCR (output processing control routine) 4-1 overlay editor 2 phase (\$RPSF) 2-19	
open a compression area routine (DRGCZA) 2-24 overlay fetch table 4-84	~ ~ ~ ~
open a compression block routine (DRGCZG) 2-25 overlay fetch and transfer vector code phase (\$RPSC	3) 2-20
open mainline overlay phases 2-19	
detailed object program flow 4-1 overlay priority 4-80	
intermediate object program flow 4-3 overlay segments 4-79	
overall object program flow 4-2 overlay subsegments 4-80	
output overlay technique 4-83 exception output 4-30 overview of the compiler 1-2	
(see also detailed output; total output)	
output buffers 4-77 output fields	
address of *PLACE 4-5 pack subroutine (\$\$PGCO) 4-44	
address of array with no index 4-5 PAGE 4-5	
address of array with numeric index 4-5 parameters	
address of array with variable index 4-5 [/O 3-16	
address of PAGE 4-5 to data management 4-81	
address of punch 4-5 to RPG II halt processor 3-16	
address of table elements 4-5 phase descriptions 2-1	
code phase 1 (\$RPPU) 2-16 phase linkage 1-3	
code phase 2 (\$RPPV) 2-16 phase load compression 3-60	
code phase 3 (\$RPPW) 2-16 PIOCS—disk control routine 2-24	
detailed object program flow 4-1 post compression and initialization phase (\$RPEW)	2-5
move optimization phase (\$RPLV) 2-13 post compression initialization phase (\$RPIC) 2-5	- 0
records code flow 4-1 pre-assemble calculations 1 phase (\$RBMB) 2-13	
output fields addressing 4-1 pre-assemble calculations 2 phase (\$RPMG) 2-13	
output fields and records code 4-1 pre-assemble calculations 3 phase (+RPLJ) 2-13	
output fields 1 code phase (\$RPPU) 2-16 pre-assemble calculations 4 phase (\$RPMH) 2-14	
output fields 2 code phase (\$RPPV) 2-16 pre-assemble indicator optimization phase (\$RPMI)	2-14
output fields 3 code phase (\$RPPW) 2-16 pre-assemble phases 2-13	
output fields move optimization phase (\$RPLV) 2-13 build segment list (\$RPMK) 2-14	
output-format compressions 3-44 constant, literal, edit word assign 1 (\$RPLG) 2	-13
output-format compression phase (\$RPEO) 2-4 constant, literal, edit word assign 2 (\$RPLR) 2	-13
output indicator optimization phase (\$RPLB) 2-13 DTF parameter assing (\$RPLG) 2-13	
output indicator testing phase (\$RPLZ) 2-14 indicator testing optimization (\$RPLZ) 2-14	
output mainline 4-2 ouptut fields move optimization (\$RPLV) 2-13	}
output object code block length phase (\$RPMA) 2-14 output indicator optimization (\$RPLB) 2-13	
output processing control routine (OPCR) 4-1 output object code block length (\$RPMA) 2-14	
output records 4-1 output segment list entries build (\$RPMM) 2-1-	
output records code 4-1 pre-assemble calculations 1 (\$RPMB) 2-13	
phase (\$RPPS) 2-15 pre-assemble calculations 2 (\$RPMG) 2-13	
· · · · · · · · · · · · · · · · · · ·	
output segment list entries build phase (\$RPMM) 2-14 pre-assemble calculations 3 (\$RPLJ) 2-13	
output segment list entries build phase (\$RPMM) 2-14 pre-assemble calculations 3 (\$RPLJ) 2-13 output table (see file input/output table) pre-assemble calculations 4 (\$RPMH) 2-14	
	-14
output table (see file input/output table) pre-assemble calculations 4 (\$RPMH) 2-14	-14

prime work area 4-73 print name table phase (\$RPGX) 2-6 print segment list phase (\$RPSD) 2-19 print symbol table phase (\$RPHD) 2-7 printer control routine (DRGCZP) 2-24 priority, overlay 4-80 process MFCM print specifications phase (\$RPGB) 2-6 program close mainline 4-80 program logic 2-1 publication organization iv publications, related iii punch 4-5	sort segment list phase (\$RPSN) 2-19 SQRT (square root) object code 4-35 square root subroutine (\$\$PGAC) 4-41 storage table, general 3-58 storage map of RPG II compiler 1-3 SUB (subtract) object code 4-35 suboverlays 4-83 subroutines (see library of subroutines) subsegments 4-80 subtract (SUB) object code 4-35 symbol table 3-60
READ object code 4-34	ta ble
read routine (see chain and read) record ID object code 4-6 record indicator 4-5 reference publications iii reformat segment list phase, Model 15 (\$RPRX) 2-21 related publications iii remote terminal/device support phase (\$RPGJ) 2-8 reserved object communications area (ROCA) 4-73 resolve EXTRN phase (\$RPSA) 2-19 RLABL object code 4-34 R-module output, generate (\$RPRZ) 2-21 ROCA 4-73 root segment 4-80 RPG II halt processor (\$SYRP) 4-49 parameters to 3-16 RPG II subroutines (see library of subroutines)	alternate collating sequence and translate 4-76 chain 3-54 compile-time symbol 3-54 compression block (CZATAB) 3-17 element address 4-5 file input/output (FIOT) 3-53 filename 3-57 indicator 4-74 internal symbol 3-58 name 3-59 overlay fetch 4-84 symbol 3-60 trailer 4-75 telecommunications 3-60 use of by calculations 4-26 table dump subroutine (\$\$PCFO) 4-46
calculations use of 4-26	table dump subroutine (\$\$PGFO) 4-46 table element address 4-5 table load subroutine (\$\$PGFI) 4-45 table load/table dump code phase (\$RPRC) 2-18 TAG object code 4-35
sample dump analysis 4-90 save areas control field 4-76 control routine 3-16 match field 4-76 secondary work area 4-75 segment list 3-57 build phase (\$RPMP) 2-15	tape (see magnetic tape) technique, overlay 4-83 terminal errors (see errors) telecommunications 3-18 compression 3-50 compression phase (\$RPEE) 2-3 table 3-60 (see also BSCA)
completion phase (\$RPMQ) 2-15 final segment list 3-57 initialize phase (Model 6, 10) (\$RPRX) 2-19 reformat phase, Model 15 (\$RPRX) 2-21	telecommunications compression phase (\$RPEE) 2-3 telecommunications table 3-60 test zone subroutine (\$\$PGTC) 4-49 TESTB (test bits) object code 4-35
segments mainline 4-80 overlay 4-79 root 4-80	TESTZ (test zone) object code 4-36 text-RLD records 3-61 TIME operation code 4-36 total calculations
subsegments 4-80 set lower limits object code 4-34 SET object code 4-34 SETLL operation code 4-34	intermediate object program flow 4-3 overall object program flow 4-2 (see also calculations) total output
SETOF (set indicator off) object code 4-35 SETON (set indicator on) object code 4-35 set resulting indicators subroutine (\$\$PGR1) 4-49 sort object code phase	intermediate object program flow 4-3 overall object program flow 4-2 (see also output) trailer table 4-75
Model 6, 10 (\$RPSE) 2-19 Model 15 (\$RPRY) 2-21	transfer vectors code-phase (\$RPSG) 2-20 translate table 4-76

UDATE 4-73 UDAY 4-73 UMONTH 4-73 unpack subroutine (\$\$PGCI) 4-43 update files move 4-5 UYEAR 4-73

variable index, addressing with calculations 4-27 with output fields 4-1 vectors, transfer 2-20

work areas

compression work area 3-17
prime work area 4-73
secondary work area 4-75
write a compression routine (DRGCZC) 2-24
write an object code block routine (DRGCZM) 2-25

XFOOT (crossfoot) object code 4-36 XR1, use by calculations 4-26

Z-ADD (zero and add) object code 4-36 Z-SUB (zero and subtract) object code 4-36

1P output 4-5



$^{\prime}$ 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

© IBM Corp. 1970, 1971, 1972, 1973, 1974, 1975

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

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