Y26-3704-0

Program Logic

IBM System/360 Operating System
Report Program Generator

Program Number 360S-RG-038

This publication describes the internal logic of the Report Program Generator for System/360 Operating System. It is intended for use by persons involved in program maintenance and by system programmers who are altering the program design.

Restricted Distribution

PREFACE

This program logic manual (PLM) supplements the program listing of the Operating System Report Program Generator compiler (referred to in this publication as RPG) by describing the program.

The first section of this PLM discusses the overall structure of the RPG compiler. An overall flowchart is presented with a storage allocation map and a table illustrating the input/output organization.

A section of the manual is devoted to each phase of the RPG compiler. Included for each phase are a flowchart of the logical elements, a storage allocation map and an input/output flowchart. The text includes a summary of each routine and subroutine (identified by the symbolic label in the program listing) and the programmed switches used by the phase.

The last two sections of the manual are appendixes containing tables and record formats and a glossary of the terms used in this publication.

The program listing and its supplementary comments supply detailed information about the functions of the logical elements.

PREREQUISITES AND RELATED LITERATURE

Effective use of this manual requires an understanding of the Operating System Report Program Generator language contained in the publication IBM System/360 Operating System, Report Program Generator, C24-3337.

For information on the Operating System that is beyond the purpose of this publication, refer to the following publications:

IBM System/360 Operating System, Assembler (E) Programmer's Guide, C28-6595

IBM System/360 Operating System Data Management, C28-6537

IBM System/360 Operating System, Concepts and Facilities, C28-6535

IBM/360 Operating System, System Programmer's Guide, C28-6550

IBM System/360 Operating System, Linkage Editor, C28-6538

IBM System/360 Operating System, Job Control Language, C28-6539

IBM System/360 Operating System, Messages, Completion Codes and Storage Dumps, C28-6631

For titles and abstracts of associated publications, see the IBM System/360 Bibliography (A22-6822).

First Edition

Specifications contained herein are subject to change from time to time. Any such change will be reported in subsequent revisions or Technical Newsletters.

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

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Department 425, Rochester, Minnesota 55901.

ILLUSTRATIONS

Figures

Tables

Charts

IBM System/360 Operating System consists of a control program and a number of processing programs (Figure 1). The control program governs the order in which the processing programs are executed and provides services that are required in common by the processing programs during their execution. The processing programs consist of language translators and service programs that are provided by IBM to assist the user of the system, as well as problem programs that are written by the user and incorporated as part of the system. Operating System Report Program Generator (RPG) converts a System/360 RPG language program directly to relocatable System/360 machine language instructions. The System/360 RPG language provides the programmer with an efficient technique for writing source programs that can be translated into object modules (machine language) by the System/360 RPG program.

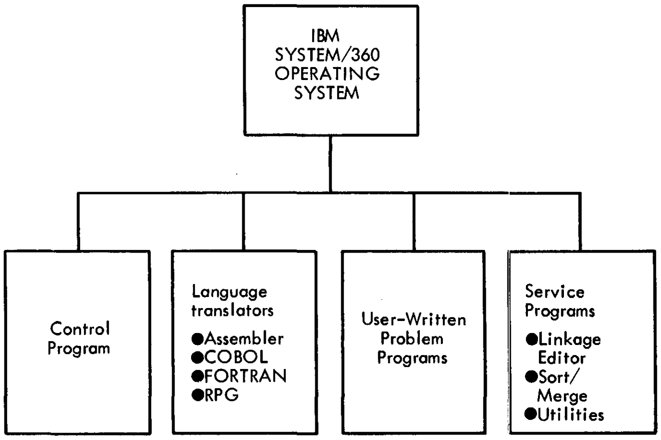


Figure 1. General Organization of the Operating System

The Operating System RPG consists of a source language and a compiler. The source language provides for specification of input and output data sets, the component fields of input data records, the literals, the operations and calculations to be performed, and the fields of the output records. The RPG compiler translates into an object module the RPG language entries specified on the RPG coding forms.

The RPG compiler is a fast, efficient program. The input/output operations are reduced by retaining as much source data as possible in core storage. To accomplish this objective, all identifying information is deleted from the source specifications and the resulting compression is placed in a reserved area of core storage.

The number of data passes is reduced by placing field names, literals, and resulting indicators into tables as they are encountered. The areas allotted to the tables are large enough to contain all of the entries in most of the programs to be compiled. As a result, addresses can be assigned to the entries immediately and machine instructions can be generated.

SYSTEM ENVIRONMENT

The RPG compiler program operates under supervision of the Operating System Control Program. The minimum System/360 and I/O requirements are

1. Minimum requirements of the Operating System Control Program.
2. Three work data sets which can be DASD, magnetic tape, or mixed. These (SYSUT1, SYSUT2, SYSUT3 (BASAM)) utility data sets are used for external storage, the use and formats of these data sets will vary from one phase to another.
3. Fixed, unblocked SYSIN data set from card reader, disk or magnetic tape.
4. Fixed, unblocked SYSPRINT data set to printer, disk or magnetic tape.
5. Either a fixed, unblocked SYSPUNCH (card punch, disk or magnetic tape) or a SYSGO (magnetic tape or disk).
6. Additional data sets as required by the object program.
7. No fewer than 15,360 bytes of core storage available for RPG use. The size of the control program must be added to this figure to obtain total core storage required.

NOTE: Standard labels are required on disk-resident data sets; standard labels or no labels may be used with magnetic tape resident data sets.

USE OF ADDITIONAL FEATURES

Additional core storage will be used if available. The field name table, literal table and the compression area will be expanded.

PROGRAM ORGANIZATION

MAJOR COMPONENTS

The RPG compiler consists of the following major components:

Compiler Input/Output Executor (CIOEX)
Prephase
Resident Phase
Enter Phases
Intermediate Phase
Assign Phases
Assemble Phases
I/O Phases
Diagnostic Phases
Linkage Phase
Terminal Phase

These major functional modules enable the RPG compiler to read the source program and generate machine language instructions. Figure 2, Table 1, and Chart AA illustrate the organization and operation of the compiler.

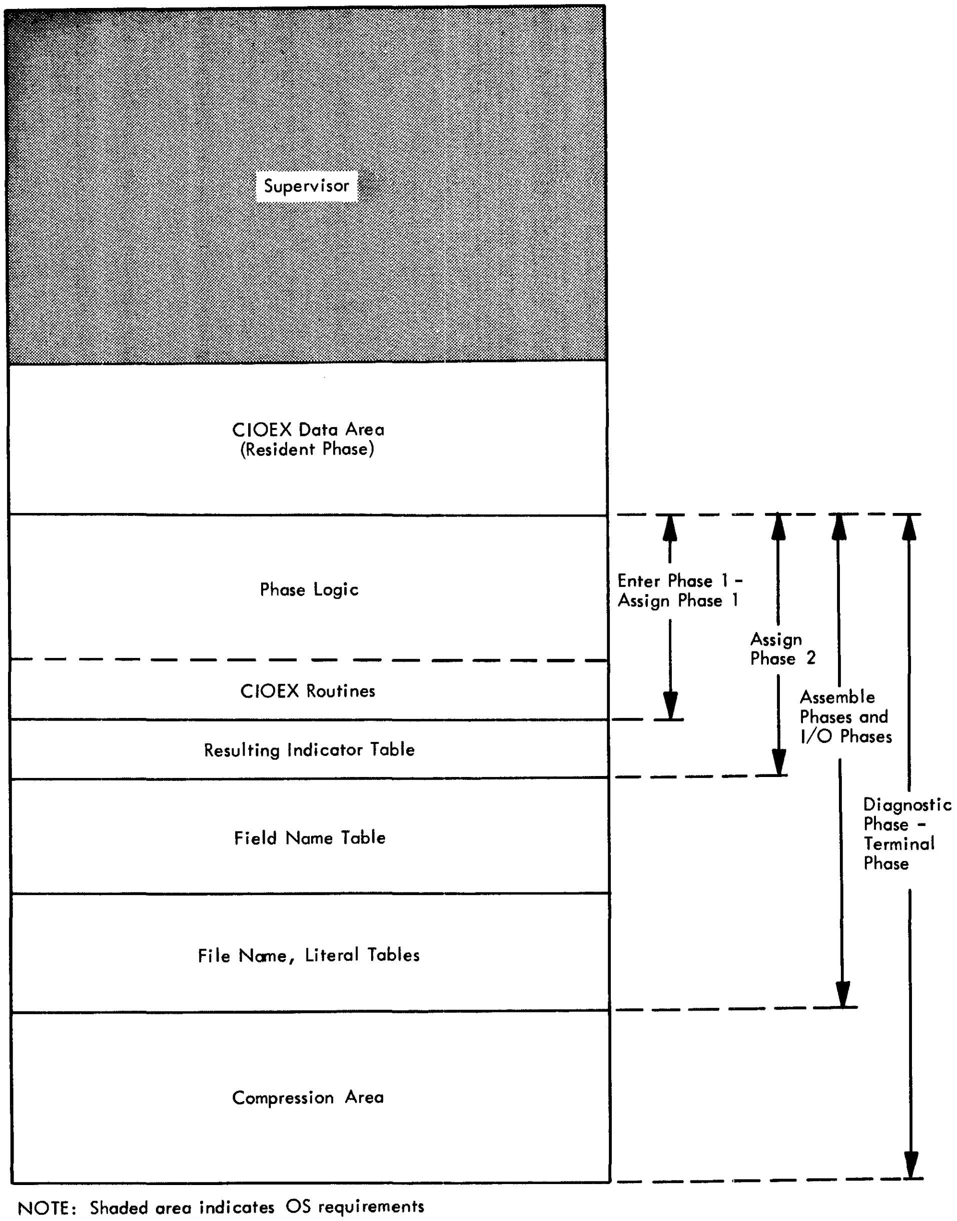


Figure 2. Overall RPG Storage Allocation Map

Table 1. Input/Output File Organization by Phase

	SYSTEM RESIDENCE VOLUME	SYSIN	SYSPRINT	SYSGO/ SYSPUNCH	SYSUT1	SYSUT2	SYSUT3
RESIDENT PHASE	Input-Compiler Program Supplies the Program Logic of Each Phase of the Compiler When Called	▨	▨	▨	▨	▨	▨
PREPHASE		OPEN-Control Card	OPEN	OPEN SYSGO	OPEN	OPEN	OPEN
ENTER PHASE 1		File Description Specification	Output-Print Specifications and Appropriate Diagnostic Codes During Each of These Phases	▨	Output-Appropriate Specification Compression Records	▨	▨
ENTER PHASE 2		File Extension and Line Counter Specifications		▨		▨	▨
ENTER PHASE 3		Input Specifications		▨		▨	▨
ENTER PHASE 4		Calculation Specifications		▨	▨	▨	Output-Preprocessed Specifications
ENTER PHASE 5		▨		▨	Output-Appropriate Specification Compression Records	▨	Input-Preprocessed Specifications
ENTER PHASE 6		Output-Format Specifications		▨		▨	▨
INTERMEDIATE PHASE		CLOSE *	▨	OPEN* SYSPUNCH	▨	▨	▨
ASSIGN PHASE 1		▨	Print Resulting Indicator, Field Name and Literal Tables and Diagnostic Codes	Output-ESD and TEXT Cards During These Phases (Object Module)	Input/Output-Compression Records	▨	Output-RLD
ASSIGN PHASE 2		▨				▨	
ASSEMBLE PHASE 1		▨	▨		Input-Compression Records During These Phases	▨	
ASSEMBLE PHASE 2		▨	Print Diagnostic Codes			Temporary Storage and Output-Blank After Entries	▨
ASSEMBLE PHASE 2.5		▨	▨			Temporary Storage and Input/Output-Blank After Entries	Output-RLD
ASSEMBLE PHASE 3		▨	Print Diagnostic Codes			Output-Blank After Entries	
ASSEMBLE PHASE 4		▨					
ASSEMBLE PHASE 4.5		▨				Input-Blank After Entries	
I/O PHASE 1		▨	▨			Output/Input Temporary Storage	
I/O PHASE 2		▨	▨		▨		▨
DIAGNOSTIC PHASES 1-5		▨	Print Diagnostic Messages	▨	▨	▨	▨
LINKAGE PHASE		▨	Print Memory Map	Output-RLD, and TEXT Cards	▨	▨	Input-RLD
TERMINAL PHASE		CLOSE **	CLOSE	CLOSE **	CLOSE	CLOSE	CLOSE

Notes: Shading indicates data sets not referenced in a particular phase.
* SYSIN is closed and SYSPUNCH is opened if the compiler option DECK, LOAD is specified.
** SYSIN, SYSGO and SYSPUNCH are closed according to the compiler option specified (see Terminal Phase).

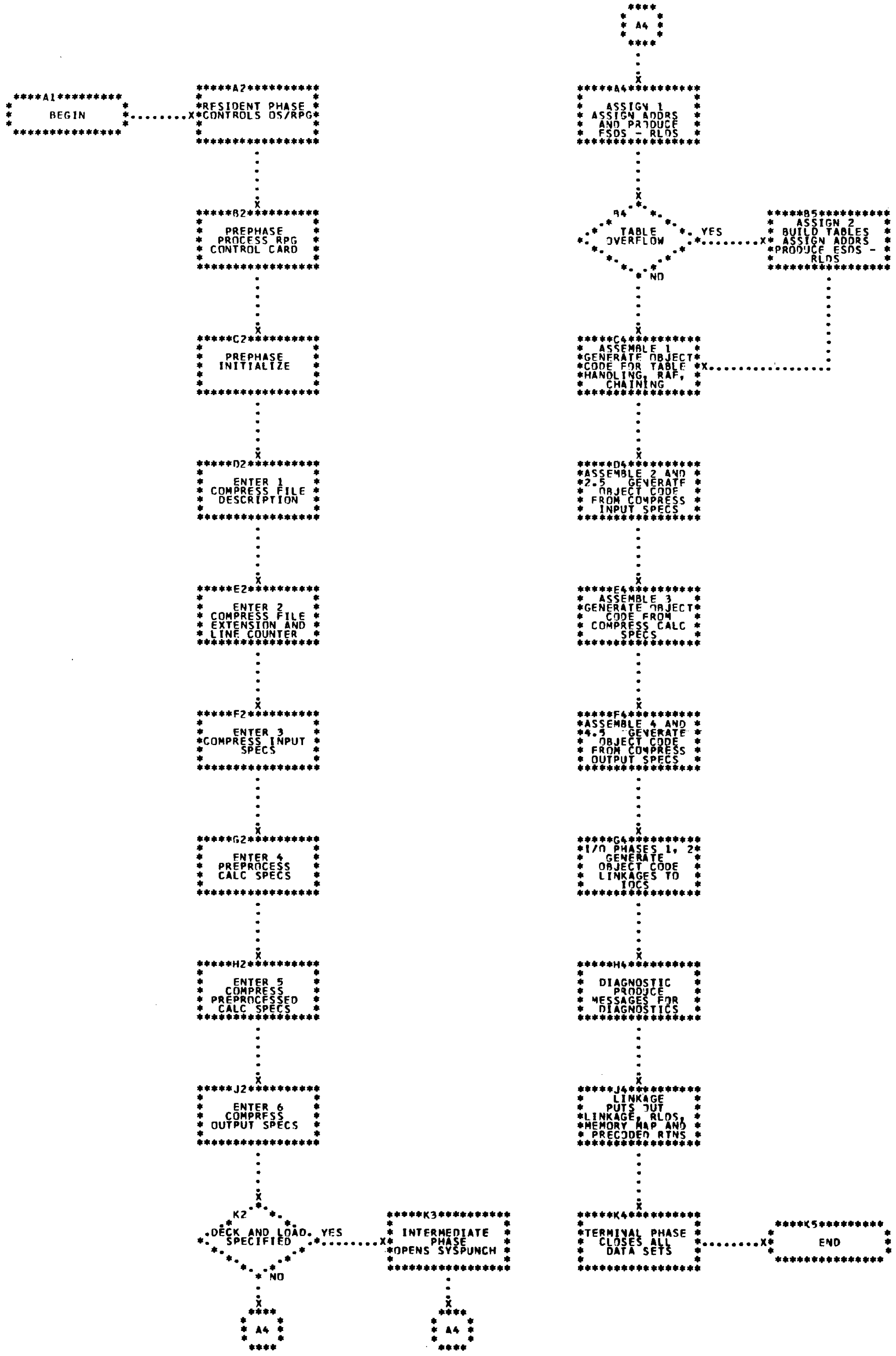

A1
BEGIN
A2
RESIDENT PHASE CONTROLS OS/RPG
B2
PREPHASE PROCESS RPG CONTROL CARD
C2
PREPHASE INITIALIZE
D2
ENTER 1 COMPRESS FILE DESCRIPTION
E2
ENTER 2 COMPRESS FILE EXTENSION AND LINE COUNTER
F2
ENTER 3 COMPRESS INPUT SPECS
G2
ENTER 4 PREPROCESS CALC SPECS
H2
ENTER 5 COMPRESS PREPROCESSED CALC SPECS
J2
ENTER 6 COMPRESS OUTPUT SPECS
K2
DECK AND LOAD SPECIFIED
YES
NO
K3
INTERMEDIATE PHASE OPENS SYSPUNCH
A4
A4
A4
A4
ASSIGN 1 ASSIGN ADDRS AND PRODUCE ESDS - RLDS
B4
TABLE OVERFLOW
YES
NO
B5
ASSIGN 2 BUILD TABLES ASSIGN ADDRS PRODUCE ESDS - RLDS
C4
ASSEMBLE 1 GENERATE OBJECT CODE FOR TABLE HANDLING, RAF, CHAINING
D4
ASSEMBLE 2 AND 2.5 GENERATE OBJECT CODE FROM COMPRESS INPUT SPECS
E4
ASSEMBLE 3 GENERATE OBJECT CODE FROM COMPRESS CALC SPECS
F4
ASSEMBLE 4 AND 4.5 GENERATE OBJECT CODE FROM COMPRESS OUTPUT SPECS
G4
I/O PHASES 1, 2 GENERATE OBJECT CODE LINKAGES TO IOCS
H4
DIAGNOSTIC PRODUCE MESSAGES FOR DIAGNOSTICS
J4
LINKAGE PUTS OUT LINKAGE, RLDS, MEMORY MAP AND PRECODED RTNS
K4
TERMINAL PHASE CLOSES ALL DATA SETS
K5
END

Chart AA. RPG Compiler Program

FUNCTIONS OF THE MAJOR COMPONENTS

Compiler Input/Output Executor

The Compiler Input/Output Executor (CIOEX) routines are loaded with each phase of the RPG compiler. Only those routines required for a particular phase are loaded. The CIOEX routines perform the input/output processing for the compiler. A CIOEX data area which resides in core for all phases is in the resident phase. It contains constants and key addresses. The functions of CIOEX are

1. Handle all linkages to the Operating System (i.e., calls to read source program specifications from the SYSIN and write object module text on SYSGO/SYSPUNCH)
2. Maintain the addresses of data areas (i.e., beginning of file description, input, calculation, and output-format specifications in the compression area)
3. Maintain key information to be used by the other phases, such as sterling code information
4. Maintain an address counter for the machine language code being generated
5. Contain routines which are common to the RPG compiler phases

Resident Phase

There is a small routine in this phase that receives control from the operating system and transfers control to Prephase. At the end of the job this routine returns control to the operating system. This phase also contains the CIOEX data area and the DCBs for SYSIN, SYSPRINT, SYSPUNCH or SYSGO, SYSUT1, SYSUT2 and SYSUT3.

Prephase

Prephase performs the initial operations necessary for the other phases of the RPG compiler. The tasks performed by Prephase are

1. Open all data sets needed by the RPG compiler
2. Determine the amount of available core for compiler tables
3. Initialize the compiler tables
4. Process and diagnose the RPG control card

Enter Phases

There are six enter phases that diagnose and compress the source program deck. Their functions are

1. Read, diagnose, and list the source statements
2. Compress the source statements
3. Build the following:
   a. File name table
   b. Resulting indicator table
   c. Field name table
   d. Literal table

While executing the enter phases, the field name table or literal table may exceed allotted core. If this occurs, no more entries are made; however, the other functions of the phases are continued.

The enter phases place the compressed source statements (specifications) in the compression area. When this area becomes filled, it is written on work data set 1 (SYSUT1).

Intermediate Phase

This phase closes the SYSIN DCB and sets up and opens the DCB for SYSPUNCH. Entry to this phase occurs if both the DECK and LOAD options have been specified by the user.

Assign Phases

There are two assign phases. Under normal conditions, only Assign Phase 1 is executed. Assign Phase 2 is executed only if the field name or literal tables generated in the enter phases exceed the allotted core. Assign Phase 2 duplicates the table building function of the enter phases and the address assigning function of Assign Phase 1. The functions of these phases are

1. Perform further diagnostics of the specifications, i.e., multidefined, undefined, and unreferenced field names and resulting indicators
2. Compute and assign addresses to the fields, literals, and resulting indicators contained in the tables
3. Replace the symbolic name in the compression area with the assigned addresses
4. Put out text card images for the resulting indicators, fields and literals
5. If any resulting indicators is an ENTRY type, an ESD (external symbol dictionary)

card image is output for that indicator
6. Put out ESD card images and RLD (relocation list dictionary) entries for EXTRN and ENTRY type field names. The ESDs are put out on SYSPUNCH or SYSGO and the RLD entries are put out on work data set 3 (SYSUT3).

Assemble Phases

Six assemble phases generate the RPG object program from the compressed specifications which were formed during the preceding enter and assign phases.

I/O Phases

The function of the two I/O phases is to generate and put out object code linkages to the Operating System.

Diagnostic Phases

The diagnostic phases put out a list of diagnostic messages which explain all diagnostic errors that have occurred during the compilation.

Linkage Phase

The linkage phase puts out the linkage program, prints the memory map and generates RLD card images. This phase also puts out precoded routines for the object program.

Terminal Phase

The terminal phase closes all compiler data sets.

PHASE NAMES

The following is a list of names used to identify the various phases. The name of the next phase to be executed is stored at PHSNAM in the CIOEX data area as RPG10xxx where xxx is the last three digits of component name.

PHASE	COMPONENT	
NAME	NAME	IDENTITY
RESIDENT PHASE	IESRPG	•038
PREPHASE	IES000	< 038
CIOEX	IES009	(038
ENTER PHASE 1	IES030	&038
CIOEX	IES039	1038
ENTER PHASE 2	IES040	$038
CIOEX	IES049	*038
ENTER PHASE 3	IES050	(038
CIOEX	IES059	;038
ENTER PHASE 4	IES060	¬038
CIOEX	IES069	- 038
ENTER PHASE 5	IES070	/038
CIOEX	IES079	,038
ENTER PHASE 6	IES080	%038
CIOEX	IES089	> 038
INTERMEDIATE PHASE	IES08A	?038
ASSIGN PHASE 1	IES090	·038
CIOEX	IES099	#038
ASSIGN PHASE 2	IES100	@038
CIOEX	IES109	'038
ASSEMBLE PHASE 1	IES110	=038
CIOEX	IES119	"038
ASSEMBLE PHASE 2	IES120	A038
CIOEX	IES129	B038
ASSEMBLE PHASE 2.5	IES130	C038
CIOEX	IES139	D038
ASSEMBLE PHASE 3	IES140	E038
CIOEX	IES149	F038
ASSEMBLE PHASE 4	IES150	G038
CIOEX	IES159	H038
ASSEMBLE PHASE 4.5	IES160	I038
CIOEX	IES169	J038
I/O PHASE 1	IES170	K038
CIOEX	IES179	L038
I/O PHASE 2	IES180	M038
CIOEX	IES189	N038
DIAGNOSTIC PHASE 1	IES190	Q038
CIOEX	IES199	R038
DIAGNOSTIC PHASE 2	IES200	S038
CIOEX	IES209	T038
DIAGNOSTIC PHASE 3	IES210	U038
CIOEX	IES219	V038
DIAGNOSTIC PHASE 4	IES220	W038
CIOEX	IES229	X038
DIAGNOSTIC PHASE 5	IES230	Y038
CIOEX	IES239	Z038
LINKAGE PHASE	IES240	0038
STERLING CONVERSION INPUT	IES241	1038
STERLING CONVERSION OUTPUT	IES242	2038
TEST ZONE AND DECIMAL	IES243	3038
TEST ZONE (BCD)	IES244	4038
SIGN CHECK	IES245	5038
TABLE LOOK-UP	IES246	6038
SET INDICATOR	IES247	7038
CIOEX	IES249	8038
TERMINAL PHASE	IES250	9038

INTRODUCTION

RPG Compiler Input/Output Executor (CIOEX) is loaded with each phase of the RPG. CIOEX consists of a method (CIOEX Driver) of interpreting an input/output request and the necessary routines to perform the required I/O for a particular phase. All CIOEX routines are read-only and use the CIOEX data area to hold constants, key addresses, switches, status information, and buffers for SYSIN, SYSPRINT and SYSPUNCH or SYSGO. The CIOEX data area resides in Resident Phase.

The CIOEX routines required for each specific phase are loaded rather than loading the entire library with each phase. Therefore, within the object module each phase consists of its own logic together with the CIOEX driver and the applicable CIOEX routines.

Figure 3 and Chart AB illustrate the organization and operation of CIOEX.

LOGIC

GR7 is the linkage register between CIOEX and other phases and contains the starting address of the CIOEX driver (address calculating instructions and table of displacements to CIOEX routines).

The calling routine loads the desired CIOEX routine number (Table 2) into GR2, stores its return address in GR5, and branches to GR7.

The CIOEX driver uses the values in GR2 to develop the branch address of the proper CIOEX routine.

Upon completion of the CIOEX routine a return to the main program occurs by way of GR5.

All calling routines use one of the two types of linkage.

SWITCHES AND INDICATORS

CXINVPRT

X'F0' Inverted print

CXCOLSEQ

X'F0' alternate collating sequence

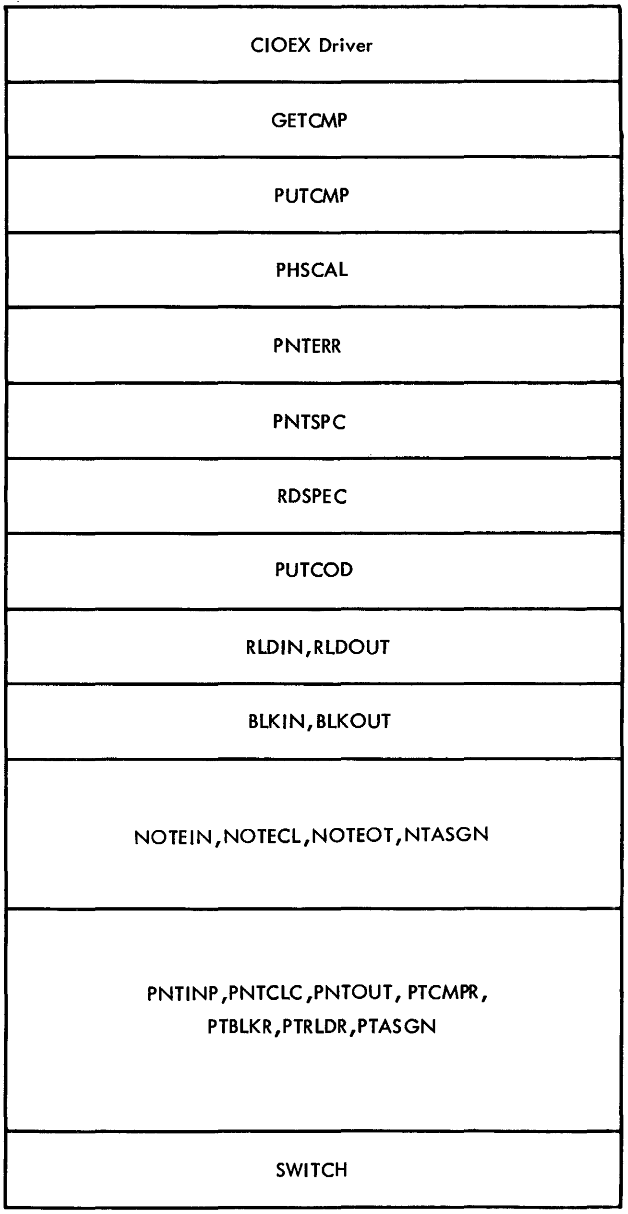
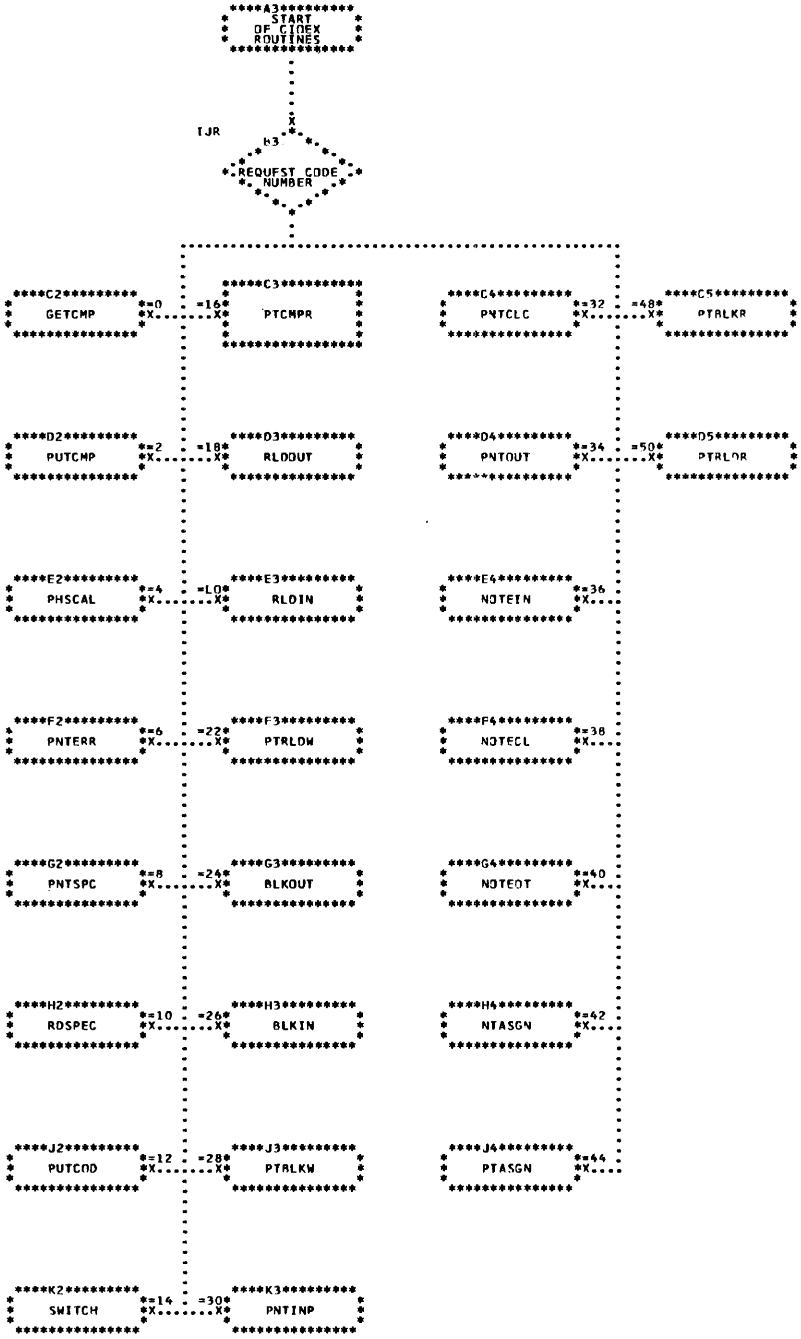


Figure 3. CIOEX Storage Allocation Map



NOTE, ALL ROUTINES RETURN DIRECTLY VIA REG 7

Chart AB. CIOEX

Table 2. Linkages to Resident Routines

Routine Name	Routine Number	Type of Linkage*
GETCMP	0	A
PUTCMP	2	A
PHSCAL	4	A
PNTERR	6	A
PNTSPC	8	A
RDSPEC	10	A
PUTCOD	12	A
SWITCH	14	A
PTCMPR	16	A
RLDOUT	18	B
RLDIN	20	B
PTRLDW	22	A
BLKOUT	24	B
BLKIN	26	B
PTBLKW	28	A
PNTINP	30	A
PNTCLC	32	A
PNTOUT	34	A
NOTEIN	36	A
NOTECL	38	A
NOTEOT	40	A
NTASGN	42	A
PTASGN	44	A
PTBLKR	48	A
PTRLDR	50	A

*Linkage in calling routine.

Type A	Type B
LA 2, Routine number	LA 1, Address of I/O area
BALR 5, 7	LA 2, Routine number
	BALR 5, 7

CXSTRLNG

00xxxxxx	Shillings column blank/input
01xxxxxx	IBM shillings for input
10xxxxxx	BSI shillings for input
xx00xxxx	Pence column blank/input
xx01xxxx	IBM pence for input
xx10xxxx	BSI pence for input
xxxx00xx	Shillings column blank/output
xxxx01xx	IBM shillings for output
xxxx10xx	BSI shillings for output
xxxxxx00	Pence column blank/output
xxxxxx01	IBM pence for output
xxxxxx10	BSI pence for output

CXCMPOPT

Compiler option byte

Not used	
xxxxx0xx	LIST
xxxxx1xx	NOLIST
xxxxxx00	LOAD, NODECK
xxxxxx01	DECK, NOLOAD
xxxxxx10	DECK, LOAD
xxxxxx11	NOLOAD, NODECK

MAIN ROUTINES

GETCMP

- Reads the next block of compression from a work data set
- Updates the count of the block being processed

PUTCMP

- Writes a block of compression on a work data set
- Keeps a total count of the number of blocks during the enter phases
- Links to the table initialization routine

PHSCAL

- Provides linkage to the Operating System for loading of the next phase
- Loads key registers
- Passes control to the phase

PNTERR

- Turns on the appropriate note switch, using the unpacked numbers in positions 97-99 of the print area
- Moves IESnnnI into the print area (where nnn is the note number)
- Passes control to PNTSPC

PNTSPC

- Prints the information in the print area
- Initiates a skip to channel 1 when a full page has been printed
- Prints a new heading line

RDSPEC

- Reads a record from the system input device

- Provides the address of this record to the calling phase (register 4)

PUTCOD

- Writes 80 bytes of information on SYSPUNCH/SYSGO
- Register 4 points to the first byte
- If NOLOAD and NODECK options have been selected, this routine is not executed, but returns to the calling routine

SWITCH

- Repositions work data set number 1 at the origin point

PTCMPR

- Positions compression for reading, at the origin, on the next GETCMP request

RLDOUT

- Writes RLD entries on work data set 3 as 132-character records
- Uses register 1 as a pointer

RLDIN

- Reads RLD entries from work data set 3
- Reads the 132-byte record into the area pointed to by register 1

PTRLDW

- Conditions the RLD to be written at the beginning of the data set

PTRLDR

- Conditions the RLD to be read at the beginning of the data set.

BLKOUT

- Puts out a blank-after entry onto work data set number 2
- Uses register 1 as a pointer to the output area

BLKIN

- Reads a blank-after entry into the area pointed to by register 1

PTBLKW

- Repositions, for writing at the origin, the work data set containing the blank-after entries

PTBLKR

- Repositions, for reading at the origin, the work data set containing the blank-after entries

PNTINP

- Conditions compression to read the block containing the first input specification compression

PNTCLC

- Conditions compression to read the block containing the first calculation specification compression

PNTOUT

- Conditions compression to read the block containing the first output-format specification compression

NOTEIN

- Notes the block containing the first input specification compression

NOTECL

- Notes the block containing the first calculation specification compression

NOTEOT

- Notes block containing the first output-format specification compression

NTASGN

- Notes the compression block currently being processed

PTASGN

- Conditions the compression to read the block referenced by the previous NTASGN

CONSTANTS ALWAYS IN CIOEX DATA AREA OF CORE STORAGE

- At the beginning of each phase the address of this data area is in GR6 and GR12
- GR6 must contain the address of the data area at all times when CIOEX routines have control
- GR12 is loaded to provide compatability betweeen phases coded prior to the redesigning of the interface into reenterable routines (CIOEX)

Name	Length	Purpose
CXDUBLWD	8	Used for work area
CXPHSNAM	8	Name of next phase to be loaded
CXDCBSRC	4	Address of DCB for input data set
CXDCBLST	4	Address of DCB for print data set
CXDCBPGO	4	Address of DCB for PUNCH/GO data set
CXDCBWK1	4	Address of DCB for SYSUT1
CXDCBWK2	4	Address of DCB for SYSUT2
CXDCBWK3	4	Address of DCB for SYSUT3
CXRIADDR	4	Address of Resulting Indicator Table
CXFLDADD	4	Address of Field Name Table
CXLITADD	4	Address of Literal Table
CXADDCNT	4	Counter of core used for object program
CXCMPADD	4	Address of compression area
CXCMPEXT	4	Length of compression area
CXHLDREG	4	Hold area for return register
CXNOTES	32	Diagnostic note array
CXFLDLEN	4	Length of Field Name Table
CXLITLEN	4	Length of Literal Table
CXCALCNT	4	Record count for locating calculation specification compressions
CXOUTCNT	4	Record count for locating output-format specification compressions
CXINPCNT	4	Record count for locating input specification compressions
CXNOTNUM	4	Record count for locating field names in compression
CXEXTCNT	2	External counter (ESD)
CXNUMCMP	2	Number of compression blocks
CXNUMBLK	2	Number of blank-after entries
CXNUMRLD	2	Number of RLD records
CXBLKNUM	2	Compression block being processed
CXIDSPLC	2	Displacement to first input specification compression
CXCDSPLC	2	Displacement to first calculation specification compression
CXODSPLC	2	Displacement to first output-format specification compression
CXANSPLC	2	Displacement to locate field names in compression
CXHDGSW	1	User page number switch
CXRETCDE	1	Return code
CXIPTEOF	2	Linkage to EOF routine
CXCMPOPT	1	Compiler option byte
CXSTRLNG	1	Sterling code switches
CXCOLSEQ	1	Alternate collating sequence
CXINVPRT	1	Inverted print switch
CXONEPCK	1	Number 1 in packed format
CXTRKBF	1	Track blocking factor for DASD
CXLITOVF	4	Literal overflow
CXFLDOVF	4	Field name overflow
CXCMPOVF	2	Compression overflow Byte 1 X'F0' Compression is on disk X'00' Compression is on tape Byte 2 X'F0' Compression overflow X'00' No compression overflow
CXSEQINP	4	Sequence number of first input specification
CXSEQCAL	4	Sequence number of first calculation specification
CXSEQOUT	4	Sequence number of first output-format specification
CXLINBCT	2	Line counter branch and count
CXLINCNT	2	Line counter total value
CXPAGNUM	4	Page number count
CXPUNCNT	4	Sequence count for punching
CXSYNERR	4	Linkage to synchronous error routine
CXPGRNAM	6	Program name ('RPGOBJ')
CXPASKEY	8	Temporary storage between phases
CXHEADIN	17	Header and version identification
CXJOBNAM	8	Job name from JOB
CXSTEPNA	8	Step name from EXEC
CXTODAY	8	Today's date
CXSYSPDD	8	ddname for SYSPUNCH
RDRPCHCD	1	Control character for stacker select
RDPRCHAR	80	Reader/Punch buffer 1
PNTRARAC	1	Control character for carriage control
PNTRAREA	120	Printer area
CXSAVEAR	72	Save area
CXSPIESV	4	Address of PICA from previous SPIE

ALLOCATION OF GENERAL REGISTERS FOR ALL PHASES

0	Volatile*
1	Volatile*
2	Volatile*
3	Compression area
4	Read in/punch out
5	Return register
6	CIOEX data area base register
7	CIOEX base register
8	Phase base register
9	Work register or phase base register
10	Print area
11	Work register
12	Work register
13	Work register
14	Volatile*
15	Volatile*

*These registers are used by CIOEX and/or OS. The contents will be destroyed.

INTRODUCTION

Input/output areas are required for the operation of the System/360 Operating System. Resident Phase provides the CIOEX data area (see Compiler Input/Output Executor) and DCBs for SYSIN, SYSPRINT, SYSGO/SYSPUNCH, SYSUT1, SYSUT2, and SYSUT3. These areas and a small routine for communications between RPG and OS remain in core throughout the execution of the RPG compiler.

The communication routine

1. Receives control for the supervisor
2. Links to the next phase specified in the CIOEX data area
3. Accepts the return from Terminal Phase
4. Returns control to the supervisor

Resident Phase also issues the SPIE macro-instruction and contains the SPIE routine for the RPG processor.

Figure 4 and Chart AM illustrate the organization and operation of Resident Phase. Figure 5 illustrates the input/output flow for the phase.

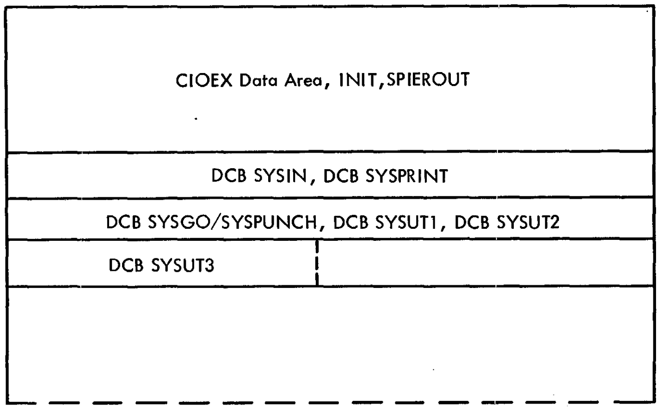

CIOEX Data Area, INIT,SPIEROUT
DCB SYSIN, DCB SYSPRINT
DCB SYSGO/SYSPUNCH, DCB SYSUT1, DCB SYSUT2
DCB SYSUT3

Figure 4. Resident Phase Storage Allocation Map

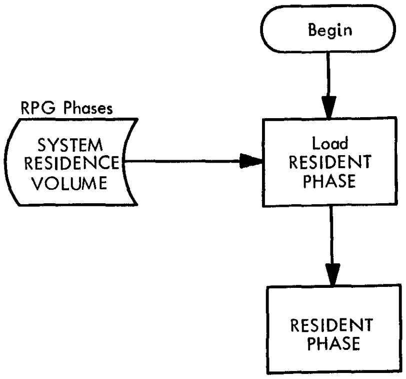


Figure 5. Resident Phase Input/Output Flow

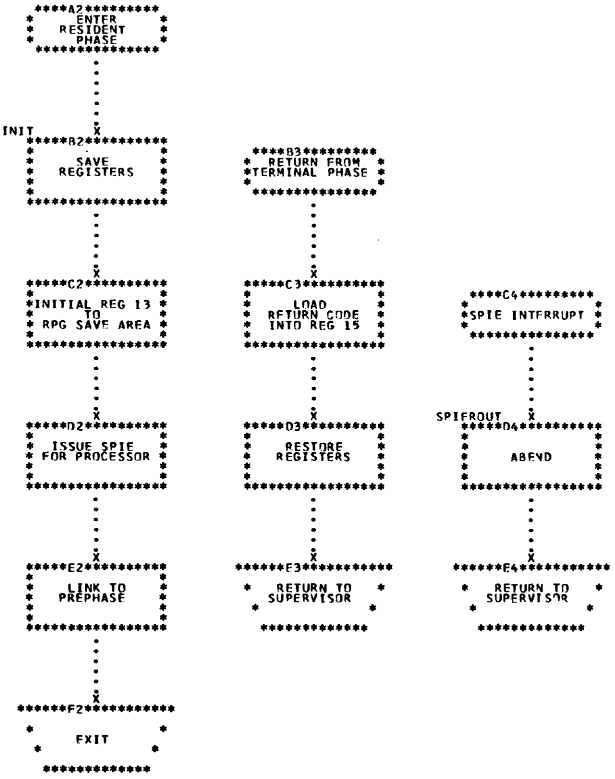


Chart AM. Resident Phase

PREPHASE

INTRODUCTION

The RPG Prephase is loaded as the first phase of the RPG processor. Prephase opens all files needed by RPG, diagnoses the RPG control card, determines the amount of available core for compiler tables, and initializes the literal table, field name table, and compression area.

Figure 6 and Chart BA illustrate the organization and operation of Prephase.

Figure 7 illustrates the input/output flow for Prephase.

LOGIC

The compiler options specified by the user affect the assignment of the DCB reserved for SYSGO/SYSPUNCH. If LOAD, NODECK is specified, the DCB is assigned to SYSGO. If NOLOAD, DECK is specified, the DCB is assigned to SYSPUNCH. If LOAD, DECK is specified, the DCB is assigned to SYSGO and SYSPUNCH overlays the SYSIN DCB in Intermediate Phase. CXCMPOPT is set accordingly.

SYSIN, SYSGO/SYSPUNCH, SYSPRINT, SYSUT1, SYSUT2, and SYSUT3 are opened.

The available core is divided between the field name, literal and resulting indicator tables and the compression area. The tables and area are initialized by being filled with X'FE'. A card is read and checked to see if it is a control card, valid specification, or other type (treated as comment card). The respective actions are: perform control card diagnostics, indicate an error (no control card), or read another card.

The switches for the RPG control card are shown in Table 3.

MAIN ROUTINES

PREPHASE

- Picks up compiler options, ddnames and heading information passed by job scheduler or invoker

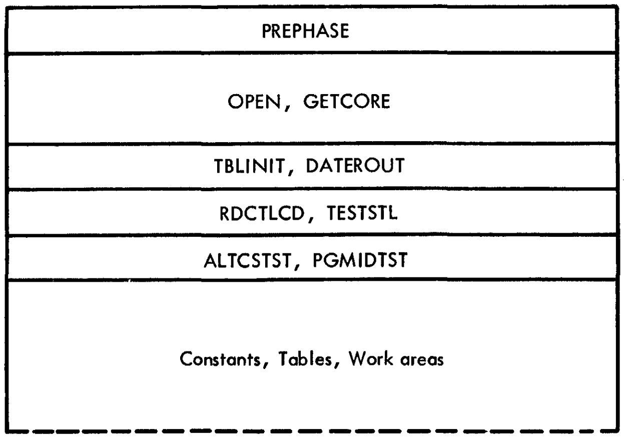


Figure 6. Prephase Storage Allocation Map

GETCORE

- Gets core for processor tables
- Computes table sizes and addresses

OPEN

- Opens data sets

DATEROUT

- Picks up data from OS
- Converts data from YYDDD format to month/day/year
- Stores converted data in CIOEX data area

RDCTLCD

- Reads a card and prints it
- Tests to determine if it is a control card

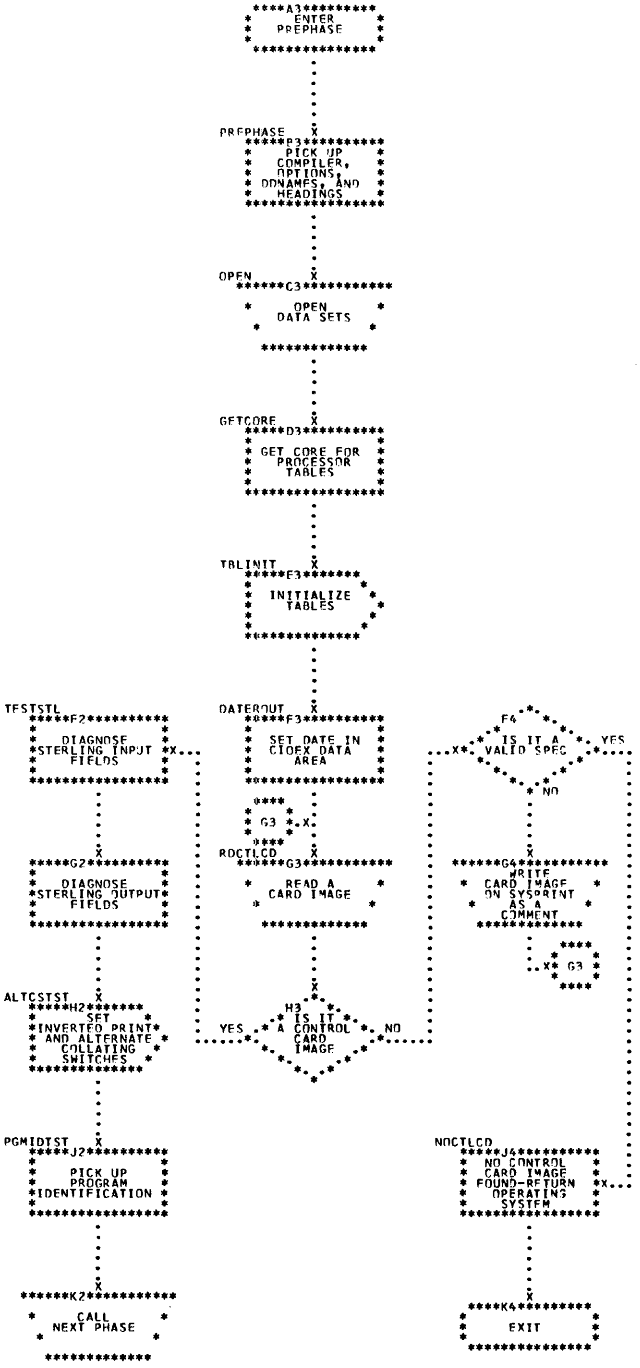

A3
ENTER PREPHASE
PREPHASE
B3
PICK UP COMPILER, OPTIONS, DDNAMES, AND HEADINGS
OPEN
C3
OPEN DATA SETS
GETCORE
D3
GET CORE FOR PROCESSOR TABLES
TBLINIT
E3
INITIALIZE TABLES
TESTSTL
F2
DIAGNOSE STERLING INPUT FIELDS
DATEROUT
F3
SET DATE IN CIOEX DATA AREA
F4
IS IT A VALID SPEC
YES
NO
G3
RDCTLCD
G3
READ A CARD IMAGE
G2
DIAGNOSE STERLING OUTPUT FIELDS
G4
WRITE CARD IMAGE ON SYSPRINT AS A COMMENT
G3
ALTCSTST
H2
SET INVERTED PRINT AND ALTERNATE COLLATING SWITCHES
H3
IS IT A CONTROL CARD IMAGE
YES
NO
PGMIDTST
J2
PICK UP PROGRAM IDENTIFICATION
NOCTLCD
J4
NO CONTROL CARD IMAGE FOUND-RETURN OPERATING SYSTEM
K2
CALL NEXT PHASE
K4
EXIT

Chart BA. Prephase

TESTSTL

- Performs diagnostics on sterling input and output entries

- Checks to determine if inverted print is requested

ALTCSTST

- Checks to determine if alternate collating sequence is requested

PGMIDTST

- Checks to determine if the program has a name

- Uses 'RPGOBJ' if it does not

SUBROUTINES

IBLINIT

- Initializes the literal table, field name table and the compression area with X'FE'

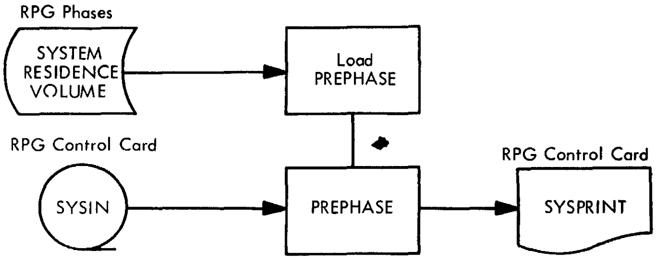


Figure 7. Prephase Input/Output Flow

Table 3. RPG Control Card Switches

Column	Entry	Setting
6	H	Condition = Diagnostic continues. ≠ Discontinue run if entry is F, E, L, I, C, O. Otherwise treat as a comment.
7-16	blank	Not used
17	 2 1 blank other	Set sterling code byte to: 10xxxxxx (BSI format) 01xxxxxx (IBM format) 00xxxxxx (no sterling) 01xxxxxx Assume IBM format
18	2 1 blank other	xx10xxxx xx01xxxx xx00xxxx xx01xxxx Assume IBM format
19	2 1 blank, 0 other	xxxx10xx xxxx01xx xxxx00xx xxxx01xx Assume IBM format
20	2 1 blank, 0 other	xxxxxx10 xxxxxx01 xxxxxx00 xxxxxx01 Assume IBM format
21	 I blank other	Set Inverted Print byte to: X'F0' X'00' X'F0' Assume I
22-25	blank	Not used
26	 A blank other	Set Alternate Collating Sequence byte to: X'F0' X'00' X'F0' Assume A
27-74	blank	Not used
75-80	blank	Program name assumed to be RPGOBJ. Otherwise use contents of 75-80 as Program Name.

INTRODUCTION

The System/360 RPG file description specifications define the characteristics of all files (input, output, RAF, table or chaining) that are to be used in an RPG program.

Enter Phase 1 processes the file description specification source statements. The functions of this phase are

1. Read specification input statement
2. Diagnose errors
3. Build file name table
4. Compress the file description specification input statement
5. Build field name table
6. Print processed statement and any applicable diagnostic codes
7. Call Enter Phase 2 or Enter Phase 3

Figure 8 and Chart CA illustrate the organization and operation of Enter Phase 1. Figure 9 illustrates the input/output flow for Enter Phase 1.

LOGIC

Each field entry in the file description specifications is analyzed for validity of format. An assumption of a valid entry will be made, if possible, to correct a recognized error condition. A diagnostic code is printed to indicate an assumption has been made for an invalid entry. Irrecoverable error diagnostics are also printed to indicate the specification was not processed.

Processed specifications provide entries for the file name table (Appendix A). Each entry in the table is 16 bytes long, and the table has a capacity of ten entries. If the table capacity is exceeded, the additional entries are treated as comments and a diagnostic code is printed. Before an entry is made in the table, a search is made to determine if the entry is already present. Multidefined files are entered in the table only once. A diagnostic code is printed for all successive occurrences of a multidefined file. For Enter Phase 1, the assembled entry is as follows:

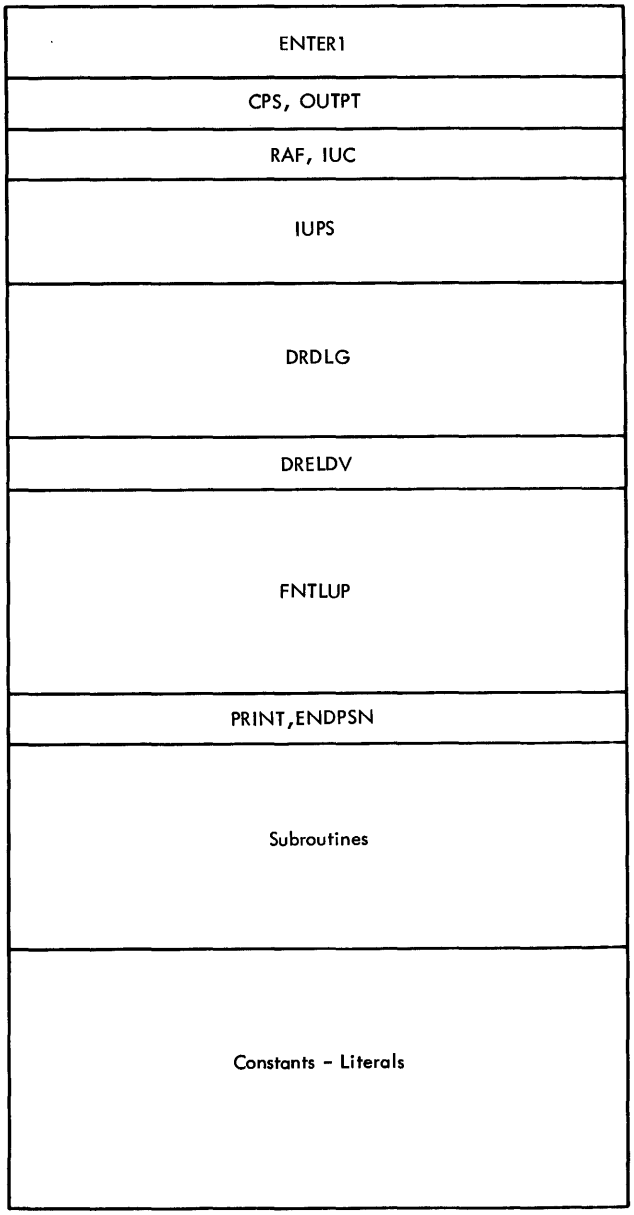
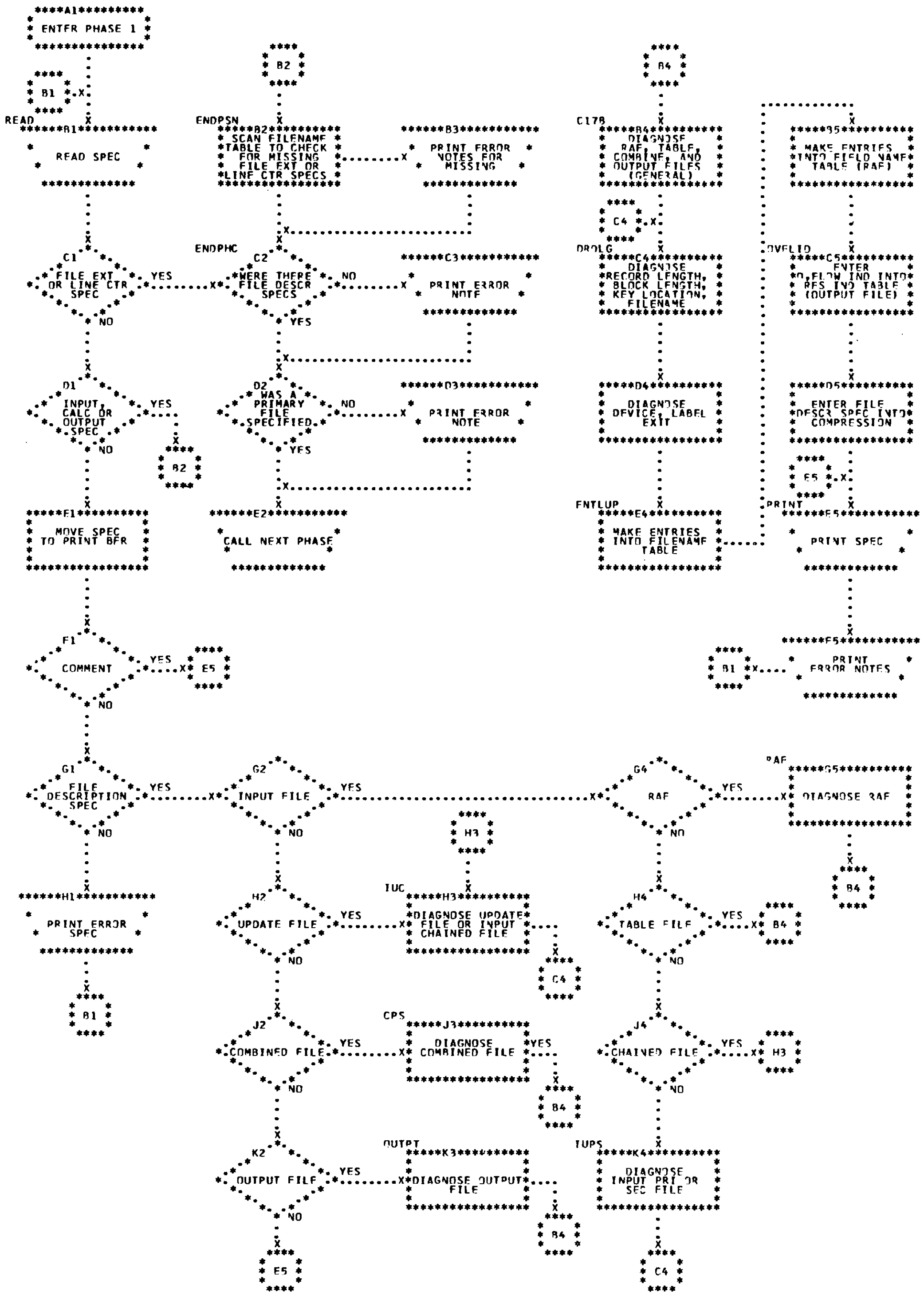


Figure 8. Enter Phase 1 Storage Allocation Map



A1
ENTER PHASE 1
B1
READ
B1
READ SPEC
C1
FILE EXT OR LINE CTR SPEC
YES
NO
D1
INPUT, CALC OR OUTPUT SPEC
YES
NO
B2
E1
MOVE SPEC TO PRINT BFR
F1
COMMENT
YES
E5
NO
G1
FILE DESCRIPTION SPEC
YES
NO
H1
PRINT ERROR SPEC
B1
B2
ENDPSN
B2
SCAN FILENAME TABLE TO CHECK FOR MISSING FILE EXT OR LINE CTR SPECS
B3
PRINT ERROR NOTES FOR MISSING
ENDPHC
C2
WERE THERE FILE DESCR SPECS
NO
YES
C3
PRINT ERROR NOTE
D2
WAS A PRIMARY FILE SPECIFIED
NO
YES
D3
PRINT ERROR NOTE
E2
CALL NEXT PHASE
G2
INPUT FILE
YES
NO
H3
IUC
H3
DIAGNOSE UPDATE FILE OR INPUT CHAINED FILE
C4
H2
UPDATE FILE
YES
NO
J2
COMBINED FILE
YES
NO
CPS
J3
DIAGNOSE COMBINED FILE
YES
B4
K2
OUTPUT FILE
YES
NO
E5
OUTPT
K3
DIAGNOSE OUTPUT FILE
B4
B4
C17B
B4
DIAGNOSE RAF, TABLE, COMBINE, AND OUTPUT FILES (GENERAL)
C4
DROLG
C4
DIAGNOSE RECORD LENGTH, BLOCK LENGTH, KEY LOCATION, FILENAME
D4
DIAGNOSE DEVICE, LABEL EXIT
FNTLUP
E4
MAKE ENTRIES INTO FILENAME TABLE
G4
RAF
YES
NO
H4
TABLE FILE
YES
B4
NO
J4
CHAINED FILE
YES
H3
NO
TUPS
K4
DIAGNOSE INPUT PRI OR SEC FILE
C4
B5
MAKE ENTRIES INTO FIELD NAME TABLE (RAF)
OVFLID
C5
ENTER O.FLOW IND INTO RES IND TABLE (OUTPUT FILE)
D5
ENTER FILE DESCR SPEC INTO COMPRESSION
E5
PRINT
E5
PRINT SPEC
F5
B1
PRINT ERROR NOTES
RAF
G5
DIAGNOSE RAF
B4

Chart CA. Enter Phase 1

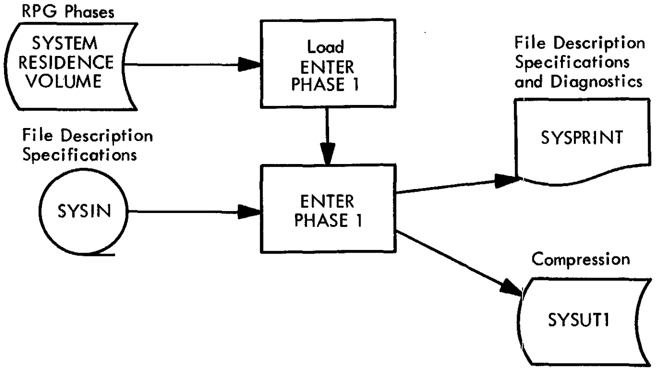


Figure 9. Enter Phase 1 Input/Output Flow

1. Bytes 1-8	Filename	Columns 7-14
2. Byte 9	Reference byte	Always blank for file description specification
3. Bytes 10-11	Sequence number	Generated internal sequence number
4. Byte 12	Type byte	File usage (Appendix A)
5. Bytes 13-16	Record length	Columns 24-27

An entry is made in the field name table (Appendix B) for every accepted specification that is a record address file.

For Enter Phase 1 the assembled entry is as follows:

1. Bytes 1-6	Field name	CONTDb or columns 54-59
2. Byte 7	Reference byte	R or X'D9'
3. Bytes 8-9	Field address	Blank until filled during assign phase X'40'
4. Byte 10 (Beta)	Field information byte	X'00' (Appendix B)
5. Byte 11	Field length	(Columns 29-30 L minus 1) or X'03'
6. Byte 12	Decimal positions	X'40' (alphameric)
7. Byte 13 (Gamma)	Field type byte	X'04' (Appendix B)

All specifications that are processed, and any applicable diagnostic codes, are printed.

At the conclusion of Enter Phase 1, a scan is made of the file name table to determine if at least one file description has been processed and that there is only one primary file. Diagnostic codes are printed for any error situation detected.

The internal sequence number is incremented by one and placed in SEQINP in the CIOEX data area for use by the next phase. The next phase called is Enter Phase 2, if the next input record is a file extension specification or line counter specification; otherwise, Enter Phase 3 is called.

If Enter Phase 3 is called, a scan is made of the file name table to determine if any files should have been referenced by a file extension or line counter specification. Diagnostic codes are printed for any error situation detected.

SWITCHES AND INDICATORS

INDSW

X'01'	Column 28--R; column 31--K; column 32--I. This indicates a random, key field location, indexed-sequential file is present
X'02'	A record address file is present. Left ON for rest of Enter Phase 1
X'04'	A primary file is present. Left ON for the rest of Enter Phase 1.
X'40'	One valid file description specification has been processed. Left ON
X'80'	Ten file description specifications have been processed. Left ON

ZEROSW

Contains the highest character processed during a scan of a numeric field. A zero or blank in this switch indicates an error.

MAIN ROUTINES

ENTER1

- Reads the input record
- Verifies form type, file type, file designation
- Enters an R or T in the type byte
- Calls the next phase

CPS

- Diagnoses a combined file designated as primary or secondary
- Enters an I or E in the type byte

OUTPT

- Diagnoses an output file
- Enters an O or P in the type byte

RAF

- Diagnoses a record address file

IUC

- Diagnoses an input file or an update file (designated as a chained file)
- Enters a C, D, W, or X in the type byte

IUPS

- Diagnoses an input file or an update file (designated as a primary or secondary file)
- Enters an I, E, U, V, Y, or Z in the type byte

DRDLG

- Diagnoses record length, block length, key field starting location, filename, device,and labels

DRELDV

- Diagnoses file type, and device combinations

FNTLUP

- Provides table lookup and entry routines for file name table and field name table
- Enters file description specification into compression (Appendix E)

PRINT

- Prints specification and diagnostic codes

ENDPSN

- Scans file name table
- Checks for missing file description cards and primary file description
- Calls next phase

SUBROUTINES

All returns are made from the subroutines by an instruction BR REG11, unless stated otherwise.

SUBC16

- Diagnoses the file designation entry (column 16) for primary (P) or secondary (S)

SUBC18

- Diagnoses the sequence entry (column 18)

SUBC17

- Diagnoses the end of file designation (column 17)

SUB19F

- Diagnoses the file format designation (column 19)

SUBC39

- Diagnoses the extension code designation (column 39)

SUBC24

- Assigns a diagnostic code to indicate an invalid or missing record length designation (columns 24-27)

SUBC20

- Assigns a diagnostic code to indicate an invalid or missing block length designation (columns 20-23)

SUBC07

- Assigns a diagnostic code to indicate an invalid or missing filename designation (columns 7-14)
- Exits to PRINT

SUBC35

- Assigns a diagnostic code to indicate an invalid or missing key field starting location designation (columns 35-38)

SUBC54

- Assigns a diagnostic code to indicate an invalid or missing name of label exit designation (columns 54-59)
- Exits to PRINT

ALPAMR

- Diagnoses individual columns of a given field for valid alphabetic or numeric characters or blanks
- Exits to the location designated by register 9 if an error is detected

SUBNUM

- Diagnoses individual columns for valid numeric characters or blanks
- Exits via register 9 if an error condition is detected

FLTLUP

- Executes a scan of the field name table
- Exits via register 14 if an equal entry is found
- Exits via register 11 if an open entry (end of the table) is found

TLOKUP

- Executes a scan of the file name table
- Exits via register 14 if an equal entry is found
- Exits via register 15 on table overflow
- Exits via register 11 if an open entry (end of the table) is found

ERRSUB

- Advances to the next entry in the error table
- Exits via register 5

TCOPOF

- Tests for compression area overflow
- Gives the calling sequence to CIOEX to put out compression

ENTER PHASE 2

INTRODUCTION

The System/360 file extension specifications provide RPG with information about chaining, table, or record address files.

The System/360 line counter specifications provide RPG with information about reports that will be printed. The carriage control information links the line to be printed with the corresponding punch in the carriage control tape.

Enter Phase 2 processes the file extension specifications and line counter specifications source statements. The functions of this phase are

1. Read specification input statement
2. Diagnose errors
3. Search and update file name table
4. Search and update field name table
5. Compress specification input statement
6. Print processed statement and any applicable diagnostic codes
7. Scan file name table for error condition diagnosis
8. Call Enter Phase 3

Figure 10 and Chart DA illustrate the organization and operation of Enter Phase 2. Figure 11 illustrates the input/output flow for Enter Phase 2.

LOGIC

In processing file extension statements, the from filename entry and the to filename entry are checked against the file name table. If the entry is found, the reference byte is changed to E. The type byte in the table entry determines the type of file to be processed. A filename entry without a corresponding entry in the file name table is an error condition.

The field name table is searched for an entry to correspond to the statement entry for conversion routine name or table name (columns 27-32) or second table name (columns 46-51). If a corresponding entry is found in the table, and the entry being processed is a table name, the reference byte is changed to M to indicate a multi-defined condition; otherwise, the entry is added to the table and the reference byte is created as blank. If the table overflows, subsequent additional table names and/or conversion routine names are diagnosed as errors and are not processed.

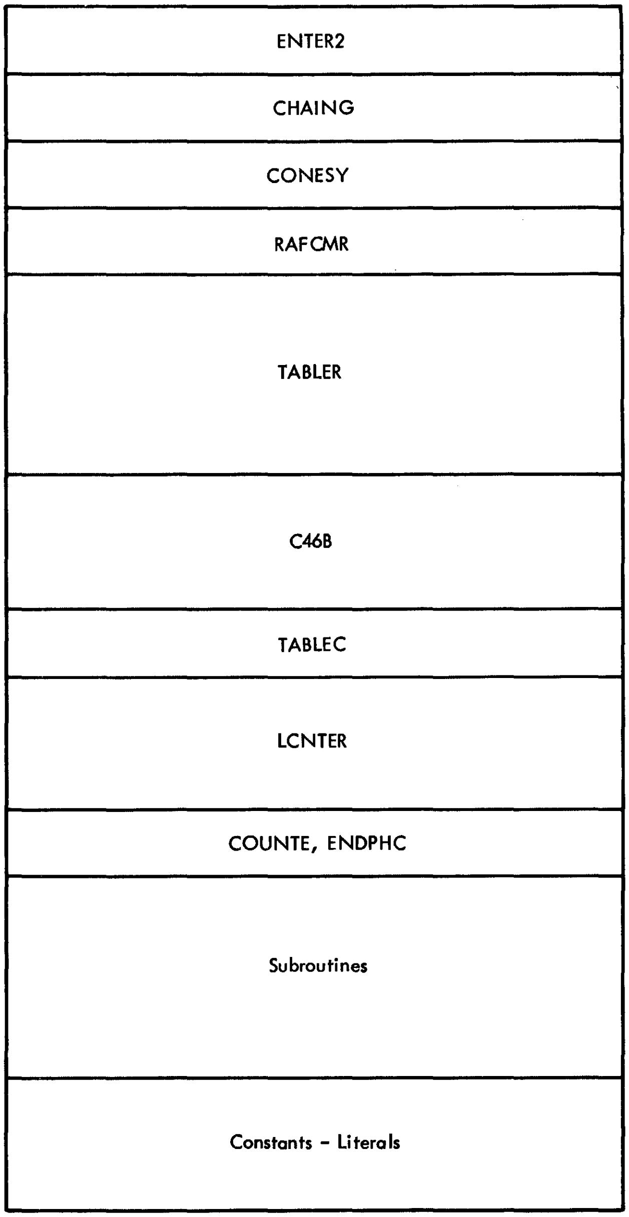


Figure 10. Enter Phase 2 Storage Allocation Map

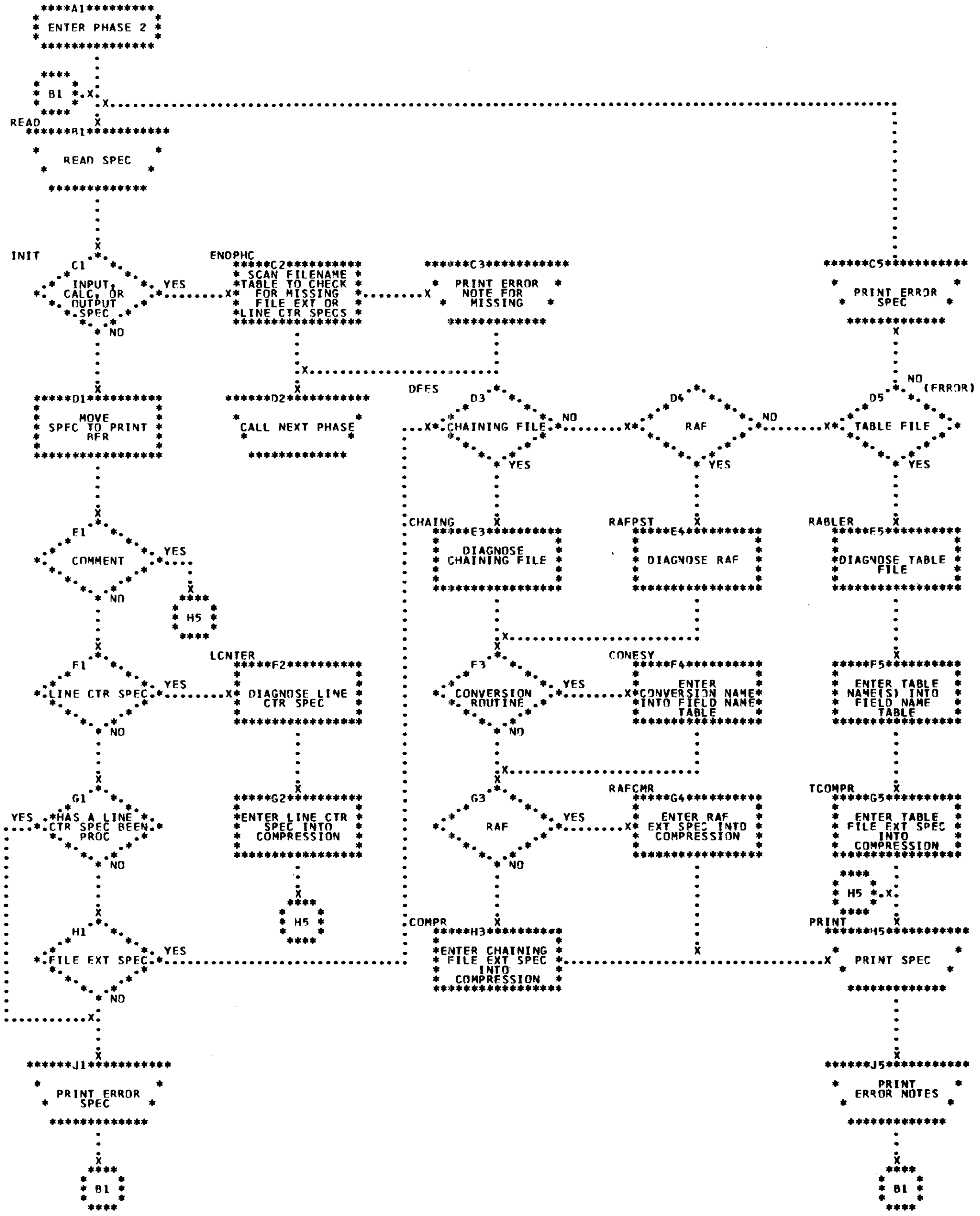


Chart DA. Enter Phase 2

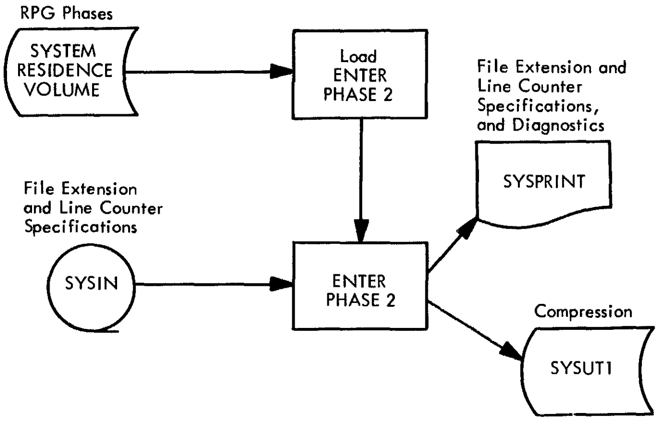

RPG Phases
SYSTEM RESIDENCE VOLUME
Load ENTER PHASE 2
File Extension and Line Counter Specifications, and Diagnostics
SYSPRINT
File Extension and Line Counter Specifications
SYSIN
ENTER PHASE 2
Compression
SYSUT1

Figure 11. Enter Phase 2 Input/Output Flow

For entries that are made in the field name table during Enter Phase 2, the assembled entry is as follows:

1. Bytes 1-6	Field name	Columns 27-32 or 46-51 for table name or conversion routine name
2. Byte 7	Reference byte	Blank or M
3. Bytes 8-9	Field address	Blank
4. Byte 10 (Beta)	Field information byte	X'00' for conversion routine name X'04', X'05', X'06' for table name (Appendix B)
5. Byte 11	Field length	Columns 40-42 or 52-54 (binary) minus 1 for table name or X'03' for conversion routine name
6. Byte 12	Decimal positions	Column 44 or 56 for table name or X'40' for conversion routine name
7. Byte 13 (Gamma)	Field type byte	X'08' for table name or X'20' for conversion routine name (Appendix B)

As each file extension specification is compressed (Appendix E), the from filename and to filename entries are replaced by the sequence numbers from the file name table entries.

In processing the line counter statements, the filename entry is checked against the file name table. If the entry is found, the reference byte is changed to L if it is blank or E. A diagnostic code is assigned to indicate a multidefined condition, if the reference byte is already L.

Each line number entry is converted to binary and placed in the corresponding channel number position (1-12) in the line counter table. If either channel 1 or channel 12 has not been specified, a diagnostic code is assigned to the specification.

As each line counter specification is compressed (Appendix E), the filename entry is replaced by the generated sequence number from the file name table entry. The sequence number is converted to binary for the compression.

At the conclusion of Enter Phase 2 a scan is made of the file name table to determine if any files should have been, but were not, referenced by file extension or line counter specifications. Diagnostic codes are printed for files which require a reference.

Enter Phase 3 is called when an input record with an I (input specification) is detected. The internal sequence number is incremented by one and moved to the sequence number of the first input specification.

SWITCHES AND INDICATORS

INDSW

X'08'	One RAF is present. Left ON for rest of Enter Phase 2
X'10'	An RAF or chaining file that specifies a conversion routine name is being processed
X'20'	An RAF (not a chaining file) is being processed or a second table name is specified
X'40'	At least one line counter specification has been processed
X'80'	100 table file extension specifications have been processed. Left ON

ZEROSW

Contains the highest character processed during a scan of a numeric field. It is an error if this switch contains a zero or blank at the end of a complete scan.

MAIN ROUTINES

ENTER2

- Reads the input record, and verifies form type

- Sends line counter specifications to LCNTER
- Verifies from filename of file extension specifications, in the file name table, along with the corresponding type byte
- Sends chaining files to CHAING

CHAING

- Diagnoses chaining files by verifying number of chaining field entry (1-9)
- Diagnoses record address files
- Verifies the to filename and corresponding type byte in the file name table

CONESY

- Diagnoses the name of the conversion routine entry (table name) for RAF or chaining files
- Compresses chaining file entries
- Verifies conversion routine name in the field name table

RAFCMR

- Compresses record address file entries (Appendix E)
- Turns off INDSW setting X'20' and turns on setting X'08'

TABLER

- Diagnoses table files by verifying the to filename entry with the corresponding valid type byte in the file name table
- Verifies number of table entries per record, number of table entries per table, length of table entry, decimal position, and sequence entries
- Converts length of table entry to binary
- Verifies table name and moves it into field name table

C46B

- Diagnoses the second group of table entries in the same manner as TABLER does for the first group

TABLEC

- Routine for table file compression

LCNTER

- Diagnoses line counter specifications by verifying filename in the file name table and verifying the line number and channel number entries
- Converts the line number entries to binary and stores them in the line counter table
- Compresses the entries and the table

COUNTE

- Updates the sequence counter
- Prints the specification and the related diagnostic codes
- Returns the program to READ

ENDPHC

- Scans the file name table to determine if there are file extension or line counter specifications missing
- Calls Enter Phase 3 as the last logical step of this phase

SUBROUTINES

All returns are made from the subroutines by an instruction BR REG11 unless stated otherwise.

SUBNUM

- Diagnoses individual columns of a numeric field for valid numeric characters or blanks
- Exits via register 9 if an error is detected

ALPAMR

- Diagnoses individual columns of an alphameric field for valid alphabetic or numeric characters or blanks
- Exits to the location designated by register 9 if an error is detected

TCOPOF

- Tests compression overflow

- Gives the calling sequence to CIOEX
- Puts out compression

ERRSUB

- Advances to the next entry in the error table
- Exits via register 5

TLOKUP

- Scans the file name table for a given entry
- Exits via register 14 if an equal entry is found
- Exits via register 11 when a complete scan of the table has been made

SUBREF

- Places the reference byte E into the file name table

SUBCON

- Sets INDSW setting X'20' OFF
- Assigns a diagnostic code for an invalid conversion name
- Exits to OUT

FLTLUP

- Scans the field name table for a given entry
- Exits via register 15 if the end of the table is reached without an equal entry
- Exits via register 9 if an equal entry is found
- Exits via register 11 if an open entry (without an equal entry) is found

FDFULL

- Entered from FLTLUP on a full field name table condition
- Sets INDSW settings X'10' and X'20' OFF
- Assigns a diagnostic code for field name table overflow
- Returns to OUT

SUBC47

- Assigns a diagnostic code for an invalid number of table entries per record

SUBC48

- Assigns a diagnostic code for an invalid number of table entries per table

SUBC49

- Assigns a diagnostic code for an invalid length of table entry
- Exits via register 15

A4042

- Assumes length of table entry 1 is 10

SUBC51

- Assigns a diagnostic code for an invalid decimal position entry or entries

SUBC50

- Assigns a diagnostic code for an invalid packed (P) entry or entries

SUBC41

- Assigns a diagnostic code for an invalid sequence (A/D) entry or entries

SUBC27

- Assigns a diagnostic code that indicates the first three characters of table name are not TAB

A5254

- Assumes length of table entry 2 is 10

SUBC45

- Assigns a diagnostic code that indicates the last three characters of table name are not alphameric
- Exits to OUT

SUBC33

- Assigns a diagnostic code that indicates the length of table entry for an alphameric field exceeds 256 characters

SUBC38

- Assigns a diagnostic code that indicates the length of table entry for a numeric field exceeds 15 digits

INTRODUCTION

The System/360 RPG input specifications are divided into two categories. The first type, record identification, specifies the record codes that identify a record and relate it to other records in the file. Field description type specifications describe the fields of the record. Record identification type specifications are followed by their corresponding field descriptions.

Enter Phase 3 processes the input specification source statements. The functions of this phase are

1. Read specification input statement
2. Diagnose errors
3. Search and update file name table
4. Search and update resulting indicator table
5. Search and update field name table
6. Compress specification input statement
7. Print processed statement and applicable diagnostic codes
8. Scan file name table for error diagnosis
9. Call Enter Phase 4 or Enter Phase 6

Figure 12 and Charts EA and EB illustrate the organization and operation of Enter Phase 3.

Figure 13 illustrates the input/output flow for Enter Phase 3.

LOGIC

In processing record identification type input specifications, the filename entry is verified in the file name table. The reference byte of the located entry is verified and changed to an R (multidefined) if it is blank or E. The generated sequence number from the table entry is entered into the compression for the specification. The specification is skipped if a table entry is not found.

Record identification specifications without a filename entry are identified by a record sequence entry. To identify the records within a file, the sequence entry is placed in the compression.

In processing field description type input specifications, the field name entry is placed in the field name table. The reference byte is created as blank for a new entry in the table. If the entry has previously been put in the table, the logical AND of the field type byte (gamma) and the search name field mask are formed. If the result is zero, the field length values of the search entry and the table entry (bytes 11,12) are compared. The reference byte (byte 7) is changed to M (multidefined) if these values are not equal.

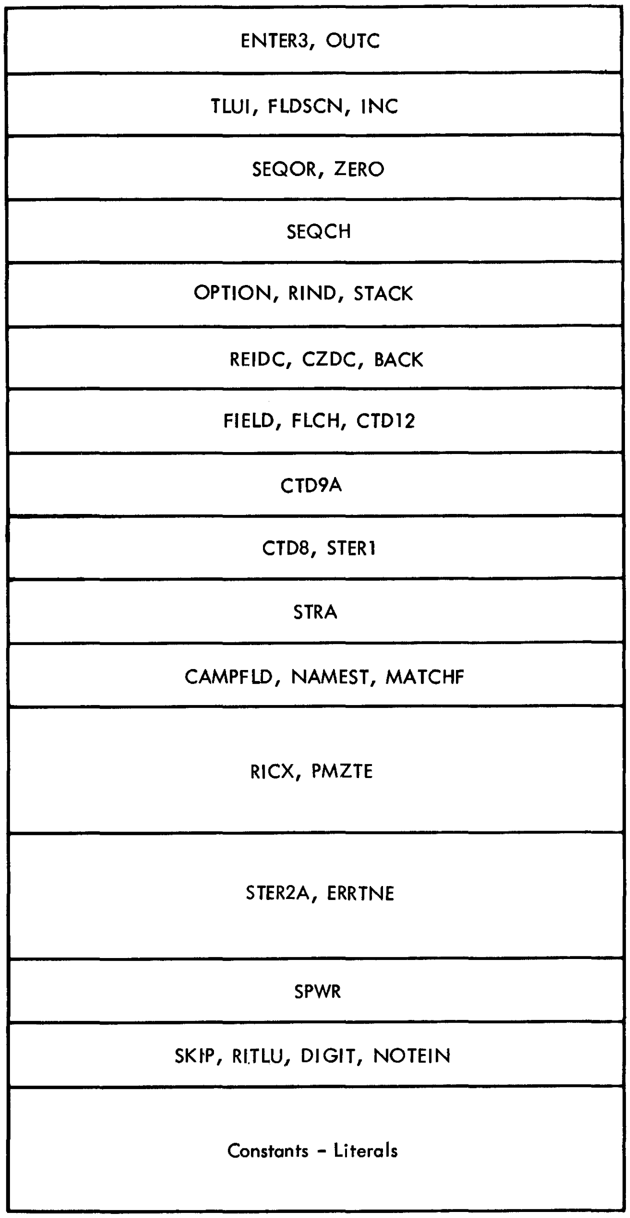


Figure 12. Enter Phase 3 Storage Allocation Map

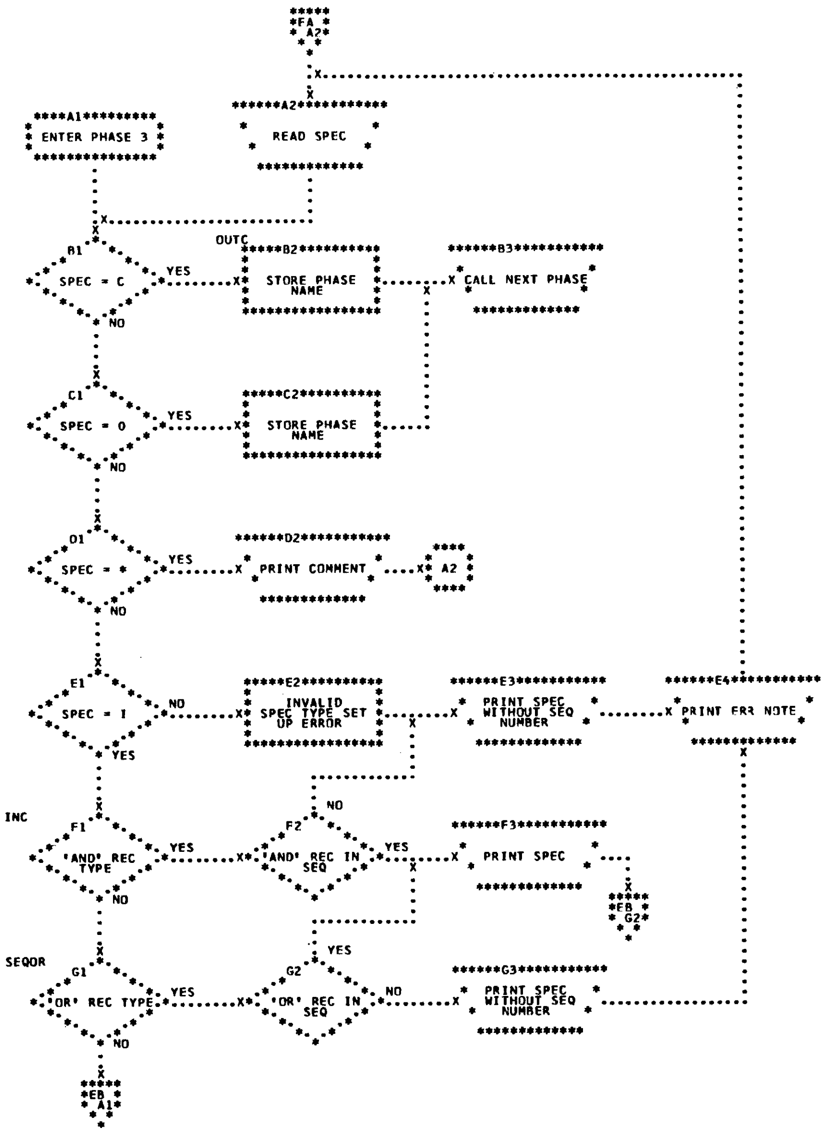


Chart EA. Enter Phase 3

When the field name table overflows, the sequence number of the specification causing the overflow is saved in the CIOEX data area. X'FD' is placed at the end of the table to indicate the table has overflowed.

The resulting indicator table (Appendix D) is referenced when Enter Phase 3 verifies the resulting indicator entry, the control level entry, matching fields entry, field-record relation entry, and field indicators entries. The resulting indicator specified in a record identification specification is defined by setting the reference byte REF (used only by this phase) to X'01' and ORing it with the reference byte of the corresponding indicator in the table (byte 3).

The control level entry from the field description type specification is defined by setting the phase reference byte (REF) to X'01'. A chaining field indicator (C1-C9) or a matching field indicator (M1-M3) causes the matching record indicator (MR) to be referenced. This is accomplished by

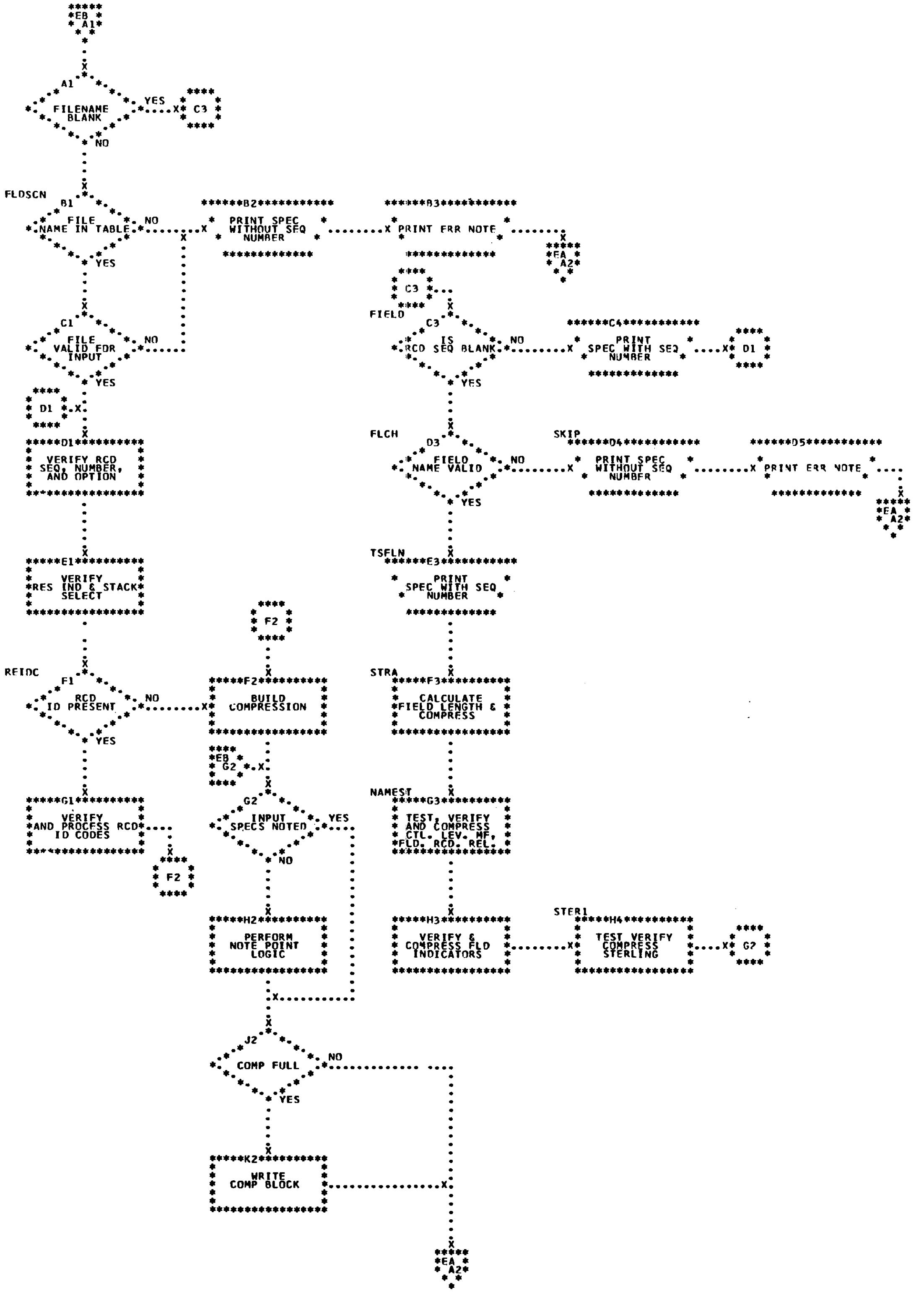

EB A1
A1 FILENAME BLANK
YES
C3
NO
FLDSCN
B1 FILE NAME IN TABLE
NO
B2 PRINT SPEC WITHOUT SEQ NUMBER
B3 PRINT ERR NOTE
EA A2
YES
C1 FILE VALID FOR INPUT
NO
YES
D1
D1 VERIFY RCD SEQ. NUMBER, AND OPTION
E1 VERIFY RES IND & STACK SELECT
REIDC
F1 RCD ID PRESENT
NO
YES
G1 VERIFY AND PROCESS RCD ID CODES
F2
F2 BUILD COMPRESSION
EB G2
G2 INPUT SPECS NOTED
YES
NO
H2 PERFORM NOTE POINT LOGIC
J2 COMP FULL
NO
YES
K2 WRITE COMP BLOCK
EA A2
C3
FIELD
C3 IS RCD SEQ BLANK
NO
C4 PRINT SPEC WITH SEQ NUMBER
D1
YES
FLCH
D3 FIELD NAME VALID
NO
SKIP
D4 PRINT SPEC WITHOUT SEQ NUMBER
D5 PRINT ERR NOTE
EA A2
YES
TSFLN
E3 PRINT SPEC WITH SEQ NUMBER
STRA
F3 CALCULATE FIELD LENGTH & COMPRESS
NAMEST
G3 TEST, VERIFY AND COMPRESS CTL. LEV. MF, FLD. RCD. REL.
H3 VERIFY & COMPRESS FLD INDICATORS
STER1
H4 TEST VERIFY COMPRESS STERLING
G2

Chart EB. Enter Phase 3

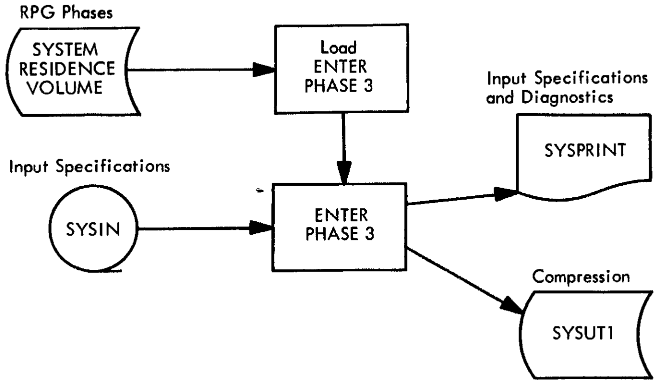


Figure 13. Enter Phase 3 Input/Output Flow

setting the phase reference byte REF to X'03'. The field-record relation entry sets REF to X'02' and is ORed with the corresponding indicator in the table. The field indicator entries set REF to X'01' for plus or minus and to X'05' for zero or blank. The status of a resulting indicator is determined by the value of its reference byte (Appendix D). The final diagnostics of the resulting indicator table are made by the assign phases.

Each specification is compressed (Appendix E) and after the last input specification has been processed a final diagnostic scan is made of the file name table for unreferenced, unused, and multidefined entries. The table is then cleared and initialized for the literal entries to be made later.

If the next input record is a calculation specification, Enter Phase 4 is called. For an output-format specification input record, Enter Phase 6 is called.

SWITCHES AND INDICATORS

CLSW

X'FF' Indicates an improperly specified control level

ENTSW

X'FF' Indicates an invalid plus or minus indicator

FLDSW

X'FF' Entry may be made in the field name table

INSW

X'FF' Input specification encountered

FSW

X'FF' Resulting indicator table search located a valid entry

INIT

X'FF' Specification is out of sequence; print without sequence number

FIIND

X'FF' AND or OR card not in sequence

MAIN ROUTINES

ENTER3

- Reads the input record and verifies the form type
- Assigns a diagnostic code for an invalid specification type
- Tests for an AND card
- Assigns a diagnostic code for an AND card which is out of sequence
- Tests for an OR card
- Assigns a diagnostic code for an OR card which is out of sequence
- Sets the alpha byte (bits 1 and 2) to reflect the record type

RECORD IDENTIFICATION

ZERO

- Tests for a valid filename entry
- Sets the file name table reference byte to R if blank or E
- Assumes file name if file name and field name entries are both present
- Sets the alpha byte (bit 5) to reflect this entry

SEQCH

- Verifies the sequence entry
- Places the sequence entry and the

generated sequence number (from the file name table) in compression

- Assumes numeric sequence for an error situation
- Sets the alpha byte (bits 1, 2, and 4) to reflect conditions tested in this routine

OPTION

- Verifies the option (0) entry
- Assigns a diagnostic code if the entry is other than blank or 0
- Sets the alpha byte (bit 3) to reflect this entry

RIND

- Verifies the resulting indicator entry
- Assigns a diagnostic code if the entry is blank or invalid
- Assumes an entry of 99 for an error situation
- References this entry in the resulting indicator table
- Places the entry in compression

STACK

- Tests the stacker select entry
- Places the stacker select entry in compression
- Sets the alpha byte (bit 1) in the record type compression to reflect the entry
- Assigns a diagnostic code for an entry other than blank or numeric

REIDC

- Diagnoses the record identification codes entries
- Assigns diagnostic codes for embedded blanks and non-numeric entries
- Assumes an entry of N if the NOT entry is other than blank or N
- Assumes an entry of C for an entry other than C/Z/D
- Moves the entry into the T byte of the record type compression (Appendix E)
- Moves the number of record codes into the alpha byte (bits 6-7) of the compression
- Writes the compression block when full

FIELD DESCRIPTION

FIELD

- Diagnoses the field name entry by checking for left-justification, alphameric characters, and embedded blanks

CTD9A

- Diagnoses and packs field location entries (from and to)
- Verifies the entries for decimal positions
- Clears field indicators if they are present
- Verifies the length of the alphameric or the numeric field

STER1

- Diagnoses the entries that describe a sterling field
- Calculates and converts the field length

STRA

- Calculates the binary length for a packed numeric field
- Stores the decimal positions and the field type indicator in compression (field type)

CMPFLD

- Searches, tests, and adds to the field name table
- Places the field name in compression (field type)

NAMEST

- Diagnoses the control level entry
- Enters the control level entry specified in the resulting indicator table and also into compression

- Assigns a diagnostic code if the specification entry is not valid

MATCHF

- Diagnoses the entry for matching fields or chaining fields
- Branches the program to RICX for a chaining fields indicator
- Verifies a matching fields indicator (any other entry is assumed to be a matching fields indicator) for an entry greater than 3
- Sets the alpha byte in compression to reflect this entry (bit 1 and bit 7)
- Moves the indicator to compression

RICX

- Diagnoses the entry for field-record relation
- References the entry in the resulting indicator table
- Moves the entry into compression
- Sets the alpha byte (bit 2) to reflect this entry
- Assigns a diagnostic code if the entry is invalid

PMZTE

- Diagnoses entries for field indicators-plus, minus, zero, or blank
- References a valid entry in the resulting indicator table
- Moves the valid entry into compression
- Sets the alpha byte (bits 3, 4, 5) to reflect the indicator entries

STER2A

- Diagnoses the entry for sterling sign position
- Analyzes each position of the field for a blank or numeric character
- Replaces blanks by zeros
- Compares the entry to the from and to entries of field location
- Moves the entry into compression
- Sets the alpha byte (bit 6) to reflect this entry
- Places a D in the type of specification position of compression

OUTC

- Sets up the linkage to Enter Phase 4 or Enter Phase 6
- Scans the file name table for unreferenced, unused, and multidefined filenames
- Clears the file name table
- Prints each error and diagnostic code

SUBROUTINES

TLU1

- Scans the file name table for a given filename entry
- Assigns a diagnostic code if the entry is not found

FLDSCN

- Scans the field name table for a given field name entry
- Sets FLDSW if there is a space for an entry to be made
- Verifies the usage of an entry that is found in the table
- Assigns a diagnostic code for invalid use of a field name

ERRTNE

- Sets up the diagnostic codes to be printed by the print error service routine
- Returns via Register 5

SPWR

- Prepares the input specification for printing by PNTSPC
- Returns via Register 14

SKIP

- Skips a specification that is a comment or contains an error that prevents further processing
- Reduces the sequence number
- Prints the specification and the error code if applicable
- Returns to RDSPEC

RITLU

- Scans the resulting indicator table for a given entry
- Updates the reference byte to reflect an OR condition if the indicator is found
- Returns via Register 15

DIGIT

- Verifies and packs a given four-position numeric field
- Replaces invalid characters and embedded blanks by zeros in a position-by-position scan for numeric characters

ENTER PHASE 4

INTRODUCTION

Enter Phase 4 is a preprocessing (partial processing) phase for the calculation specifications. The preprocessing consists of scanning the specifications, building literal entries, and performing preliminary diagnostics. In addition, specifications containing the operation codes EXTCV, RPGCV, KEYCV, RLABL, ULABL, TAG, GOTO, EXIT, and ERPGC are compressed in format, but not actually entered into compression until Enter Phase 5.

The remaining specifications, partially processed, are referred to as preprocessed specifications. They will be compressed in Enter Phase 5.

All specifications, whether compressed or preprocessed, are written on work data set 3 (SYSUT3), with any applicable error notes, for processing during Enter Phase 5.

When the first output-format specification is encountered, it is written on SYSUT3 and Enter Phase 5 is called.

Figure 14 and Charts FA and FB illustrate the organization and operation of Enter Phase 4.

Figure 15 illustrates the input/output flow for Enter Phase 4.

LOGIC

The specification is checked to determine its type, i.e., output-format, comment, or calculation. Only calculation specifications are processed by Enter Phase 4.

The factor 1, factor 2, result, and operation fields are scanned in that sequence. Blanks or a field name in factor 1 cause a branch to scan factor 2. If factor 2 is blank or contains a field name, the result field is scanned. A literal in either factor field causes a literal entry to be built (for use in building a literal table in Enter Phase 5) before proceeding with the next scan. A literal entered in result field is an error condition.

Following the factor entries and result field entry processing, the operation entry is checked. If it contains one of the operation codes to be compressed by Enter Phase 4, it is verified (Table 4), compressed, and written out on SYSUT3.

Verification consists of checking for the presence of the proper factor and/or result field entries for the particular operation code (Table 5). If the operation code is not one of those to be compressed by Enter Phase 4, the specification is written on SYSUT3 without being compressed, and the next specification is read.

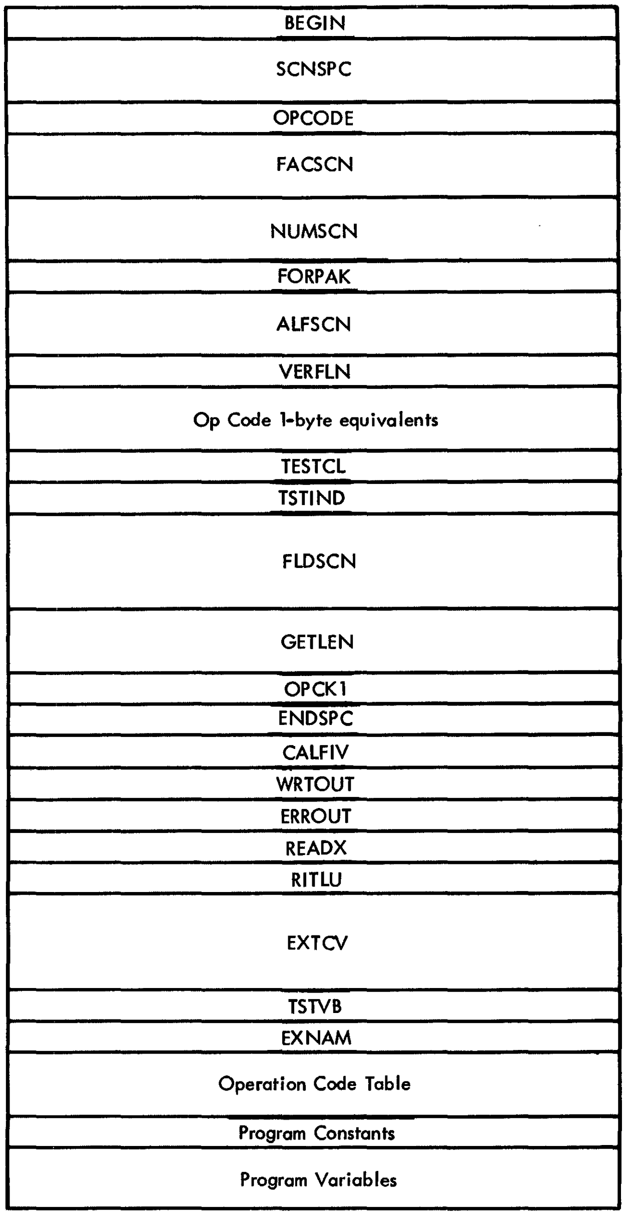


Figure 14. Enter Phase 4 Storage Allocation Map

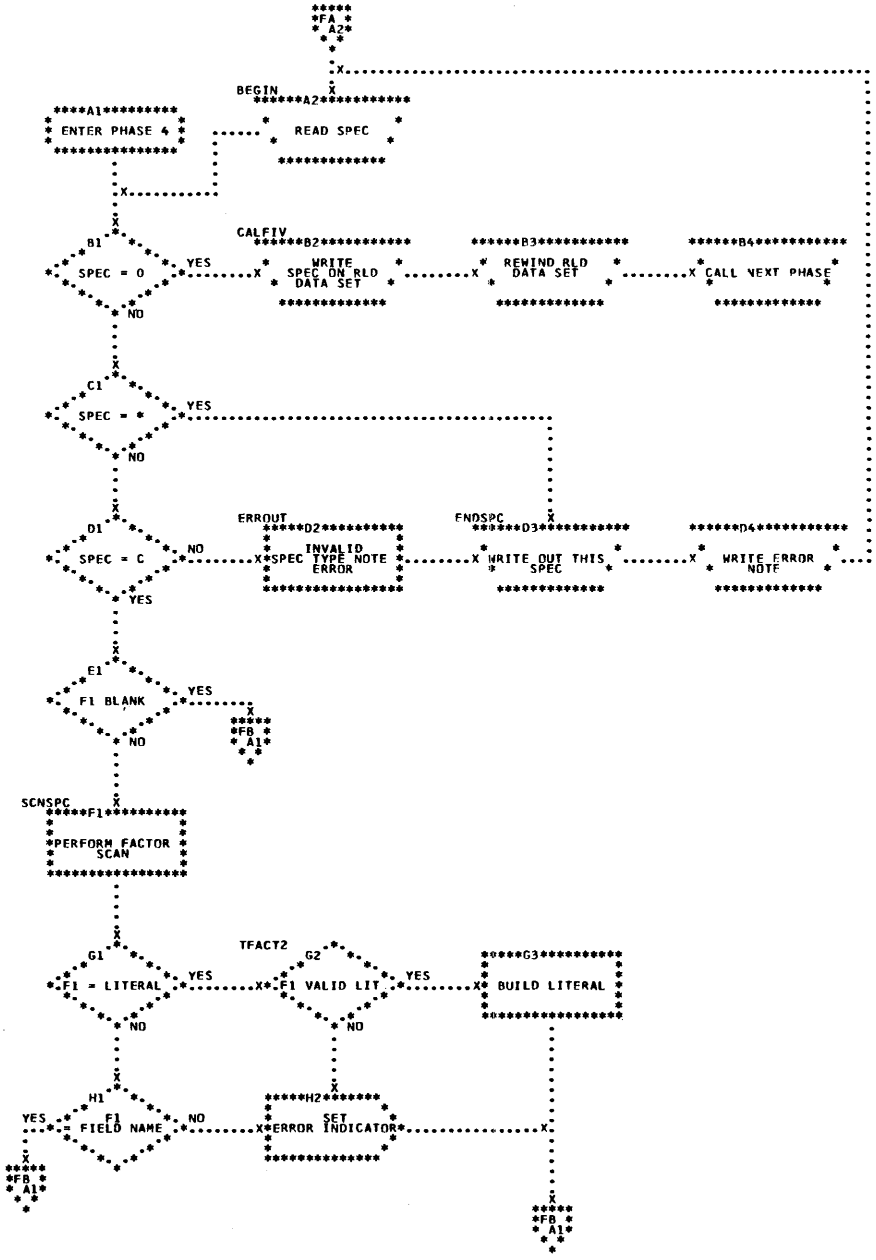


Chart FA. Enter Phase 4

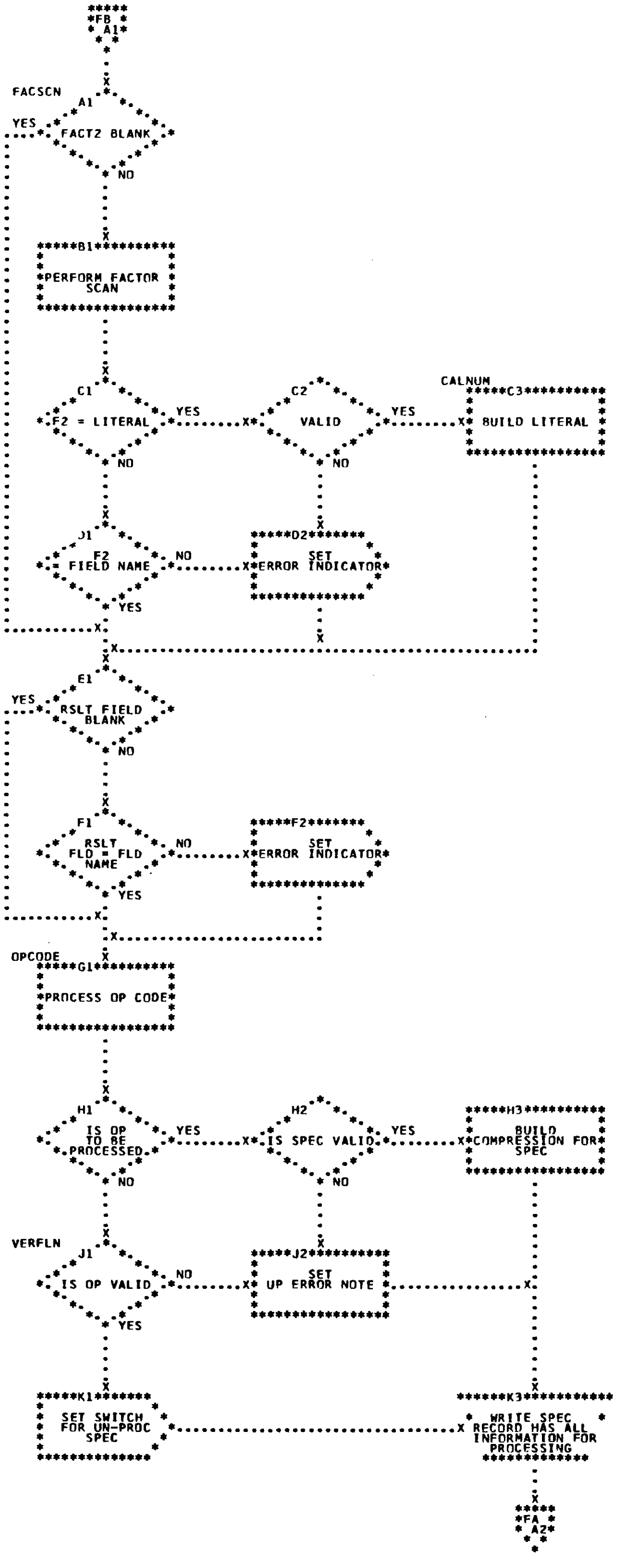


Chart FB. Enter Phase 4

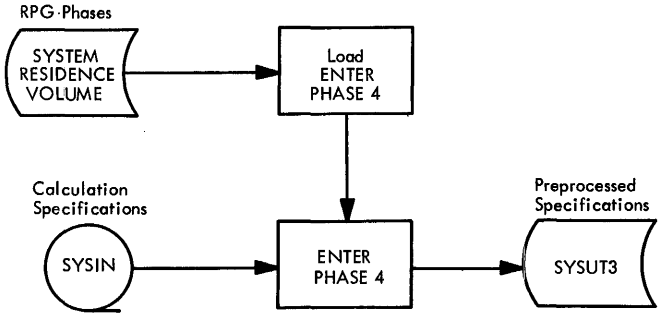


Figure 15. Enter Phase 4 Input/Output Flow

Table 4. Operation Code 1-Byte Equivalence Table

Operation Code	Hexadecimal Equivalent	Operation Code	Hexadecimal Equivalent
ADD	FA	TESTZ	DC
Z-ADD	F8	GOTO	C4
SUB	FB	TAG	E8
Z-SUB	F9	EXIT	C5
MULT	FC	RLABL	E9
DIV	FD	ULABL	EA
MVR	FE	SETON	C1
MOVE	D2	SETOF	C3
MOVEL	D6	LOKUP	C8
MLHZO	D7	RPGCV	C2
MHLZO	D5	EXTCV	C7
MHHZO	D1	ERPGC	C6
MLLZO	D3	KEYCV	EB
COMP	DA		

SWITCHES AND INDICATORS

DECSW

X'FF' Set if factor scanned in FACSCN starts with a decimal point

X'00' Cleared

ERSW

X'FF' Set if a required field name is missing on operation codes EXTCV or RPGCV

X'00' Cleared

FDWS

X'FF' Indicates field length entry is not valid numeric

X'00' Indicates field length entry is not blank

FSW

X'00' Cleared at start of table search

X'FF' Set if find made

MINSW

X'FF' Set if factor scanned in FACSCN starts with minus sign

X'00' Cleared

PAKSW

X'00' Indicates factor scanned in FACSN starts with plus sign or apostrophe

X'FF' Set when character from specification has been moved to PAKAR

TOTSW

X'FF' Set if control level specified

X'00' Cleared

USESW

X'FF' Indicates factor field name

X'00' Indicates result field name

FLDMSK

xx11xx1x	All other field names (normal)
xx111x11	Result field of a KEYCV following an EXTCV
11x11111	Factor 1 of EXTCV, RPGCV
111x1111	Factor 2 of EXTCV, EXIT
x111xx11	Factor 2, result field of a LOKUP
xx11xx11	RLABL
111111x1	TAG, GOTO
x111111x	ULABL

Table 5. Summary of Operation Specifications

O = Optional R = Required b = blank

Operation	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust	Resulting Indicators
Add	O	O	R	ADD	R	R	O	O	O	O
Zero and Add	O	O	b	Z-ADD	R	R	O	O	O	O
Subtract	O	O	R	SUB	R	R	O	O	O	O
Zero and Subtract	O	O	b	Z-SUB	R	R	O	O	O	O
Multiply	O	O	R	MULT	R	R	O	O	O	O
Divide	O	O	R	DIV	R	R	O	O	O	O
Move Remainder	O	O	b	MVR	b	R	O	O	b	b
Move	O	O	b	MOVE	R	R	O	b	b	b
Move Left	O	O	b	MOVEL	R	R	O	b	b	b
Move High-to-Low Zone	O	O	b	MHLZO	R	R	O	b	b	b
Move Low-to-High Zone	O	O	b	MLHZO	R	R	O	b	b	b
Move High-to-High Zone	O	O	b	MHHZO	R	R	O	b	b	b
Move Low-to-Low Zone	O	O	b	MLLZO	R	R	O	b	b	b
Compare	O	O	R	COMP	R	b	b	b	b	R
Test Zone	O	O	b	TESTZ	b	R	R	R	b	R
Exit to a Subroutine	O	O	b	EXIT	R	b	b	b	b	b
RPG Label	O	b	b	RLABL	b	R	O	O	b	b
User's Label	O	b	b	ULABL	b	R	R	R	b	b
Branching or GOTO	O	O	b	GOTO	R	b	b	b	b	b
Providing a Label for GOTO	O	b	R	TAG	b	b	b	b	b	b
Set Indicators ON	O	O	b	SETON	b	b	b	b	b	R
Set Indicators OFF	O	O	b	SETOF	b	b	b	b	b	R
Table Lookup	O	O	R	LOKUP	R	O	O	O	b	R
RPG Conversion	O	b	R	RPGCV	b	R	R	R	b	b
End of RPG Conversion	O	b	b	ERPGC	b	b	b	b	b	b
External Conversion Routine	O	b	R	EXTCV	R	R	R	R	b	b
Record Key	O	b	b	KEYCV	b	R	O	O	b	b

FACTSW

lxxxxxxx	Factor l if field name
xlxxxxxx	Factor l is a literal
xxxxlxxx	Factor 2 is field name
xxxxxlxx	Factor 2 is a literal
xxxxxxxl	Result field name

ERBYT

lxxxxxxx	Factor l not a field name
xlxxxxxx	Decimal positions invalid
xxlxxxxx	Field length not specified or invalid
xxxlxxxx	Invalid operation code
xxxxlxxx	Required factor l missing
xxxxxlxx	Factor 2 missing or invalid
xxxxxxlx	Result field not factor name
xxxxxxxl	Specification types not 0,*, or C

ERBYT+l

lxxxxxxx	First column of resulting indicator not blank or N
xxlxxxxx	Factor name used improperly
xxxlxxxx	Field length exceeds 256 bytes
xxxxlxxx	No control level entry but TOTSW is ON
xxxxxlxx	First position of control level not L or invalid
xxxxxxlx	Invalid resulting indicator entry
xxxxxxxl	RPGCV or EXTCV result field length invalid

IBYT

lxxxxxxx	Compressed specification in record
xlxxxxxx	Invalid specification
xxlxxxxx	Preprocessed specification
xxxxxlxx	Compressed specification and it caused name table overflow

ABYT

Refer to alpha byte in Appendix E.

MAIN ROUTINES

BEGIN

- Rewinds the work data set
- Moves a specification to the output buffer
- Determines specification type
- Checks control level indicator usage for calculation specifications

SCNSPC

- Determines which field (factor l, factor 2, or result field) will be scanned by FACSCN

OPCODE

- Compares the operation entry to the list of valid operation codes in table OPSYM
- Branches to either an error routine or an operation code handling routine

CALFIV

- Calls in the next phase

EXTCV

- Processes EXTCV, RPGCV, KEYCV, RLABL, and ULABL operation codes
- Places the compressed specification in the output buffer

EXNAM

- Handles EXIT, TAG, and GOTO operation codes

SUBROUTINES

FACSCN

- Scans factor and result field entries
- Branches to the appropriate routine to handle an alpha literal, numeric literal, or field name
- Determines which branch to take by the first position of the field

NUMSCN

- Scans either factor l or factor 2 for a valid numeric literal
- Packs the literal into the work area called WKCOMP

ALFSCN

- Builds the alpha literal entry for the literal table

VERFLN

- Scans either the factor 1 or factor 2 or the result field entries for a valid six-character (maximum) name

TESTCL

- Verifies the control level entry

TSTIND

- Checks the validity of any indicators in the 9 byte indicator entry

FLDSCN

- Scans field names and builds the field name table
- Verifies field name usage (factor or result field entry usage sometimes not allowed)

GETLEN

- Checks the validity of the field length entry and converts it to binary
- Checks the decimal positions entry for proper usage

ENDSPC

- Initiates a branch to write a specification (WRTOUT)
- Reads a specification (READX)
- Continues (START)

WRTOUT

- Writes a record on SYSUT3

ERROUT

- Sets error indication
- Branches to ENDSPC

READX

- Initiates reading a specification

RITLU

- Searches the resulting indicator table
- Sets FSW to X'FF' if a find is made

TSTVB

- Checks for an IDCV or a KEYCV operation code following an RPGCV or EXTCV

FORPAK

- Moves information from the specification into an intermediate storage location in preparation for packing

OUTPUT RECORD FORMAT (RLDBUF)

Compressed Specification

RLDBUF = 132 bytes	
RLDBUF to RLDBUF+79	Original specification
RLDBUF+80	IBYT
RLDBUF+81	SAVEOP - 1 byte operation code (present only with IBYT of X'80', X'88',or X'20')
RLDBUF+120	ERBYT
RLDBUF+121	ERBYT+1
RLDBUF+84	Length - 1 of compression (used with IBYT of X'80', or X'88')
RLDBUF+85 to RLDBUF+131	Available for compression

Preprocessed Specification

RLDBUF = 132 bytes	
RLDBUF to RLDBUF+79	Original specification
RLDBUF+80	IBYT
RLDBUF+81	SAVEOP 1-byte operation code
RLDBUF+82	FACTSW
RLDBUF+83 to RLDBUF+94	Factor 1 literal* as follows: 1 byte; length of entry +2 (hexadecimal) 1 byte; decimal positions (unpacked number) 1-8 bytes; literal 1st byte after literal; number of digits in original entry minus 1 2nd byte after literal; decimal positions

*If there is no factor 1 and/or factor 2 literal, data remaining from previous specifications will be present except in byte RLDBUF+86 (alpha byte), which is initialized to X'00'.

	3rd byte after literal; padding X'FF' for numeric literal, X'F0' for alphameric literal
RLDBUF+95 to RLDBUF+106	Factor 2 literal*, as above
RLDBUF+107 to RLDBUF+131	X'00' or information

remaining from previous specifications

*If there is no factor 1 and/or factor 2 literal, data remaining from previous specifications will be present except in byte RLDBUF+86 (alpha byte), which is initialized to X'00'.

ENTER PHASE 5

INTRODUCTION

Enter Phase 5 reads, lists, diagnoses, and compresses the calculation specifications prepared in Enter Phase 4. This phase uses data from the preprocessed calculation specifications to continue to search and build the field name table, resulting indicator table and literal table.

The data record from the SYSUT3 can have one of five formats

1. Specification with compression and error notes (if any)
2. Specification with an error note
3. Specification with literal entries
4. Output-format specification
5. Comments

Those specifications with compression are assigned a number in sequence and written on SYSPRINT. If there are errors, the error note numbers are printed immediately following the specification. The compression built in Enter Phase 4 is entered into the compression area.

Calculation specifications that have been diagnosed as invalid are printed without a sequence number. The error note assigned to the specification is printed immediately following the specification.

Those specifications that were preprocessed in Enter Phase 4 are verified and compressed in this phase. The specification is examined to determine if it meets the requirements of the operation code (e.g., SETON requires that resulting indicators be specified and valid). If it does, the specification is considered valid and is assigned a sequence number, printed, and compressed.

If the specification is invalid, it is printed without a sequence number, and the error note associated with the specification is printed immediately following.

In order to retain as much source data in storage as possible, all unnecessary data is deleted from the calculation specifications, and the resulting data is placed in the compression area. Appendix E illustrates the format of the compressed specification.

Figure 16 and Charts FF and FG illustrate the organization and operation of Enter Phase 5.

Figure 17 illustrates the input/output flow for Enter Phase 5.

LOGIC

The specification is checked to determine its type, i.e., output, comment, or calculation. Only calculation specifications are processed by Enter Phase 5.

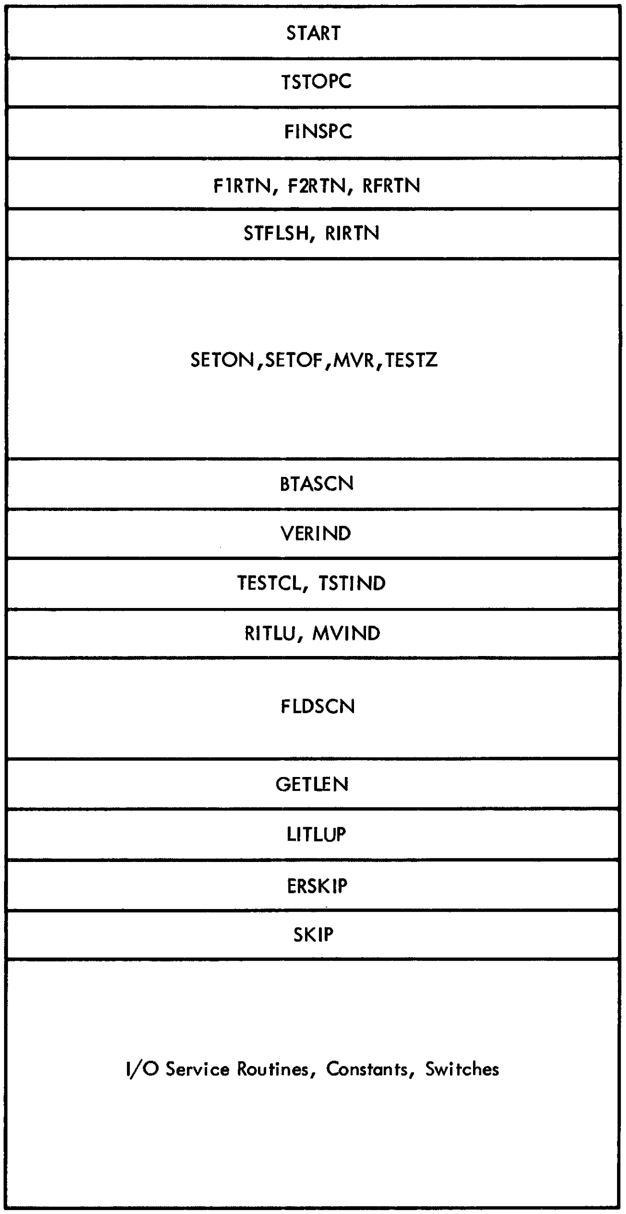


Figure 16. Enter Phase 5 Storage Allocation Map

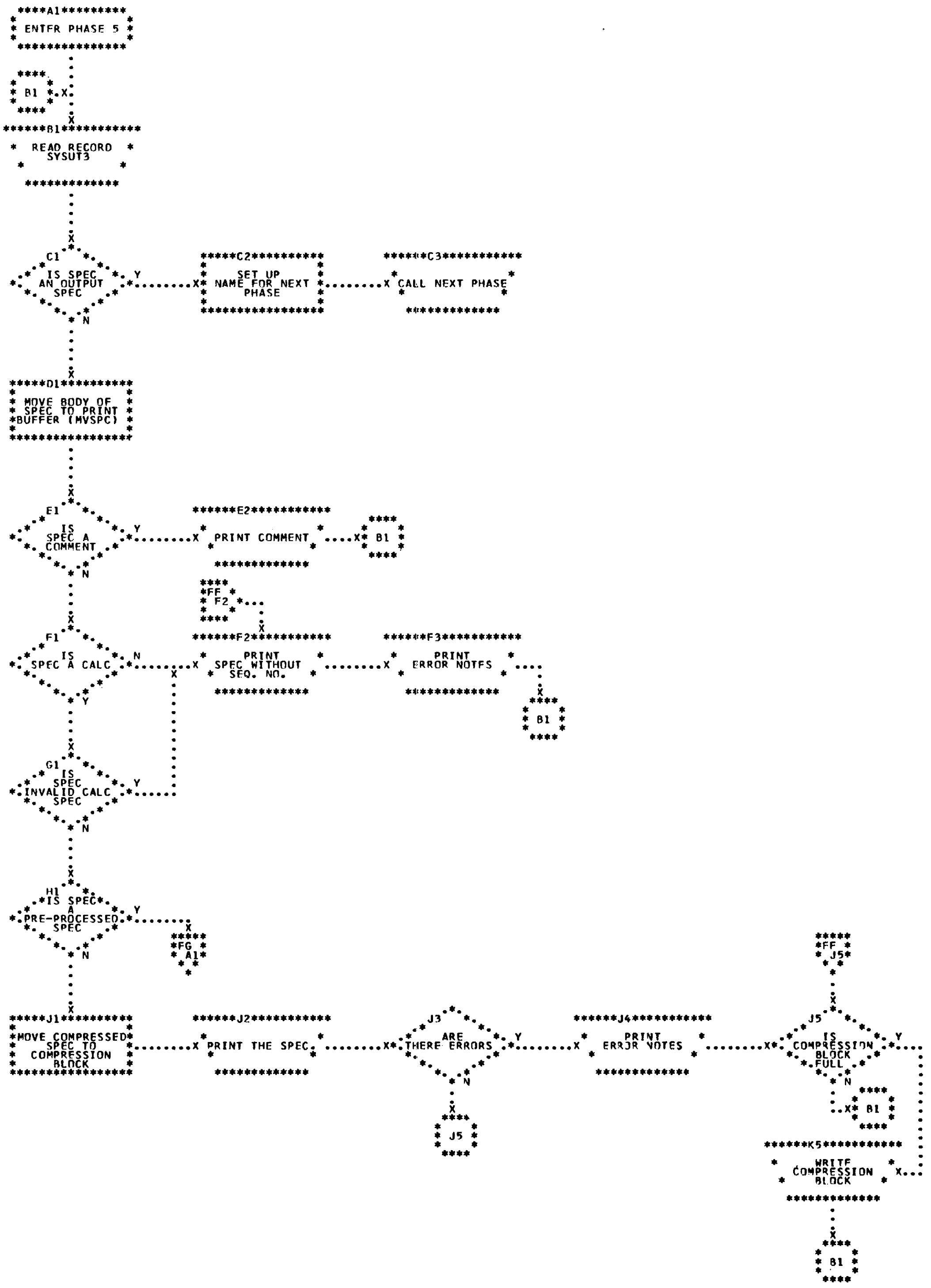


Chart FF. Enter Phase 5

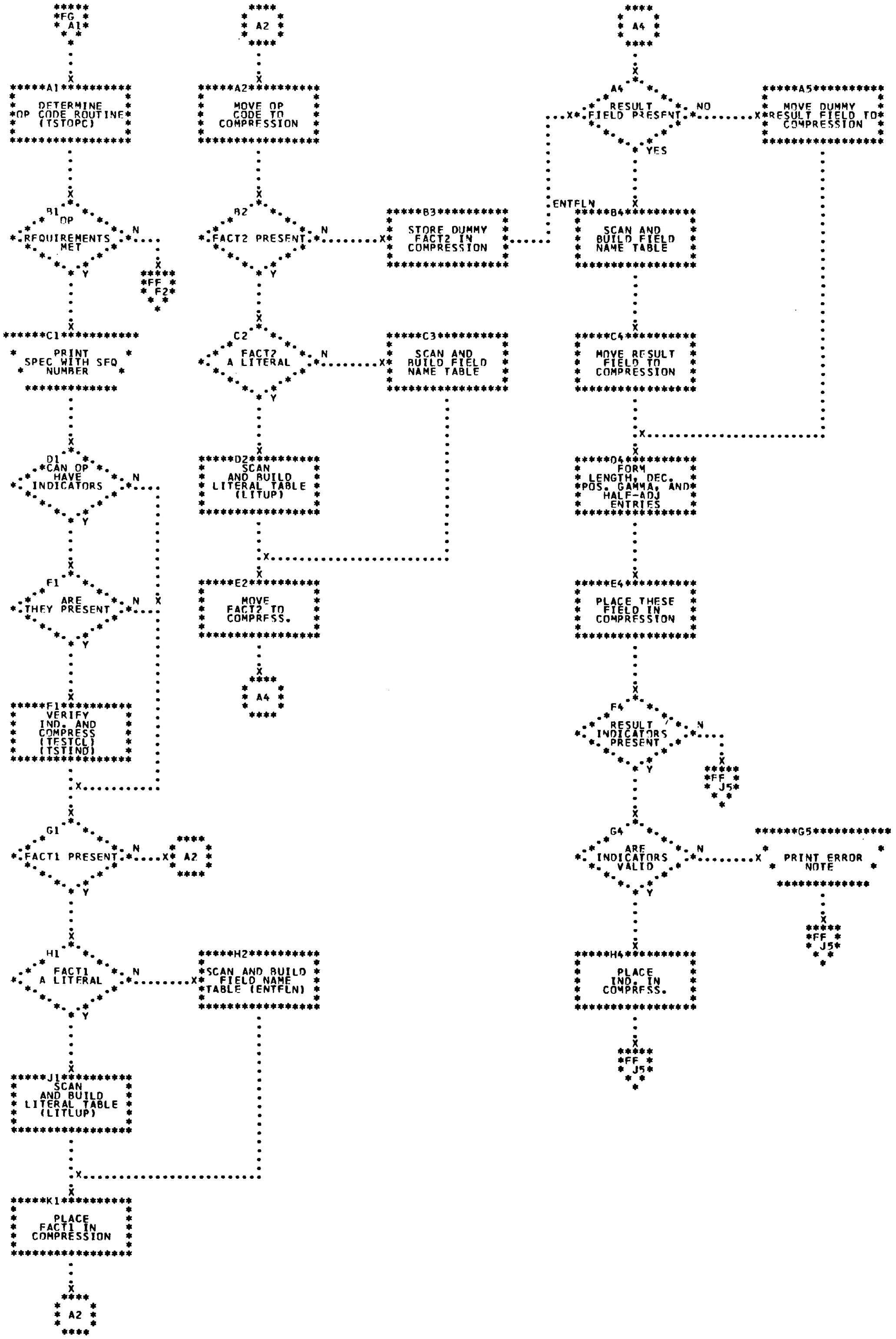

FG
A1
A1
DETERMINE OP CODE ROUTINE (TSTOPC)
B1
OP REQUIREMENTS MET
N
Y
FF F2
C1
PRINT SPEC WITH SEQ NUMBER
D1
CAN OP HAVE INDICATORS
N
Y
E1
ARE THEY PRESENT
N
Y
F1
VERIFY IND. AND COMPRESS (TESTCL) (TSTIND)
G1
FACT1 PRESENT
N
Y
A2
H1
FACT1 A LITERAL
N
Y
J1
SCAN AND BUILD LITERAL TABLE (LITLUP)
K1
PLACE FACT1 IN COMPRESSION
A2
A2
A2
MOVE OP CODE TO COMPRESSION
B2
FACT2 PRESENT
N
Y
C2
FACT2 A LITERAL
N
Y
D2
SCAN AND BUILD LITERAL TABLE (LITUP)
E2
MOVE FACT2 TO COMPRESS.
A4
H2
SCAN AND BUILD FIELD NAME TABLE (ENTFLN)
B3
STORE DUMMY FACT2 IN COMPRESSION
C3
SCAN AND BUILD FIELD NAME TABLE
A4
A4
RESULT FIELD PRESENT
NO
YES
A5
MOVE DUMMY RESULT FIELD TO COMPRESSION
ENTFLN
B4
SCAN AND BUILD FIELD NAME TABLE
C4
MOVE RESULT FIELD TO COMPRESSION
D4
FORM LENGTH, DEC. POS. GAMMA, AND HALF-ADJ ENTRIES
E4
PLACE THESE FIELD IN COMPRESSION
F4
RESULT INDICATORS PRESENT
N
Y
FF J5
G4
ARE INDICATORS VALID
N
Y
G5
PRINT ERROR NOTE
FF J5
H4
PLACE IND. IN COMPRESS.
FF J5

Chart FG. Enter Phase 5

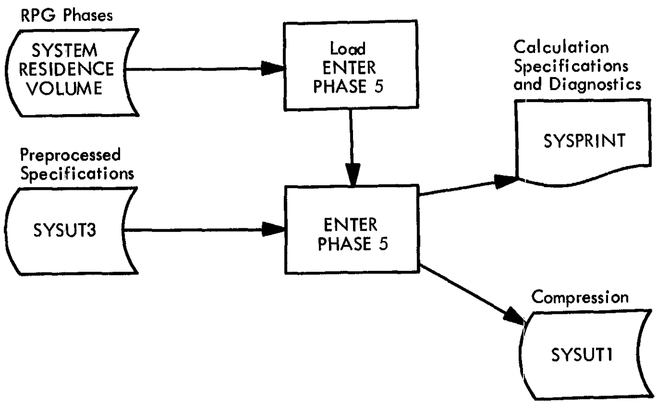


Figure 17. Enter Phase 5 Input/Output Flow

The information byte (IBYT) is checked to determine whether this is a preprocessed specification (not compressed in Enter Phase 4).

If it is not a preprocessed specification this means that it already contains the calculation specification compression. If no errors have been noted, the specification is assigned a sequence number, printed, and entered in the compression and a new specification is read.

If IBYT indicates a preprocessed specification, proper usage of factor and result field are checked. If no error is found, a sequence number is assigned, the specification is printed, the calculation specification is entered in the compression, and a new specification is read.

SWITCHES AND INDICATORS

ERRSW

X'FF' Invalid specification

ERIND

X'FF' Invalid resulting indicator

FLDERR

X'FF' Multidefined field or improper field name usage

FACTSW, USESW, FSW, FDSW, ERSW, ABYT, ERBYT, IBYT, FLDMSK are carried over from previous phase (see Enter Phase 4 Switches and Indicators).

MAIN ROUTINES

START

- Rewinds the work data set
- Reads a specification
- Determines the type
- Branches to the appropriate routine

TSTOPC

- Checks the operation code of a preprocessed specification for validity
- Causes a branch to the proper routine to handle the particular operation code

FINSPC

- Finalizes the specification compression
- Checks for a full block
- Branches to read a record

SUBROUTINES

F1RTN, F2RTN

- Checks the presence of factor 1, 2
- Verifies the usage of the field name

RFRTN

- Tests the result field for a field name
- Verifies the usage of the field name

STFLSH

- Sets up scan of field name table
- Verifies the usage of factor 1, factor 2, or result field entries

RIRTN

- Scans the resulting indicators entries for the presence of valid indicators. The specifications for SETON, SETOF, COMP, TESTZ, and LOKUP cannot be accepted unless this requirement is met.

BTASCN

- Scans the resulting indicators entries for validity and also for valid combinations, e.g., greater than and less than cannot be specified together.

VERIND

- Checks to make sure that L0 or 00 are not specified as resulting indicators
- Verifies that the resulting indicator entries are each one of those listed in the resulting indicator table
- Sets the alpha byte in the calculation compression according to which indicators, if any, are present

TESTCL

- Tests for and, if present, verifies a control level specification
- Places the specification in the calculation compression
- Sets the alpha byte accordingly
- Assumes L0 for an invalid specification

TSTIND

- Tests, then verifies any indicators entries specified in positions 9-17 of the calculation specifications
- Moves verified indicators into the calculation compression
- Sets the alpha byte accordingly
- Assumes indicator 00 for an invalid indicator entry
- Produces a diagnostic message

RITLU

- Scans the resulting indicator table
- Sets X'FF' in FSW if a matching entry is found

MVIND

- Moves any valid resulting indicators (as specified by alpha byte bits) to the calculation compression

FLDSCN

- Checks field names against the field name table
- Verifies that each field name is being used properly

GETLEN

- Verifies the field length entry
- Converts the entry to binary
- Verifies the decimal positions entry

LITLUP

- Searches and builds the literal table, using the literal entries formed in Enter Phase 4

ERSKIP

- Checks the error bytes associated with the specifications processed in Enter Phase 4
- Branches to an error-handling routine (ERRTNE) if an error bit is found

SKIP

- Prints invalid specifications without a sequence number

INTRODUCTION

Enter Phase 6 reads, sequences, lists, diagnoses, and compresses the output-format specifications and makes certain entries into the literal, resulting indicator, and field name tables. The output-format specifications entries define the characteristics of the records within each output file, i.e., whether printed or punched, when produced, etc., and of the fields within these records, i.e., position in record, type of data, etc.

In order to retain the maximum data in storage, unnecessary information is deleted from the specifications and the result is placed in the compression area. When the compression area is filled, it is written out and a new compression is started. The format of the compressed output-format specifications is shown in Appendix E.

Figure 18 and Chart GA illustrate the organization and operation of Enter Phase 6.

Figure 19 illustrates the input/output flow for Enter Phase 6.

LOGIC

When Enter Phase 6 is called, the first output-format specification has already been read and its type has been determined.

The phase must first determine if the specification is defining a <u>record</u> to be output or a <u>field</u> within the output record defined previously. It must also recognize AND and OR types, which are actually continuations of or new sets of conditions for the preceding record line.

If the specification is a record type, the appropriate diagnostics and compression are performed. Undefined and unreferenced output file names are not detected at this time, because the file name table has been destroyed to make way for the literal table. The AND and OR types are checked to see that they follow record specifications rather than field specifications.

If the specification defines a field in the output record, the column entries are checked and the appropriate diagnostics and compression are produced. The presence of an entry in field name is tested and, if there is one, the field name table is searched to determine if the field has been defined previously. The result of the table search will be one of the following:

1. The field is in the table. The gamma (γ) byte is tested with the proper mask.

   If the result of the test indicates that the field is inappropriate for an output, the reference byte is changed to M (multidefined) regardless of its present contents.

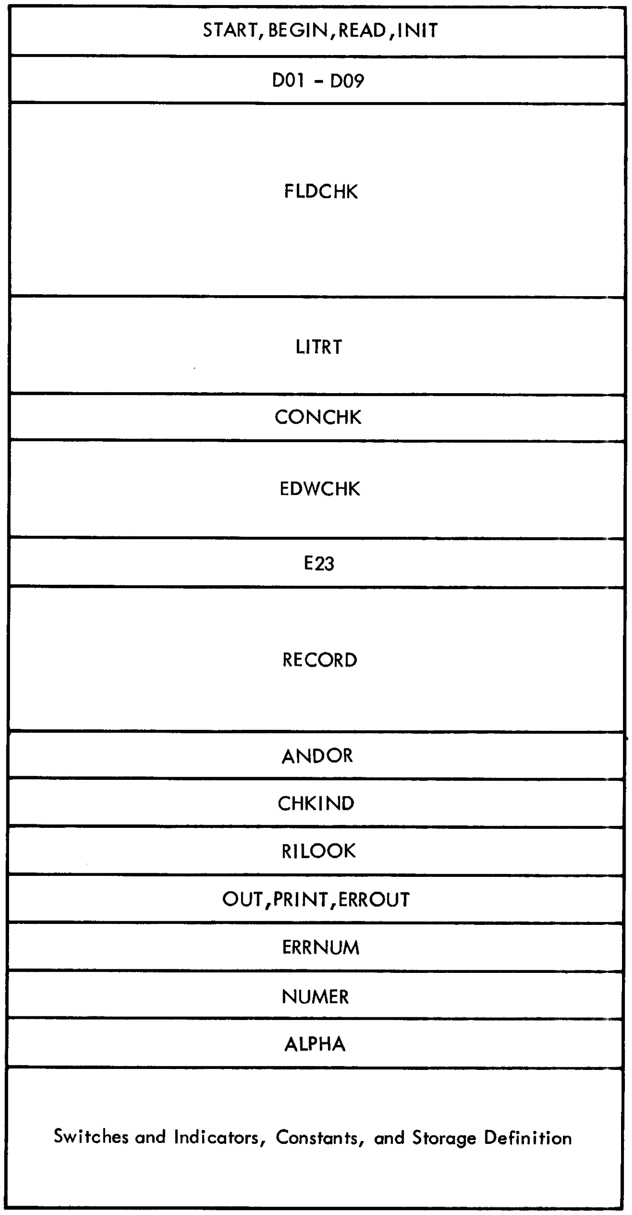


Figure 18. Enter Phase 6 Storage Allocation Map

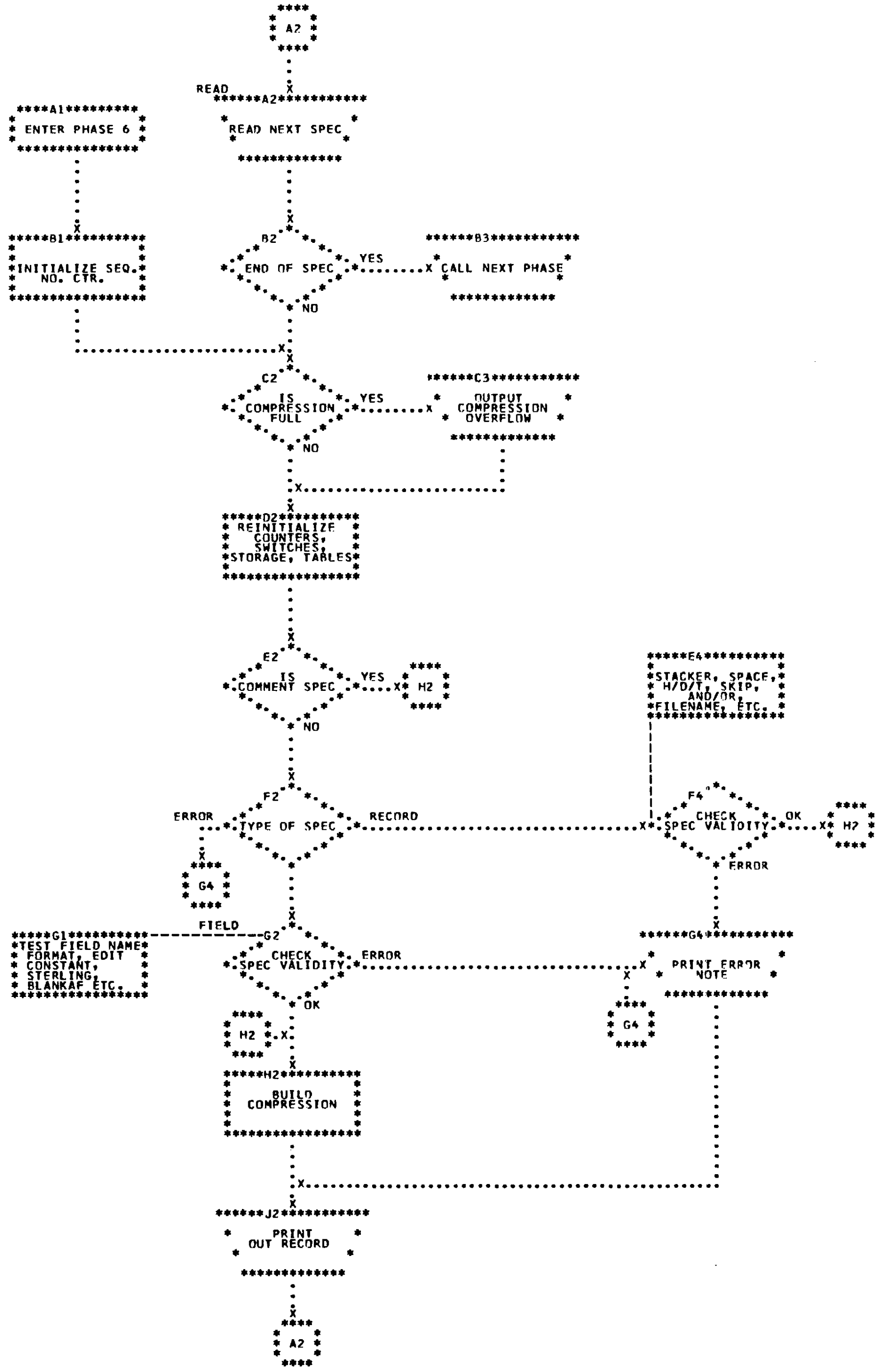


Chart GA. Enter Phase 6

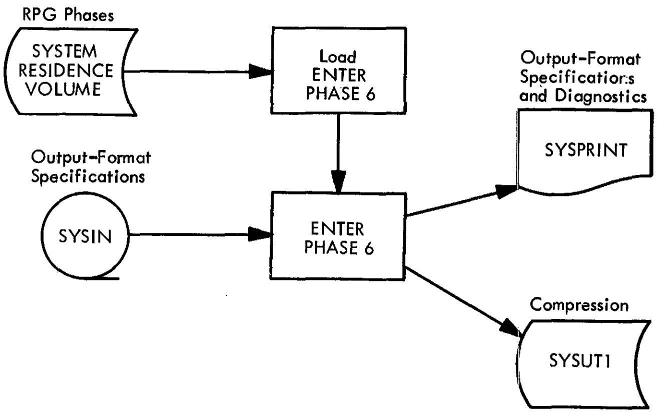


Figure 19. Enter Phase 6 Input/Output Flow

If the test indicates the field is appropriate for an output, then the action depends on whether it is a PAGE(n) field. For a field that is not PAGE(n), the reference byte is checked and changed to R (referenced) if it contains a blank (defined but unreferenced). Otherwise, the reference byte is not changed. If it is a PAGE(n) field, both the reference byte and the decimal position bytes are checked. If the reference byte is blank (defined but unreferenced) or U (referenced but undefined), it is changed to R. The decimal position must be zero. If it is not, then it is changed to zero and the reference byte is changed to M.

2. The field is not in the table, the table has not overflowed, and it is a PAGE(n) field. PAGE(n) fields are entered with a reference byte of R, a length of four (entry of X'03'), a decimal position of zero, and a gamma byte of X'80'. If a PAGE(n) entry causes a table overflow, X'FD' indicator is placed at the end of the table and the specification sequence number causing the overflow is saved in the CIOEX data area. The PAGE(n) entry then will be handled later by Assign Phase 2.
3. The field is not in the table and the table has overflowed. The specification is compressed and processing continues. Any such overflow fields will be handled by Assign Phase 2.
4. The field is not in the table and the table has not overflowed, and it is not a PAGE(n) field. All such fields are not processed but are given a diagnostic noting that the field is undefined.

A field specification may also have an edit word associated with it. If there is an edit word, it is translated into an EDIT instruction pattern and the result treated in the same manner as the constants mentioned below.

If a field type specification has no entry in the field name positions, the specification is tested for the presence of a constant. If no constant is defined, the specification is in error, a diagnostic code is assigned and processing of the entry is terminated.

If there is a constant, the literal table is searched to determine if the constant has been defined previously. If it has been defined previously, no further action is required. If the constant has not been defined and the table is not full, the constant is added, along with eight information bytes (entry length, literal length, type, etc.). If the table is full, the procedure is much the same as for the field name table above. For a detailed description see the literal table in Appendix C.

In addition to the previous testing for record and field type specifications, the output indicator positions of the specification are checked. If an indicator is found, the resulting indicator table is searched. This table is pre-loaded with all valid indicators; thus, overflow can never occur. If the indicator from the specification is found in the table, the table reference byte is changed to indicate the reference. This byte is also checked to see if the indicator has been previously defined. If the indicator is undefined, or if it is invalid, i.e., not found in the table, it is replaced by the L0 indicator. (The actual substitution is done in the assign phases, although the diagnostic code is assigned here.)

If an incorrect entry is encountered in a specification, a valid entry is assumed in its place whenever possible and the processing continued. When an invalid entry that causes the specification to be discarded is found, all other entries on the specification are checked and diagnosed, if possible.

SWITCHES AND INDICATORS

SWITCH - 1 byte

CONSW	xxxxxxx1	Constant edit is a constant
EDITSW	xxxxxx1x	Constant edit is an edit word
PAGESW	xxxxx1xx	Field is a page
OUTSW	xxxx1xxx	Invalid specification
RECDSW	xxx1xxxx	Tells field type specification if preceding specification was a record type specification
ZEROSW	xx1xxxxx	Decimal positions for page field must be changed to zero in the field name table

TABSW - 1 byte

	xxxxxxx1	Page field is to be placed in the field name table
RREFSW	xxxxxx1x	Reference byte is to be placed in the field name table
FDSW	xxxx1xxx	A page field has caused the field name table to overflow
BLAFSW	xx11xxxx	Blank after field
RITSW	11xxxxxx	Second indicator blank
	1xxxxxxx	First indicator blank

EDSW - 1 byte

xxxxxxx1	Zero suppress
xxxxx1xx	Floating dollar
xxxx1xxx	Zero switch
xxx1xxxx	Negative (credit) switch
xx1xxxxx	Fill switch

RECERR - 1 byte

RECERR	xxxxxxx1	Invalid record
NAMERR	xxxxxx1x	Invalid or missing filename

EDBYT

xxxxxxx1	CR (credit) symbol
xxxxxx1x	Minus sign
xxxxx1xx	Floating dollar
xxxx1xxx	Fixed dollar

INDBYT

x1xxxxx1	Valid first resulting indicator
1xxxxx1x	Valid second resulting indicator
11xxxx11	Valid third resulting indicator
xxx1xxxx	Overflow indicator

SPCBYT

xxxxxxx1	Packed
xxxxxx1x	Sterling
xxxxx1xx	OR record or literal
xxxx1xxx	AND record or no blank-after
xxx1xxxx	Zero suppress
xx1xxxxx	Field name or T (total) specification
x1xxxxxx	D (detail) specification
1xxxxxxx	H (header) specification

MAIN ROUTINES

D01 - D09

- Determine the type of specification, i.e., comment, record, field, edit word, constant, page, or invalid specification

FLDCHK

- Diagnoses and compresses all field type specifications

LITRT

- Diagnoses literals (constant or edit word entries)

CONCHK

- Moves constant entry characters to compression

EDWCHK

- Analyzes edit word entry and places in compression

RECORD

- Diagnoses and compresses all record type specifications including AND or OR

ANDOR

- Diagnoses AND and part of OR specifications

SUBROUTINES

CHKIND

- Checks presence of output indicators

- Sets bits in compression to indicate how many indicators are present

RILOOK

- Diagnoses output indicator entries

- Puts the entries in the compression

ERRNUM

- Obtains the address of the next empty slot in error table

NUMER

- Checks for a valid numeric digit or high order blanks

ALPHA

- Checks for a valid alphabetic character or $, #, or @

INTERMEDIATE PHASE

INTRODUCTION

This phase is entered and executed only if the user has specified both of the compiler options LOAD and DECK. The functions of this phase are

1. Close SYSIN
2. Overlay the DCB (in Resident Phase) for SYSIN with the DCB for SYSPUNCH
3. Open SYSPUNCH

Figure 20 and Chart GM illustrate the organization and operation of Intermediate Phase. Figure 21 illustrates the input/output flow for the phase.

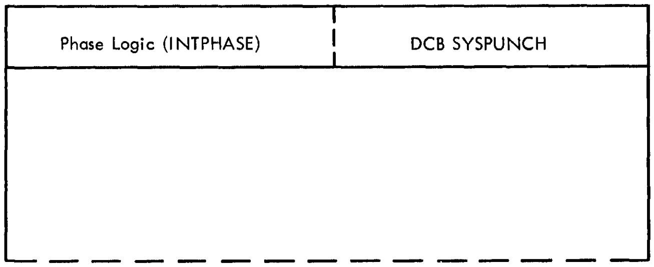


Figure 20. Intermediate Phase Storage Allocation Map

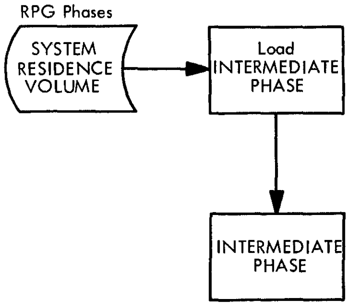


Figure 21. Intermediate Phase Input/Output Flow

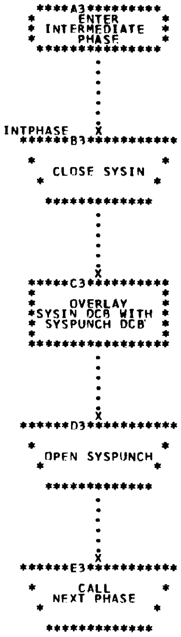


Chart GM. Intermediate Phase

INTRODUCTION

Assign Phase 1 is the first of two phases designed to compute and assign addresses for table entries. Under normal conditions the first is the only assign phase executed. Assign Phase 2 is executed only if the field name or literal tables overflow.

The functions of Assign Phase 1 are

1. Put out the program card to SYSGO/ SYSPUNCH
2. Compute and assign the address for each entry in the resulting indicator table
3. Compute and assign the address for each entry in the field name table
4. Compute and assign the address for each entry in the literal table
5. Put out text cards from the entries in the three tables
6. Print a symbol table from the entries in the three tables
7. Enter into the compression records the addresses from the three tables
8. Put out ESD card images for entry type and external type table entries
9. Put out RLD entries for external type field name entries
10. Print diagnostic codes for multidefined, undefined, and unreferenced field names and resulting indicators

Figure 22 and Chart HA illustrate the organization and operation of Assign Phase 1.

Figure 23 illustrates the input/output flow for Assign Phase 1.

LOGIC

The addresses that are assigned to the table entries are developed in a full word field in the CIOEX data area. This phase increments the counter by the appropriate amount as each address is assigned.

Each resulting indicator table entry requires one byte of storage and each is assigned an address. The assigned address is placed in bytes 4 and 5 of the table entry (Appendix D).

The number of bytes required for each entry in the field name table is determined by three factors.

1. The field name type (byte 13, Appendix B)
2. The length of the field (byte 11)
3. The type of data to be contained in the field (byte 12)

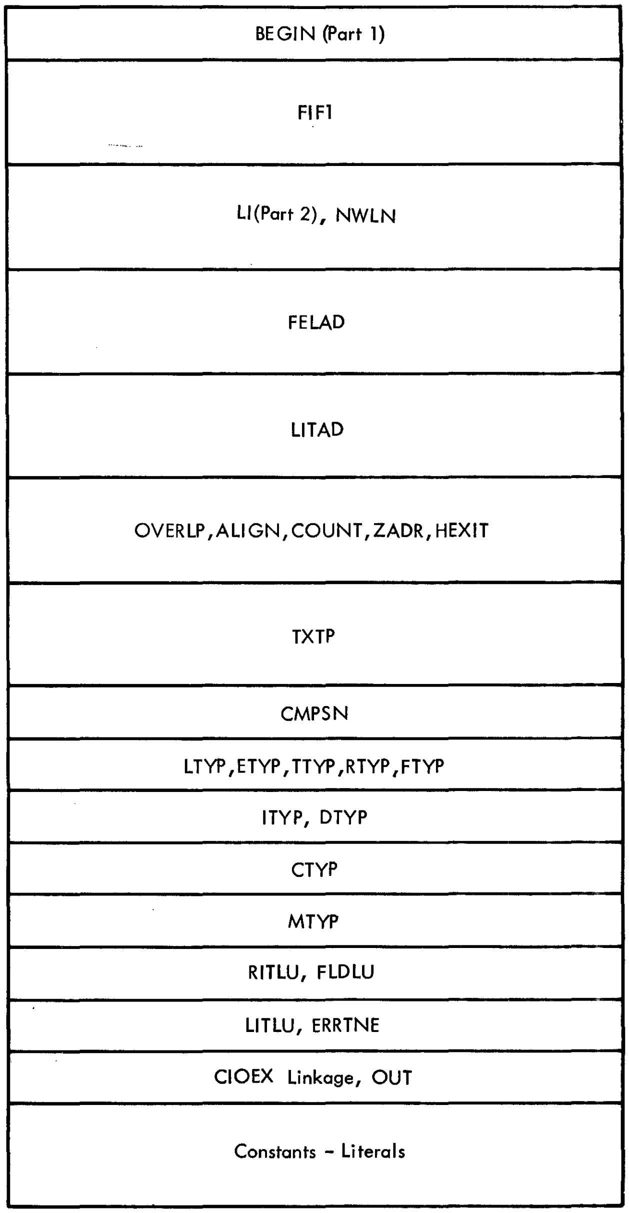


Figure 22. Assign Phase 1 Storage Allocation Map

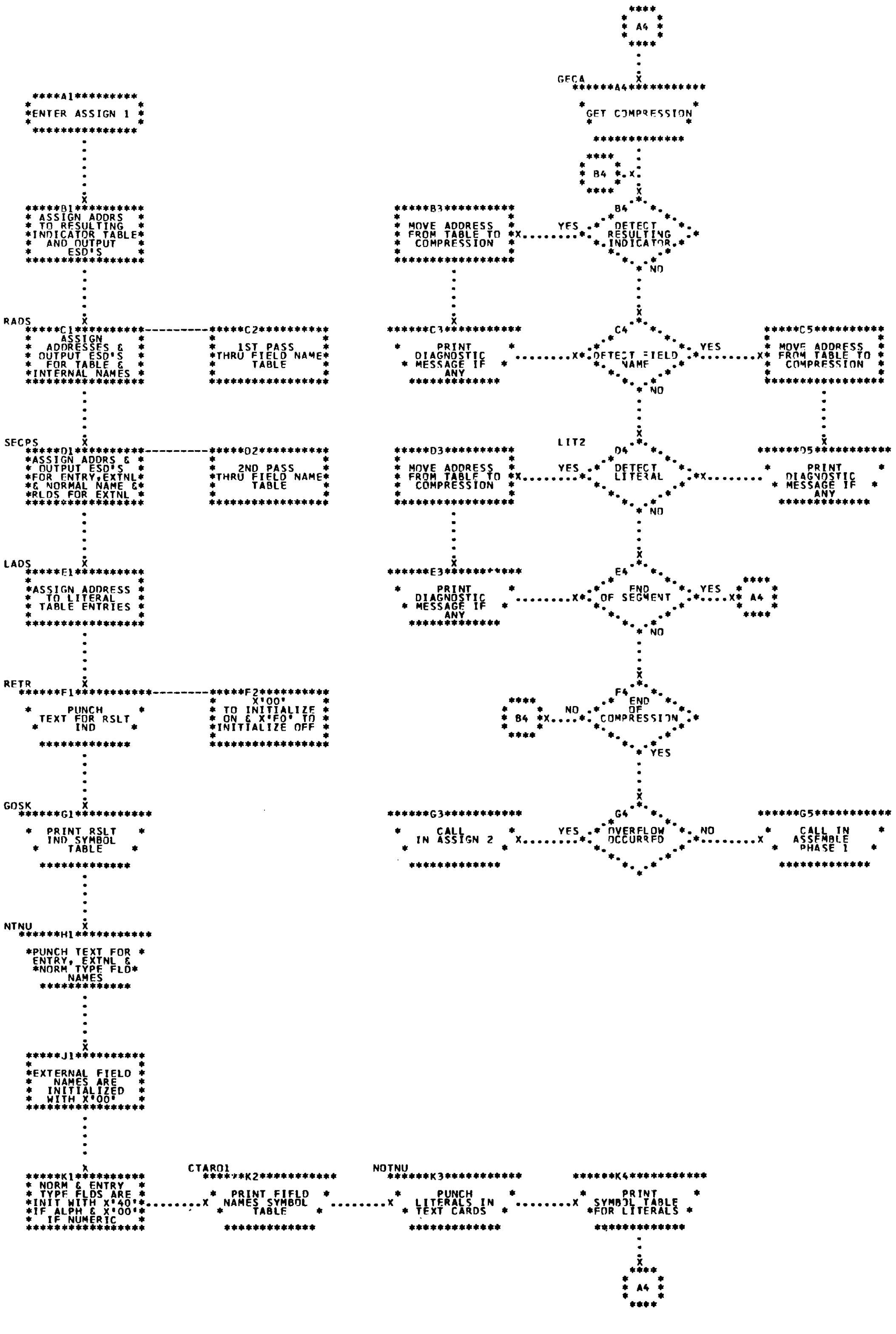

A1 ENTER ASSIGN 1
B1 ASSIGN ADDRS TO RESULTING INDICATOR TABLE AND OUTPUT ESD'S
RADS C1 ASSIGN ADDRESSES & OUTPUT ESD'S FOR TABLE & INTERNAL NAMES
C2 1ST PASS THRU FIELD NAME TABLE
SECPS D1 ASSIGN ADDRS & OUTPUT ESD'S FOR ENTRY,EXTNL & NORMAL NAME & RLDS FOR EXTNL
D2 2ND PASS THRU FIELD NAME TABLE
LADS E1 ASSIGN ADDRESS TO LITERAL TABLE ENTRIES
RETR F1 PUNCH TEXT FOR RSLT IND
F2 X'00' TO INITIALIZE ON & X'F0' TO INITIALIZE OFF
GOSK G1 PRINT RSLT IND SYMBOL TABLE
NTNU H1 PUNCH TEXT FOR ENTRY, EXTNL & NORM TYPE FLD NAMES
J1 EXTERNAL FIELD NAMES ARE INITIALIZED WITH X'00'
K1 NORM & ENTRY TYPE FLDS ARE INIT WITH X'40' IF ALPH & X'00' IF NUMERIC
CTAR01 K2 PRINT FIELD NAMES SYMBOL TABLE
NOTNU K3 PUNCH LITERALS IN TEXT CARDS
K4 PRINT SYMBOL TABLE FOR LITERALS
A4
A4
GECA A4 GET COMPRESSION
B4
B4 DETECT RESULTING INDICATOR
YES
NO
B3 MOVE ADDRESS FROM TABLE TO COMPRESSION
C4 DETECT FIELD NAME
YES
NO
C3 PRINT DIAGNOSTIC MESSAGE IF ANY
C5 MOVE ADDRESS FROM TABLE TO COMPRESSION
LIT2 D4 DETECT LITERAL
YES
NO
D3 MOVE ADDRESS FROM TABLE TO COMPRESSION
D5 PRINT DIAGNOSTIC MESSAGE IF ANY
E4 END OF SEGMENT
YES
NO
A4
E3 PRINT DIAGNOSTIC MESSAGE IF ANY
F4 END OF COMPRESSION
NO
YES
B4
G4 OVERFLOW OCCURRED
YES
NO
G3 CALL IN ASSIGN 2
G5 CALL IN ASSEMBLE PHASE 1

Chart HA. Assign Phase 1

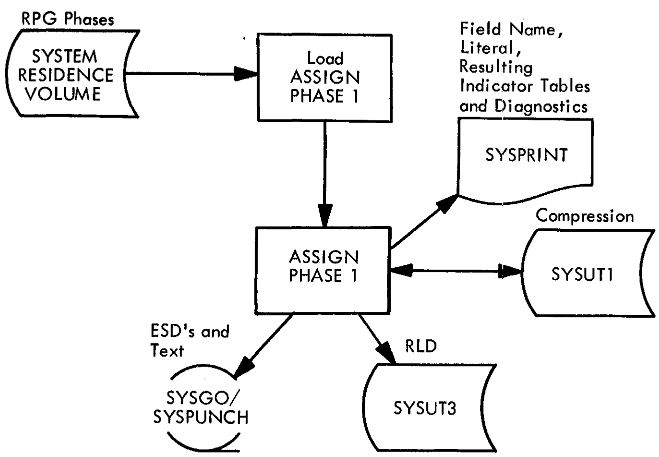


Figure 23. Assign Phase 1 Input/Output Flow

The number of bytes required for each entry is determined as follows:

Field Name Type	Number of Bytes If Alphameric	Number of Bytes If Numeric
Table	16 + L + 1	$16 + \left[\frac{L + 1}{2}\right] + 1$
Internal	4	4
External	4	4
Entry	L + 1	$\left[\frac{L + 1}{2}\right] + 1$
Normal	L + 1	$\left[\frac{L + 1}{2}\right] + 1$

where x =(L + 1)/2 (L is the length of the field; where [x] is the greatest integer less than or equal to x.

Two passes are made through the field name table. On the first pass addresses are assigned to <u>table</u> and <u>internal</u> type field names. On the second pass, addresses are assigned to the remaining types (external, entry and normal). Addresses for table, internal and external type field names are full word aligned.

The assigned address is placed in bytes 8 and 9 of the table entry (Appendix B). The overflow indicator is placed in byte 13.

The number of bytes required for each entry in the literal table is determined by the literal type. For an alphameric literal, the number of bytes required is equal to one plus the length contained in the length byte of the entry (byte 1, Appendix C). For a numeric literal, the number of bytes required is equal to [(L + 1)/2] + 1 where L is the length of the literal in bytes.

If no literal or field name table overflow occurs during this phase, Assemble Phase 1 is called. Otherwise, Assign Phase 2 is called to duplicate the table-building function of the enter phases and the address assigning function of Assign Phase 1.

SWITCHES AND INDICATORS

<u>PASID</u>

X'B0' Second pass of field name table

<u>LTKEY</u>

X'01' Calculation specification compression record

<u>KEY(EQU LTKEY)</u>

X'40' More text to be put out

MAIN ROUTINES

<u>BEGIN (Part I)</u>

- Initializes registers and address counter for the phase
- Puts out the program card image
- Puts out an entry type ESD card image for each entry-type resulting indicator table entry
- Places the computed machine address with each entry in the resulting indicator

<u>FIF1</u>

- Makes two passes through the field name table
- Assigns addresses to table and internal type field name entries on the first pass
- Assigns addresses to external, entry and normal type field name entries on the second pass
- Puts out an ESD card image for each entry and external type field name entry in the table
- Puts out an RLD entry (Appendix F) for each external type field name entry in the table

LI

- Assigns a computed address to each entry in the literal table
- Branches to OVERLP to be sure that no part of the entry is more than 4096 bytes from the base address
- Saves the ESD count for use by Assign Phase 2

NWLN (Part II)

- Saves the overflow count for use by Assign Phase 2
- Puts out the text card images for the entries in the resulting indicator table
- Prints the symbol table at the same time

FELAD

- Puts out the text card images for the entries in the field name table
- Prints the symbol table at the same time

LITAD

- Puts out the text card images for the entries in the literal table
- Prints the symbol table at the same time

CMPSN (Part III)

- Gets each compression and analyzes it for type
- Branches out to the appropriate routine for the type of record
- Updates a block of compression records
- Puts out each block and obtains the next
- Repeats this procedure until all of the compression records have been updated
- Calls Assign Phase 2 at the end of the compression if there has been an overflow in the field name table or literal table
- Otherwise calls Assemble Phase 1

LTYP (Line Counter)

- Skips this type of compression

ETYP (File Extension)

- Obtains the address of the conversion routine name in this compression type from the field name table
- Inserts it in compression

DTYP (Input Field Name)

- Locates the address for the field name entry in the field name table
- Inserts it in compression
- Obtains the addresses for resulting indicators for control levels, matching fields, or chaining fields from the resulting indicator table
- Inserts them in compression

CTYP (Calculation)

- Locates the addresses for the resulting indicator entries in the resulting indicator table
- Inserts them in compression
- Locates the address for each factor entry in either the field name table or the literal table depending on the type of entry

MTYP (Output-Format Field)

- Obtains the addresses for resulting indicator, field name, and literal entries from the resulting indicator, field name, and literal tables
- Inserts them in compression
- Skips blank-after and sterling entries

TTYP (Table)

- Searches the field name table for the table name or names in this type of compression
- Inserts the address found in the table in compression

RTYP (Record Address File)

- Searches the field name table for CONDT
- Inserts the address found in compression

FTYP (File Description)

- Searches the field name table for the field name from this type of compression
- Inserts the address found in the table in compression
- Searches the resulting indicator table
- Obtains the address for an 0 type file indication

ITYP (Input Record)

- Searches the resulting indicator table if there is a file name or if this is a compression for an OR type record
- Inserts the address found in the table in compression

OTYP (Output Record)

- Obtains the addresses for the resulting indicators from the resulting indicator table
- Inserts addresses in compression

SUBROUTINES

The following subroutines exit via register 15 unless stated otherwise.

OVERLP

- Checks the length of a literal to determine if it exceeds 4096 positions from the base
- Adds the length to the address counter if the overflow exists

ALIGN

- Increments the address counter
- Adjusts the address to a full word boundary
- Adjusts the counter to show the true address

COUNT

- Increments the address counter
- Increments the overflow counter if overflow occurs
- Stores the updated counter in the interface

ZADR

- Converts the base and displacement form of address to absolute form
- Restores the overflow register
- Stores the absolute address

HEXIT

- Converts the base and displacement form of address to printable hexadecimal form
- Restores the print area pointer

TXTP

- Assembles the text into punched card format
- Creates multiple card images if the text length exceeds 56 characters
- Puts them out to the go/punch data set

RITLU

- Looks up the resulting indicator in the resulting indicator table
- Assigns a diagnostic code for an undefined or an unreferenced resulting indicator

LITLU

- Looks up a literal from the compression in the literal table
- Inserts the corresponding length, address, and decimal position into compression

FLDLU

- Looks up a field name from the compression in the field name table
- Assigns a diagnostic code for an undefined, multidefined, or an unreferenced field name
- Inserts the corresponding address from the table into compression

Linkage to CIOEX

PUTCOD, PHSCAL, PNTERR, RLDOUT, PNTSPC, PUTCMP, SWITCH, GETCMP

- Load call numbers
- Branch to OUT
- Link to the service routines in CIOEX

ERRTNE

- Moves the diagnostic code number and unpacked specification number to the print area
- Branches to the CIOEX linkage to print the error indication

INTRODUCTION

Assign Phase 2 is executed only if a table overflow occurred during the enter phases The table overflow condition indicates that because the number of entries exceeded the allotted table area, not all field names and literals have been assigned addresses. Assign Phase 2 creates the table or tables that overflowed, and assigns addresses to the entries.

Figure 24 and Chart HB illustrate the organization and operation of Assign Phase 2.

Figure 25 illustrates the input/output flow for Assign Phase 2.

LOGIC

The CIOEX data area contains the location of the specification that caused the table overflow. Assign Phase 2 begins with this specification and creates the necessary tables from the entries of field names or literals that do not have assigned addresses.

Any compression that has been assigned an address will have had the 6-byte name field replaced by the last six bytes of the corresponding table entry (see Appendix E). Knowing this, Assign Phase 2 has only to check the location in the compression where the beta byte would be. Since the zone portion of beta byte is always X'0', the presence of anything other than X'0' in this position of the compression indicates that the actual name is still present and therefore an address has not yet been assigned.

An exception to this method is a literal in the calculation compression; here the byte checked is the 12th byte of the literal area. When an address is assigned to the literal, the byte is set to zero.

When the last specification is checked, Assign Phase 2 computes addresses for the table entries and replaces field names and literals in compression with addresses assigned to the table entries. If an overflow occurs in building the tables, Assign Phase 2 is executed again; otherwise, Assemble Phase 1 is called.

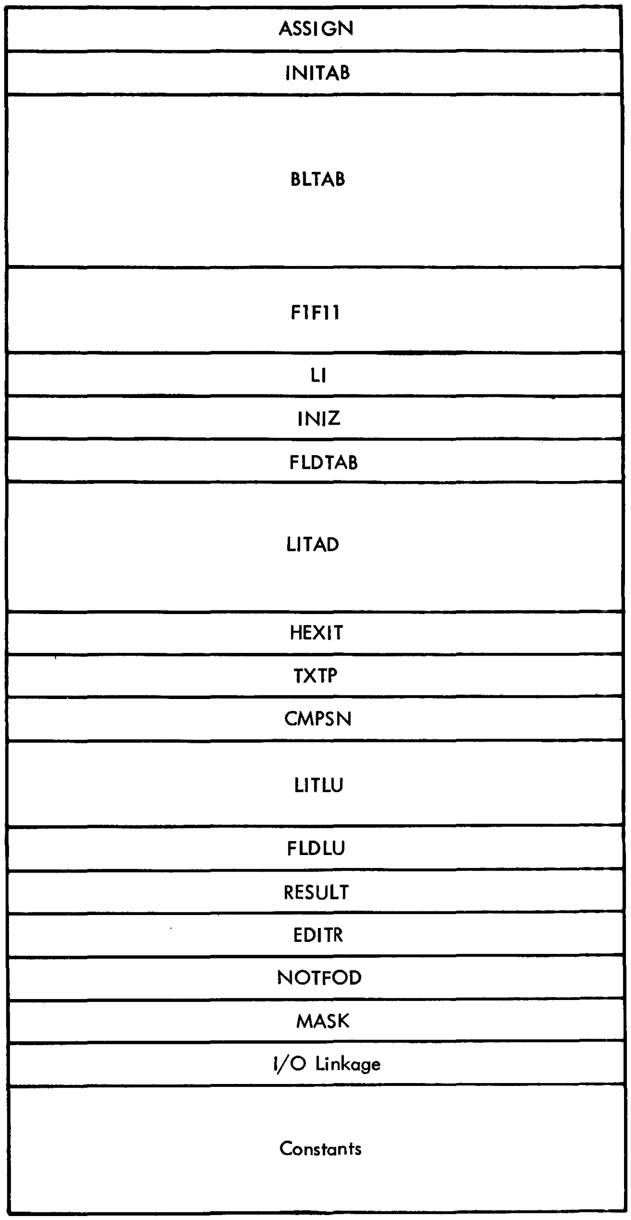


Figure 24. Assign Phase 2 Storage Allocation Map

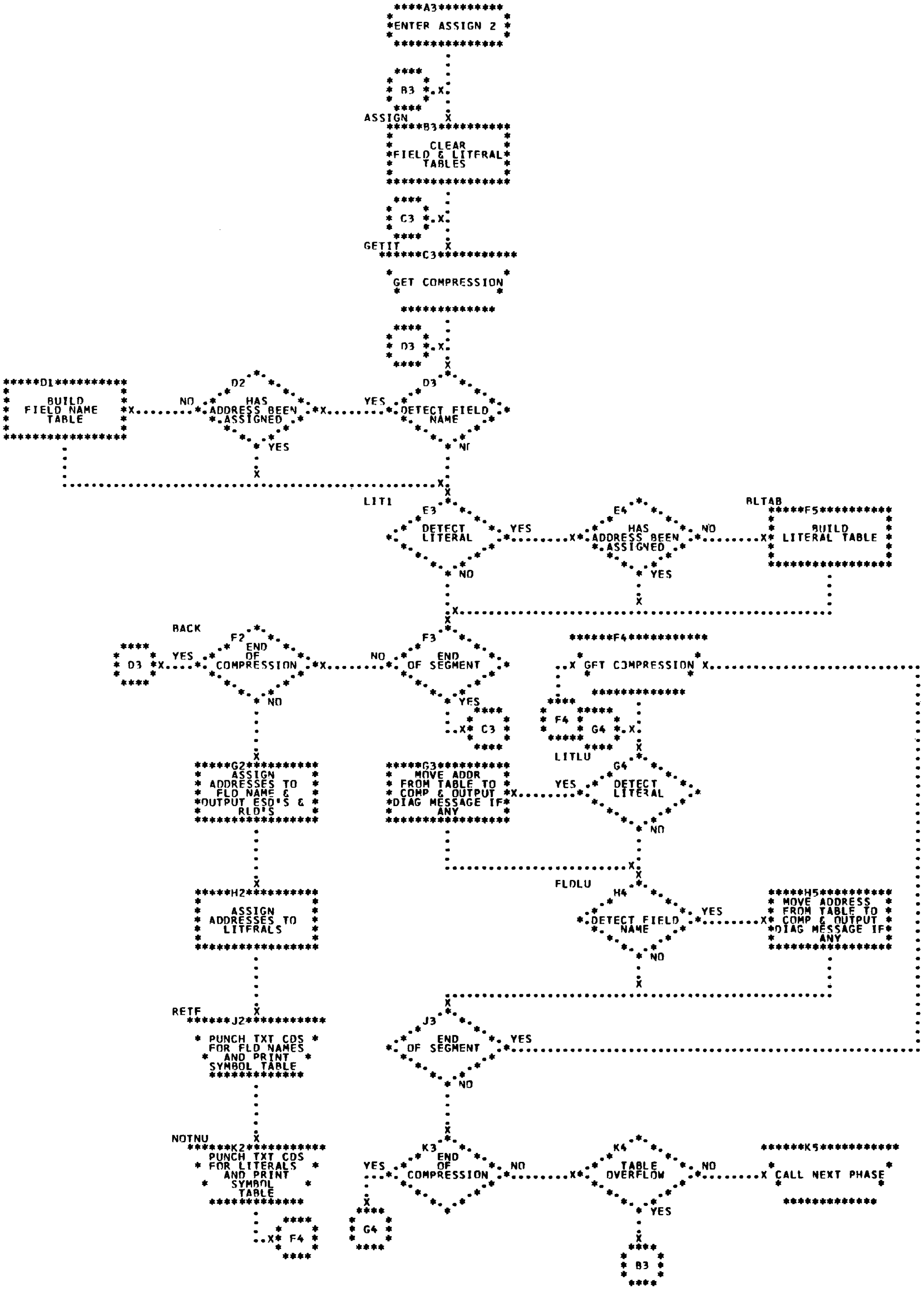


Chart HB. Assign Phase 2

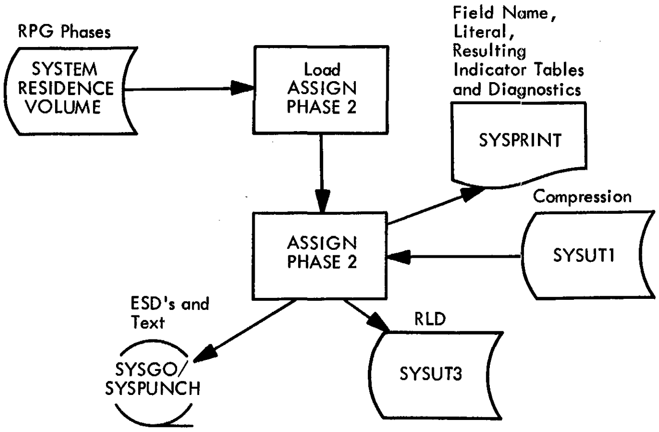


Figure 25. Assign Phase 2 Input/Output Flow

SWITCHES AND INDICATORS

LTKEY

LTKEY	xxxxxxx1	Calculation type compression
PAGE	xxxxxx1x	Page type field name
TAB	xxxxx1xx	On during table building phase of program
RESID	xxxx1xxx	Field name is from result field
BLKAF	xxx1xxxx	Blank-after
MCOMP	xx1xxxxx	M type compression
LNSW	1xxxxxxx	There is a length associated with name from type D and C compression
KEY	x1xxxxxx	More text to be output

PASID

Full word storage area. First byte used as indicator

X'00'	Off status
X'B0'	Second pass indicator for assign addresses to field names routine (FlFll); otherwise used as storage area.

MAIN ROUTINES

BEGIN

- Saves CIOEX key registers
- Moves overflow byte from CIOEX data area
- Saves the print buffer address

ASSIGN

- Clears field name and literal tables
- Goes to the specification that caused the overflow

BLTAB

- Build tables from compression

FlFll

- Makes two passes through field name table
- Assigns addresses to internal names on the first pass
- Assigns addresses to entry, external, and normal fields on the second pass
- Puts out ESD's for entry and external fields

LI

- Assigns addresses to literals
- Sets the decimal position to indicate the type of literal

INIZ

- Initializes for the text card routine and printing of symbol table

FLDTAB

- Puts out text card images for fields

LITAD

- Puts out text card images for literals

TXTP

- Formats and puts out text card images

CMPSN

- Prints last line of literals
- Puts out the last text card image
- Gets first input specification compression

RESULT

- Sets the reference byte for the result field names

EDITR

- Sets reference byte for the factor field names

NOTFOD

- Enters the field name in the table if there is room

MASK

- Finds the field mask for the corresponding gamma byte

SUBROUTINES

INITAB

- Initializes table area

ZADR

- Converts base and displacement form of address to absolute form

COUNT

- Increments the address counter

- Sets the overflow indicator when overflow occurs

ALIGN

- Aligns the address on a full word boundary

OVERLP

- Checks for a field that would overlap a 4096 boundary

HEXIT

- Converts the base and displacement form of address to printable hexadecimal format

LITLU

- Looks up literal in literal table

- Inserts the corresponding length, address, and decimal position into the compression

- Adds the literal to the table if there is room

FLDLU

- Looks up field name in table

- Inserts the address in compression

- Adds the field name to the table if there is room

INTRODUCTION

Assemble Phase 1 puts out precoded machine language subroutines for RAF, chaining and table files. This program builds parameter tables from the file description specification and file extension specification compressions for these types of files. These parameter tables are put out and provide the information necessary for the subroutines to operate at object time. Assemble Phase 2 also provides parameters for chaining files.

Figure 26 and Chart KA illustrate the organization and operation of Assemble Phase 1.

Figure 27 illustrates the input/output flow for Assemble Phase 1.

LOGIC

The addresses of the routines I/O intercept, record address file, chaining and table (input/output) are written on the work data set as V-type records (Appendix F). The linkage phase uses these records to develop the linkage vector.

The enter phases have set switches in the CIOEX data area to indicate the presence of RAF, chaining and table files. This allows Assemble Phase 1 to overlay the precoded routines after they are put out or if they are not needed by the program.

The order in which the routines and the associated information are put out is as follows:

1. Precoded chaining routine (CCOUNT) or the dummy chaining routine (CCDUMY)
2. Chaining parameter table records
3. Precoded table logic (input/output) (TC0000) or the dummy table routine (TCDUMY)
4. Parameter list table for table files
5. The table linkage field (TLF) for the first table. The TLF for the second table (if applicable)
6. The input/output interceptor routine (INTRCP)
7. Precoded RAF support (DC0000/IC0000)
8. V-type records for the linkage vector
9. R-type records for the RLD

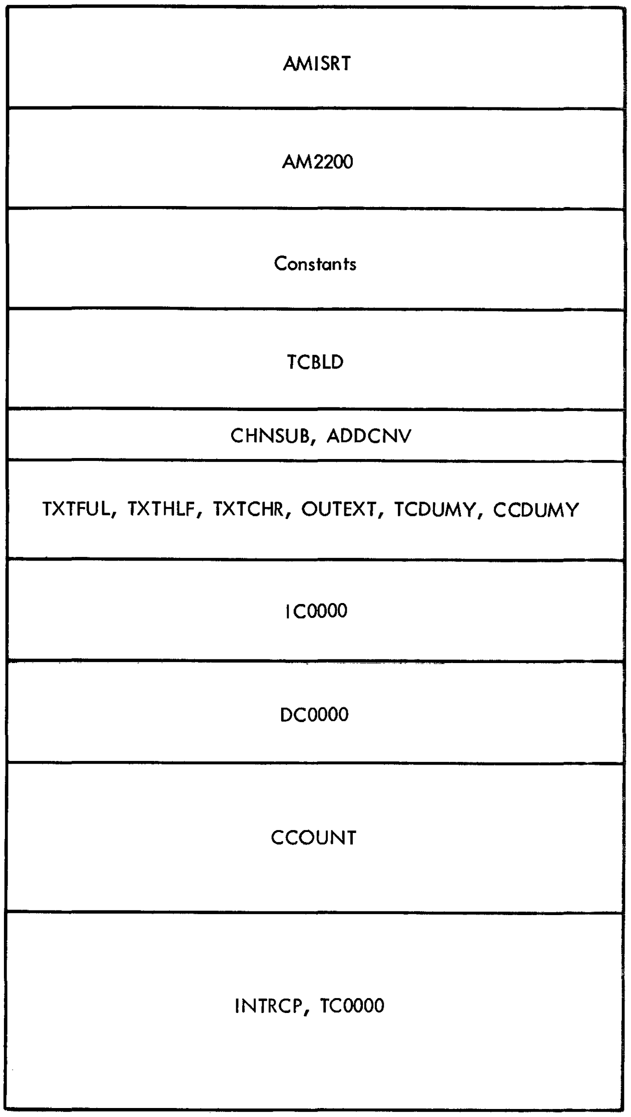


Figure 26. Assemble Phase 1 Storage Allocation Map

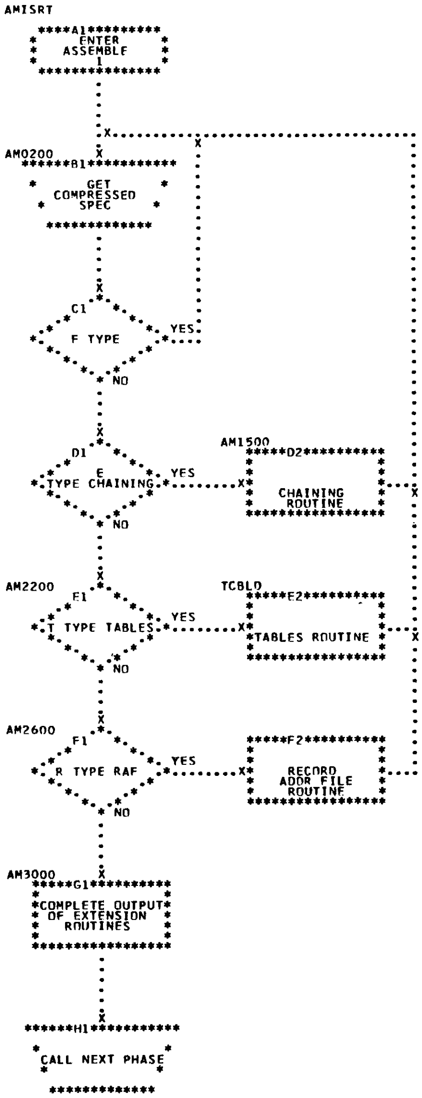


Chart KA. Assemble Phase 1

SWITCHES AND INDICATORS

CPSSWT

First Pass	X'F0'	Routine clears chaining buffer area
Last Pass	X'00'	Not first pass, bypass clear instructions

CLSTPS

Last Pass	X'F0'	Last file extension processed. Chaining is completed and buffer has been put out
	X'00'	Not last pass, bypass complete chaining instructions

MAIN ROUTINES

AMISRT

- Reads the compression record as input
- Verifies the type of specification compression
- Stores the length of record from a file description specification compression for an RAF
- Returns to read the next compression record
- Puts out the chaining logic (CCOUNT) for a file extension compression if the first pass switch (AMSWTH) is set as X'00'
- Sets AMSWTH as X'F0'
- Builds a parameter table for chaining files
- Builds a parameter table for table files

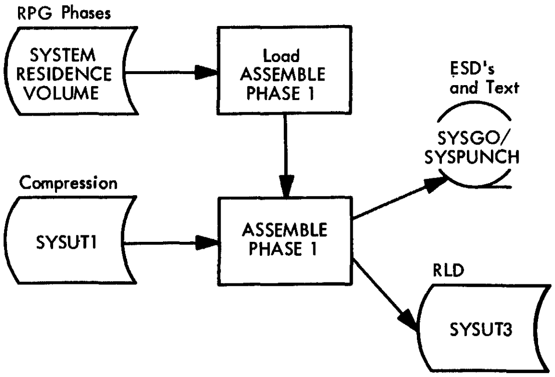


Figure 27. Assemble Phase 1 Input/Output Flow

- Puts out the filled chaining parameter table block directly behind the chaining logic
- Extracts the from filename and to filename fields from a compression record for an RAF
- Inserts them into the RAF routines for indexed-sequential (IC0000), and direct organization (DC0000)

AM2200

- Completes and puts out the chaining parameter table when the last file extension specification compression has been processed
- Puts out the table logic (TC0000) and the parameter table for table files
- Puts out the dummy table routine (TCDUMY) if there have been no table files
- Puts out the dummy chaining routine (CCDUMY) if there have been no chaining files
- Builds the TLF (table linkage field) for the first table entry and for the second table entry if necessary. This field is built from the entries in the parameter table for table files. The table linkage field is assembled and put out as follows:

Table Linkage Output 16 Bytes:

Bytes				
1	Table type byte			
		Bits 0-3	0000	Not used
		4	0	Numeric table
			1	Alphameric table
		5	0	Packed input
			1	Unpacked input
		6,7	01	Ascending table
			10	Descending table
			00	Neither
			11	Not used
2	Length of entry			
3-4	Number of table entries			
5-8	Address of start of table			
9-12	Address of end of table			
13-16	Work address initialized as zero			

- Puts out the I/O intercept routine and the RAF support routine specified (direct organization or indexed-sequential)
- Sets AMDIOR as X'F0' for direct organization; X'00' for indexed-sequential
- Sets up and puts out the linkage vector images
- Calls Assemble Phase 2

SUBROUTINES

The following subroutines exit via register 15 unless stated otherwise.

TCBLD

- Builds a list of parameters for a table file defined in a file extension compression. The parameter entry or entries are as follows:

Byte 1	Action byte	X'01' for input/ no second entry. X'03' for input/ second entry		
2	Input file number (binary)			
3	Output file number (binary)			
4,5	Number of entries per record (binary)			
6,7	Number of table entries (binary)			
8-10	Address of TABNNN-16 for first file			
11	Length of table entries in first table			
12	Unpacked/packed, numeric/alphameric, ascending/descending for first table			
		Bits 0-3	0000	Not used
		4	0	Numeric table
			1	Alphameric table
		5	0	Packed input
			1	Unpacked input
		6,7	01	Ascending table
			10	Descending table
			00	Neither
			11	Not used
13-15	Address of TABNNN-16 for second file			
16	Length of table entries in second table			
17	Unpacked/packed, numeric/alphameric, ascending/descending for second table			

TXTFUL, TXTHLF, TXTCHR

- Prepares text for output to a given target area. The text is either full word or half word aligned in segments of 56 bytes, or less

- Builds the parameters for use by the output routine as follows:

SLVRG1	Number of bytes to be put out
SLVRG2	Address of the target area

- Links to OUTEXT
- Returns with the target address in SRTPNT

ADDCNV

- Converts an address from the form BDDD to absolute form. The address is contained in a field described as follows:

  WRKRG1 Points to BDDDXXXXXX0K/Returns with converted address.
  B = Data base register (DATBS1 to DATBS1+4)
  DDD = Displacement
  K = Overflow key (0,1,2,3,4)
  DATBS1 Represents lowest data base register
  HLFWRD Is a half word aligned work area
  SLVRG2,LNKREG are used as work registers

  The overflow key (0K) specifies the number of times 4096 must be added to the base

- Places the absolute address developed in WRKRG1

OUTEXT

- Puts out text using the parameters developed by TXTCHR plus PUNREG (the address of the output buffer)and LOCREG (the address of the field)
- Links to CIOEX to put out the text card image

CHNSUB

- Builds a list of parameters for a chaining file defined in a file extension compression. The parameter entry is as follows.

Byte 1	From file
2,3	Record sequence number
4	Numeric portion of chaining field number
5	To file
6-8	Conversion routine linkage point address

  Entries are put out with six entries per card immediately following the chaining logic.

- Links to TXTCHR to put out the chaining parameter list buffer when six entries have been developed

PRECODED ROUTINES

The following precoded routines are designed to exit via register 15 unless stated otherwise.

TC0000

- Included in the output of this phase if there has been a file extension specification compression for a table file
- Inserts entries in a table or extracts entries from a table when executed in the RPG object program

CCOUNT

- Included in the output of this phase if there has been a file extension specification compression for a chaining file
- Supports the split chaining fields option
- Uses the switch CCFOND set as X'F0' to indicate the first entry for split chaining fields has been processed
- Sets CCPROS as X'00', X'02', X'04', X'06' for the particular processing pass of the routine
- Sets the halt indicator INDHHO to X'F0' when the chaining request is not in the table of acceptable chaining requests

DC0000

- Included in the output for this phase if there is a file extension specification compression for an RAF with direct organization. This form of access must have a conversion routine
- Builds the linkage to the conversion routine within this routine during the first pass. The first pass switch is DCPSWT.
- Links with the input/output routine (IOROUT) to get records from the input file

IC0000

- Included in the output for this phase if there is a file extension specification compression for an RAF with indexed-sequential specified. The first pass switch is ICPSWT

- Links with the input/output routine (IOROUT) to get records from the input file

TCDUMY

- Put out if there is not an actual table file defined
- Links back to the program directly

CCDUMY

- Put out if there is not an actual chaining file defined
- Turns on halt indicator H0 (INDHHO set as X'F0')

INTRCP

- Establishes linkage to the RAF support routine included in the program if the object program refers to an RAF
- Otherwise establishes linkage to the input/output routine included in the program

INTRODUCTION

Assemble Phases 2 and 2.5 process the input specifications compressions and generate RPG object program text. Two phases are needed to complete the processing required.

The functions of Phase 2 are

1. Scan the file description specifications and enter information regarding multiple file specifications in the file environment table (FET)
2. Generate object program routines to move input data into the appropriate fields and to call the chaining subroutine if chaining fields are present
3. Put out a record to the work data set for each control level field and matching field. These records are used for generating object code to determine control breaks and matching records
4. Reserve a common working storage area in the object program to be used for processing the control level and matching fields

The functions of Phase 2.5 are

1. Complete processing of the input specification record type. Process OR specifications by completing the object code instruction generation
2. Complete processing of the input specification field type. Generate object code instructions for the last OR specification of a record group and for the processing of matching fields
3. Put out blank-after entries to be processed in Assemble Phase 4
4. Put out the precoded object program routines to control and analyze input

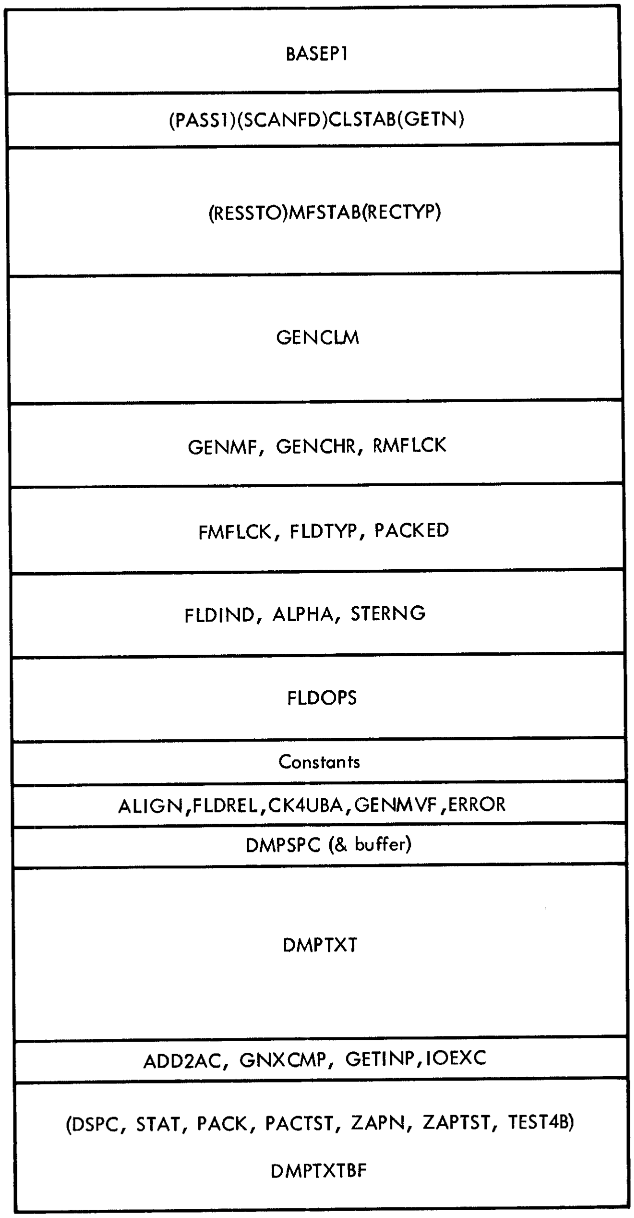
Figures 28 and 29 and Charts LA and LB illustrate the organization and operation of Assemble Phase 2 and 2.5.

Figures 30 and 31 illustrate the input/output flow for Assemble Phase 2 and 2.5.

LOGIC (PHASE 2)

The instructions that are generated for moving input data are created by this pass for both record and field type input specifications. The object program precoded subroutine output moves data and provides linkage to the subroutines to pack and unpack fields and to check for blanks and field status.

A chaining request block (CHB) is created for each field designated as a chaining field. These CHBs are linked by pointers. In the object routine generated for the record group is a call for the chaining subroutine with a pointer at the first chaining field CHB.



NOTE: Labels within parenthesis identify routines that are overlaid.

Figure 28. Assemble Phase 2 Storage Allocation Map

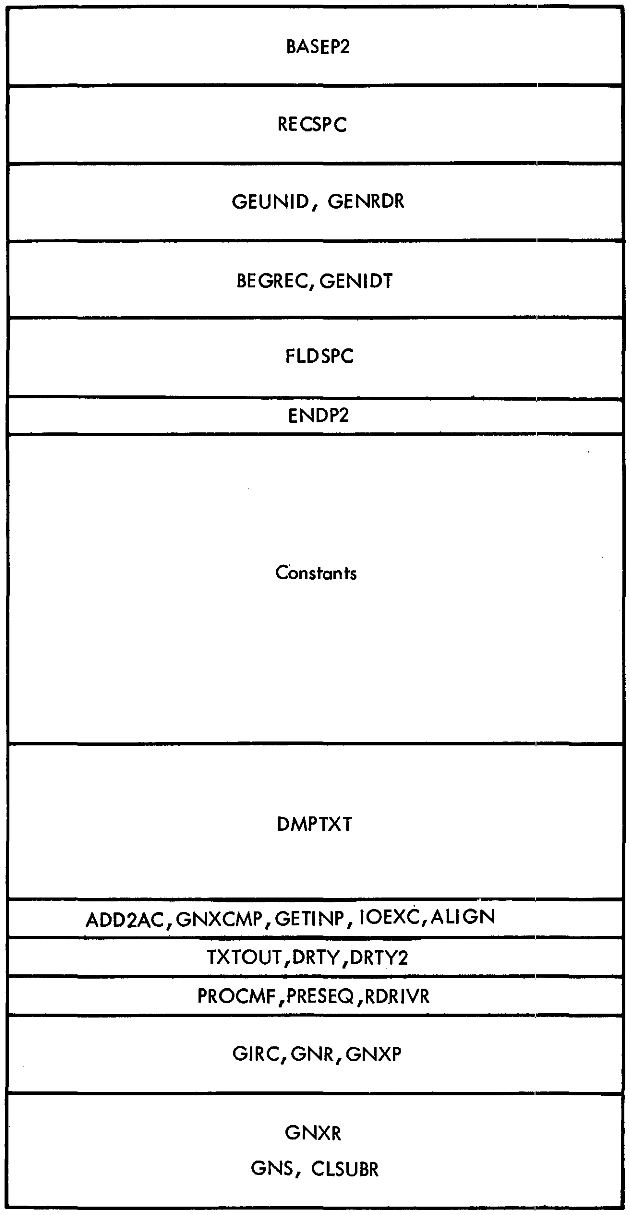


Figure 29. Assemble Phase 2.5 Storage Allocation Map

LOGIC (PHASE 2.5)

The blank-after entries consist of 12 bytes and are blocked six per physical record. The format of these entries is illustrated in Appendix G. If a block is not filled when the phase is completed, it is filled with X'00'.

The data that is common to Phase 2 and 2.5 is generated by Phase 2 and written to a work data set. Subsequently the data is read in to overlay a similarly defined area in Phase 2.5.

SWITCHES AND INDICATORS

CLS

X'F0' Control levels are present

CLSIND

X'F0' Set during Phase 2 to cause to put out CLSUBR

FLDSW

X'F0' Field specifications are present

IDCSW

X'F0' No ID codes are present

PHASE 2 MAIN ROUTINES

SCANFD

- Conditions CIOEX to read and make available the file description specification compressions
- Tests for file type (primary, secondary, chained) and the bits are set in the file environment table (FET) as follows:

BITS	VALUE	
0-1	00	Primary file
	01	Secondary file
	11	Chained file
2	Not used	
3	Not used	
4	1	E specified in column 17 of file description specification
	0	Blank in column 17
5	1	Ascending sequence
	0	Descending sequence or blank
6	Not used	
7	1	Card input device
	0	Other device

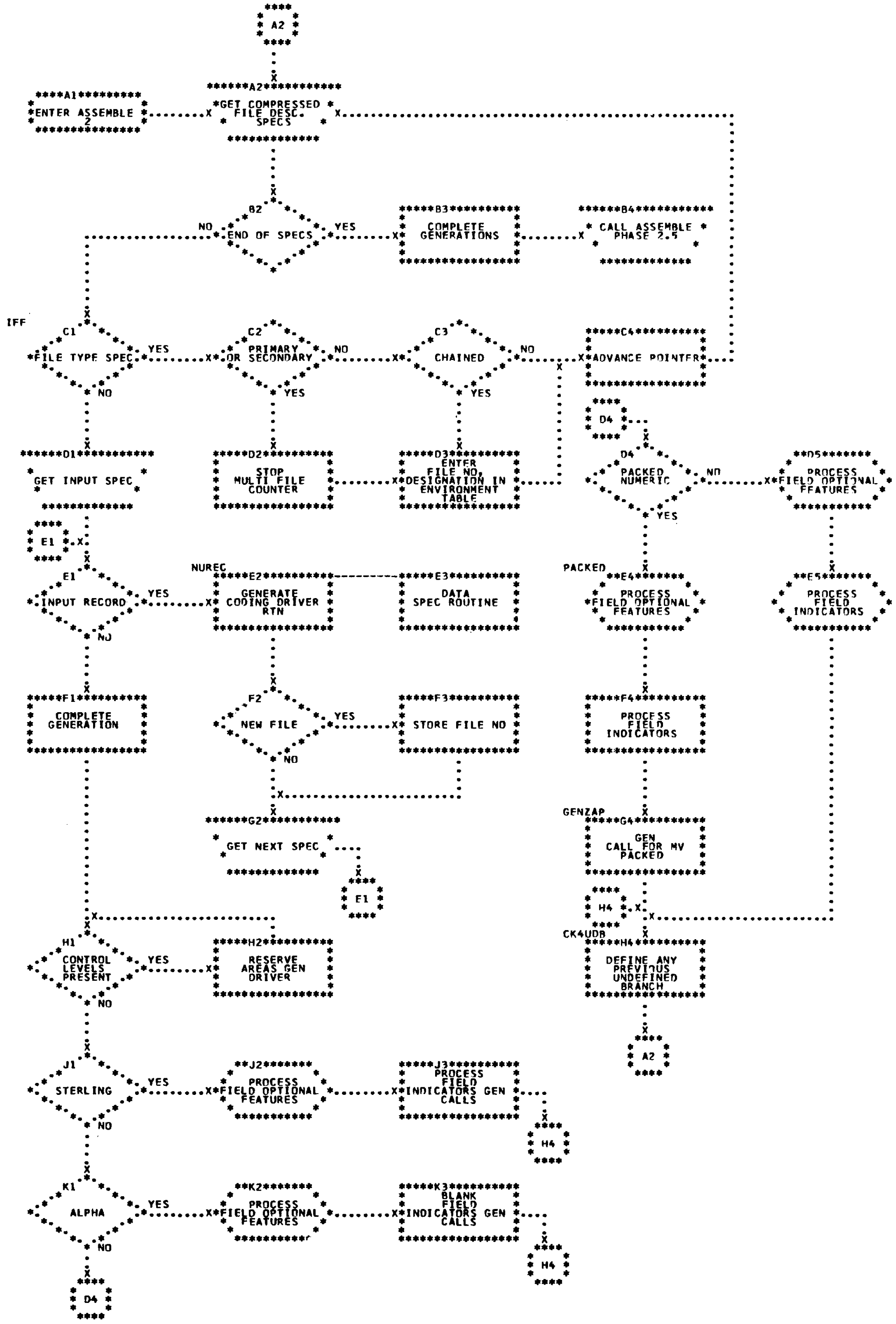


Chart LA. Assemble Phase 2

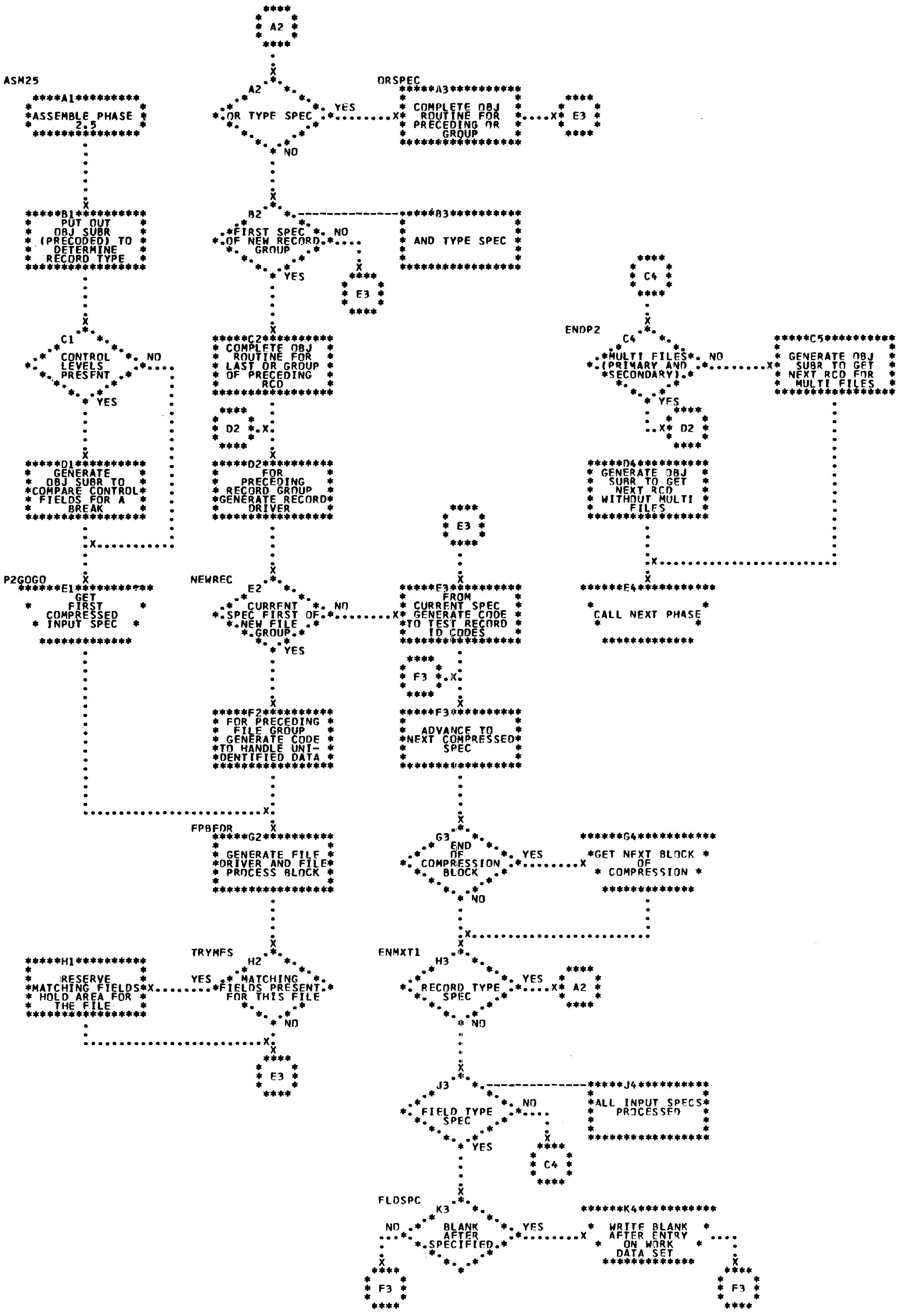

A2
ASM25
A1
ASSEMBLE PHASE 2.5
A2
OR TYPE SPEC
YES
ORSPEC
A3
COMPLETE OBJ ROUTINE FOR PRECEDING OR GROUP
E3
NO
B1
PUT OUT OBJ SUBR (PRECODED) TO DETERMINE RECORD TYPE
B2
FIRST SPEC OF NEW RECORD GROUP
NO
B3
AND TYPE SPEC
E3
YES
C4
C1
CONTROL LEVELS PRESENT
NO
YES
C2
COMPLETE OBJ ROUTINE FOR LAST OR GROUP OF PRECEDING RCD
ENDP2
C4
MULTI FILES (PRIMARY AND SECONDARY)
NO
C5
GENERATE OBJ SUBR TO GET NEXT RCD FOR MULTI FILES
YES
D2
D2
D1
GENERATE OBJ SUBR TO COMPARE CONTROL FIELDS FOR A BREAK
D2
FOR PRECEDING RECORD GROUP GENERATE RECORD DRIVER
D4
GENERATE OBJ SUBR TO GET NEXT RCD WITHOUT MULTI FILES
E3
P2GOGO
E1
GET FIRST COMPRESSED INPUT SPEC
NEWREC
E2
CURRENT SPEC FIRST OF NEW FILE GROUP
NO
E3
FROM CURRENT SPEC GENERATE CODE TO TEST RECORD ID CODES
E4
CALL NEXT PHASE
YES
F3
F2
FOR PRECEDING FILE GROUP GENERATE CODE TO HANDLE UNIDENTIFIED DATA
F3
ADVANCE TO NEXT COMPRESSED SPEC
FPBFOR
G2
GENERATE FILE DRIVER AND FILE PROCESS BLOCK
G3
END OF COMPRESSION BLOCK
YES
G4
GET NEXT BLOCK OF COMPRESSION
NO
H1
RESERVE MATCHING FIELDS HOLD AREA FOR THE FILE
TRYMFS
H2
YES
MATCHING FIELDS PRESENT FOR THIS FILE
NO
ENMXT1
H3
RECORD TYPE SPEC
YES
A2
NO
E3
J3
FIELD TYPE SPEC
NO
J4
ALL INPUT SPECS PROCESSED
C4
YES
FLOSPC
K3
NO
BLANK AFTER SPECIFIED
YES
K4
WRITE BLANK AFTER ENTRY ON WORK DATA SET
F3
F3

Chart LB. Assemble Phase 2.5

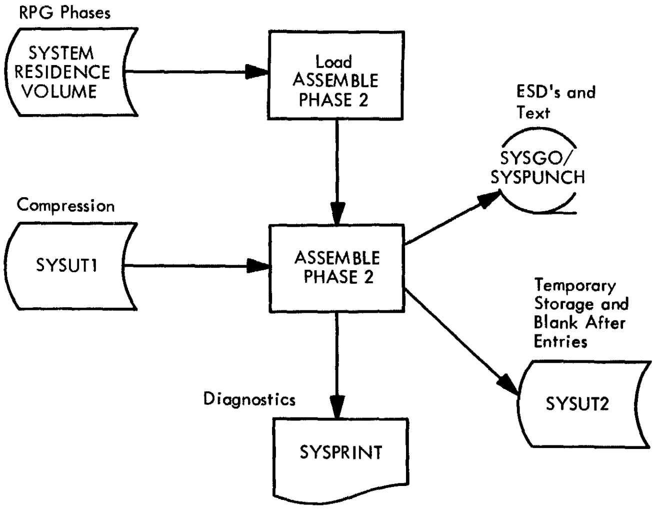


Figure 30. Assemble Phase 2 Input/Output Flow

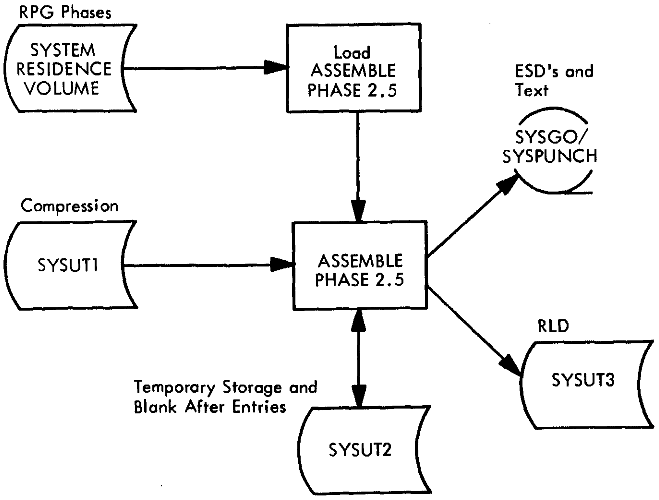


Figure 31. Assemble Phase 2.5 Input/ Output Flow

GETN

- Uses GETIN to get the first input specification compression record for processing

- Calls ALIGN to force the object program addresses to be aligned as a full word

- Clears MFTARG, MFLTAB, and MFDUPS tables which are used for processing specifications for matching fields

RESSTO

- Overlays the preceding routines starting at the symbolic location PASS1

- Reserves working storage in the object program for processing control levels and matching fields specifications. Register 2 contains the number of bytes to be reserved

RECTYP

- Generates the entry and exit points of the MVFLDS object program subroutine

- Generates MVFLDS subroutine for each record group to move the input data into assigned fields, sets field indicators, and calls the chaining subroutine if chaining fields are specified

- Completes the MVFLDS subroutine for the preceding group when a new record group is detected

- Performs validity checking to check the matching fields specifications (if applicable) for the preceding group

- Calls CIOEX to get the next input specification

- Starts again for a record type

- Branches to FLDTYP for a field type

- Completes Phase 2 processing

Phase 2 processing is completed by reserving a working storage area within the object program if control levels and/or matching fields are specified. The reserved area is equal to either the sum of the maximum length specified for control levels or the sum of the maximum length of the matching fields. Generates a table of the displacements to the starting byte for each control level (1-9) within the common working storage.

Finally, the control level and matching fields records that have been written on the work data set during the execution of Phase 2 are processed (GENCLM). The object program subroutines, MCL2WS and MFSUBR, for processing control levels and matching fields are generated.

Phase 2 reinitiates the work data set, writes out the data common to Phase 2 and 2.5 and calls CIOEX to bring in Phase 2.5.

FLDTYP

- Generates the instructions that move input data into assigned fields

- Tests field status

- Sets the field indicators

- Generates the chaining request block for chaining field specifications

- Processes alphabetic, sterling, and packed numeric fields through subroutines

- Generates the appropriate calling sequences for the object program

- Calls CIOEX to get the next input specification compression

PHASE 2.5 MAIN ROUTINES

BASEP2

- Brings in the common data and tables that were saved on the work data set from Phase 2 as the first logical step in Phase 2.5

- Puts out the object program subroutine DRTY that calls the FDRIVR (file driver or linkage table) for the requesting file

- Puts out other precoded object program routines under certain circumstances. For instance, if control levels are present, CLSUBR is generated and put out

- Generates FDRIVR and FPB (file processing block) at the beginning of each file group (FPBFDR)

- Branches into the section of BEGREC that generates the first instruction for the object program for this input record

RECSPC

- Performs the processing of the input record type specification

- Completes generating the object code for the preceding OR group for an OR type record

- Completes the linkage between the groups

GENIDT

- Generates object program instructions that test record ID codes to determine record type

- Calls CIOEX (GNXCMP) to get the next input specification

- Exits to FLDSPC for a field type

- Exits to RECSPC for a record type

- Exits to ENDP2 to complete the processing for Assemble Phase 2 and 2.5

FLDSPC

- Performs the processing of input field type specifications

- Completes the generation of the object code for the last OR of the current record group

- Calls CIOEX to get the next input specification

- Branches to exit as described in GENIDT

ENDP2

- Completes generating the object code for the last record group of the last input file group

- Generates and puts out GIRC which is the object program subroutine that gets the next input record for processing

- Branches to put out the last block of blank-after entries

- Restores CIOEX registers

- Branches to CIOEX to call Assemble Phase 3

SUBROUTINES COMMON TO PHASE 2 AND PHASE 2.5

The following subroutines exit via register 15 unless stated otherwise.

ALIGN

- Obtains the object program counter from the CIOEX data area

- Forces the object program counter to full word alignment

- Restores the updated counter to CIOEX

- Exits via register 14

GETINP

- Calls CIOEX to get the first input specification compression

- Exits with the first byte address of the input specification compression in register 3

GNXCMP

- Examines the next available byte in the compression block

- Exits if the field type is I (record) or D (field) or if the next specification is other than an input type

- Calls CIOEX to get the next compression block

- Returns to the test for record type

ADD2AC

- Maintains the object program address counter that contains the absolute address, plus one, of the last location used by the object program being compiled

DMPTXT

- Accepts object code and generates a binary text card for output

- Writes card images to be punched eventually on the go/punch data set

PHASE 2 SUBROUTINES

GENCHR

- Generates the exit (BR) from the MVFLDS object program subroutine for the preceding record group

- Generates a calling sequence for the chaining subroutine if chaining fields were present in the preceding group

- Places the decrement to the first chaining block into the calling sequence

- Advances the object program address counter (ADD2AC)

- Exits via register 12

RMFLCK

- Executed when matching fields were present in the record group preceding a new record group just detected

- Checks the equality of the total length of each set of M1-M3 fields in the preceding record group if there is more than one set to a record group

- Assigns a diagnostic code if the length of each set is not the same

- Sets an indicator for a diagnostic code if the length of each set is not also equal to the length of each set in all previously processed record groups of the file group

FMFLCK

- Executed when a new input file group is detected or after the last input file group has been processed

- Checks multi-files (primary and secondary) if there were matching fields in the preceding file group

- Assigns a diagnostic code if matching fields are not specified for the primary file

- Assigns a diagnostic code if matching fields are missing for the secondary file

- Assigns a diagnostic code if the lengths of the M1-M3 fields for the primary and secondary file are not equal

- Assigns a diagnostic code if the indicator has been set for it during execution of RMFLCK

CK4UBA

- Defines the branch address for the BNE instruction generated by FLDREL when the field-record relation is present

- Exits via register 12

FLDREL

- Executed if a field has an associated field-record relation indicator

- Generates instructions that test the status of the indicator and branch to the next field if the indicator is not on

- Exits via register 12

FLDOPS

- Processes the control level field to determine the maximum length of each level (L1-L9) in order to reserve common working storage in the object program

- Processes the field-record relation indicator (FLDREL)

- Generates the object code to test the field-record relation indicator if a specification has a field-record relation indicator and chaining fields (IFCHF)

- Sets the CHB (data identifier) processed code ON if the indicator is not ON. The chaining request block generated is as follows:

```
CHB     DC  X'B00C'   Data identifier
                      (compiler use)
        DC  X'F0'     Processed code and
                      field address over-
                      flow
FILENO  DS  CL1       File number
                      (binary)
RECSEQ  DS  CL2       Record sequence
CHFNO   DC  X'00'     Chaining field
                      number
        DS  CL1       Field length
        DS  CL2       Field address
        DS  CL2       Decrement for
                      accessing preceding
                      CHB
```

- Checks the matching field table for duplicate matching fields if there are no chaining fields but there are matching fields
- Eliminates duplicates because M1-M3 is allowed once with each field-record relation indicator and once without an associated indicator
- Makes an entry in the table if this M number is not a duplicate
- Accumulates the total length of the fields of each set
- Puts out a matching field specification on the work data set

FLDIND

- Places the address of each indicator used into the calling sequence for the move fields subroutine (MV2FLD) in the object program
- Changes the entry displacement to the move
- Performs status checks
- Exits via register 14

DMPSPC

- Controls blocking and output of control level specifications and matching field specifications onto the work data set

These specifications are subsequently processed by GENCLM after the execution of Phase 2 is complete and before Phase 2.5 is brought into core. When the work data set is brought back into core, the records overlay the part of the Phase 2 program logic starting at symbolic location PASS1.

The format of the records that are written on the work data set is

RRFFLNA

where

RR = Field-record relation indicator (the first byte of RR = X'FD' if no field-record relation indicator
FF = Address of input data (BDDD)
L = Field length
N = Level number, 1-9 for control levels and 1-3 for matching fields. The zone of this byte indicates the specification type, C for control level or E for matching field
A = Decimal position and packed indication (same as compression)

GENCLM

- Processes control level and matching fields specifications

The following instructions are generated for the object program for a control level specification with or without a field-record relation indicator. These instructions move the control level data from the input area to the working storage.

```
MVC  TA(L,GR9),FF(GR2)  Moves alpha and
                        packed data
MVN  TA(L,GR9),FF(GR2)  Moves unpacked
                        data
```

For fields without a field-record relation indicator

TA = CLWSDP(N-1)+DPWORR(N-1)

For fields with a field-record relation indicator

TA = CLSWDP(N-1)+DPWRR(I)

where

(N = Control level number)

- Defines any preceding undefined branches (for a control level with a field-record relation)
- Generates the instructions necessary for the object program to test the field indicator (FLDREL)
- Generates for a matching fields specification
- Supplies the input data area address and length of matching fields to the routine DRTY2 which is an object code subroutine put out during 2.5

- Makes a test for the type of specifications being processed at the end of a record group

- Generates a branch as the last instruction of MFSUBR which is the object program subroutine for matching fields specifications

ALPHA

- Generates an instruction for alphameric input data
- Processes the alphabetic input data by branching out to FLDOPS, GENMVF and DMPTXT

- Returns to the main program to get the next specification

STERNG

- Generates the calling sequence for the sterling input conversion subroutine in the object program

- Processes sterling input data by branching out to FLDOPS, GENMVF, and DMPTXT

- Returns to the main program to get the next specification

PHASE 2.5 SUBROUTINES

The following subroutines exit via register 15 unless stated otherwise.

TXTOUT

- Checks the buffer area used for assembling text output for overflow

- Generates an additional branch if the current request would cause an overflow

- Prevents DMPTXT from creating output by branching around the undefined branch addresses and creating a new undefined branch address. The preceding UBAs are defined to branch to the new UBA

- Branches to DMPTXT to put out the text if there is no overflow

GENRDR

- Generates instructions and fills in pertinent data in RDRIVR at the end of a record group. RDRIVR (an object program subroutine) is generated for each record group and follows the coding which tests the record ID codes

- Branches to TXTOUT to put out the text before the exit is taken

GEUNID

- Checks for ID codes in the preceding group at the end of a record group

- Defines the BNE instructions generated in the preceding group to go to the next group if the ID codes are not equal

- Generates a stacker select code if the file device specified by the FET entry is a card reader; the generated code is put out by TXTOUT

PHASE 2 PRECODED ROUTINES

DSPC

- Receives control from either the linkage routine or the chaining subroutine

- Executes the MVFLDS subroutine for the input record type being processed; the MVFLDS subroutine moves the input data into the assigned fields and calls the chaining subroutine for chaining fields

- Contains the transfer vector (table) to branch to the following subroutines:

  PACK Packs unpacked numeric input data into a packed field

  PACTST Packs unpacked numeric input data into a packed field and checks the status of the field (STAT)

  ZAPN Moves packed numeric input data into a field

  ZAPTST Moves packed numeric input data into a field and checks the status of the field (STAT)

  TEST4B Checks an alphameric field to see if it contains all blanks and, if so, sets the blank indicator ON

  STAT Checks the status of a numeric field and sets the respective plus, minus, or zero indicator ON

PHASE 2.5 PRECODED ROUTINES

DRTY

- Called by the chaining subroutine to determine the record type and set the resulting indicator ON

- Register 1 contains the number of the requesting file when this routine is entered

- Calls the FDRIVR (file driver or linkage routine) corresponding to the requesting file number

DRTY2

- Called from DRTY and GNR to determine the record type and to process matching fields

- Entry conditions are as follows

GR9	Address of IORB
GR1	Address of FPB
GR13	Address of RPG object program common area

PROCMF

- Moves matching fields to working storage and compares them with matching fields of the preceding record group

- Turns on halt indicator H0 to indicate an out of sequence condition (for file sequence checking)

- Entered from DRTY2, the conditions are as follows

GR9	Address of common working storage for matching fields and control levels (reserved by Assemble Phase 2)
GR10	Address of M1 field in input area
GR11	Address of M2 field in input area
GR12	Address of M3 field in input area

PRESEQ

- Checks input records that are designated to be in a predetermined sequence

- Included in the object program immediately following DRTY and DRTY2 object program subroutines

- Predetermined sequence is specified in the input record type specifications

- Entered from RDRIVER (record driver or linkage routine)

- Sets halt indicator (H0) if the sequence number is not equal to the expected sequence number, the option was not specified, and there is only one specification; if there is more than one specification in the above instance, the indicator will be set if there is not at least one specification with the expected sequence number

RDRIVR

- Generated for each record group

- Included in the object program following the coding that tests the record ID codes for the record group

- Turns on the proper resulting indicator when the record is identified

- Places the addresses, necessary for processing the record, in the FPB (file processing block)

GIRC

- Receives control from the linkage routine in order to get the next input record for processing

- Turns on the resulting indicator associated with the record

- Tests for control level breaks

GNR

- Called from either GNP or GNS to read a record and determine the record type; address of the FPB (file processing block) for the requesting file is in register 1

- Calls the input/output routine to read the record and DRTY2 to determine the record type

GNXP

- Generated as a part of GIRC when multifiles are present

- Calls GNR to read the next record from the primary file

- Sets the end-of-file indicator for the primary file

- Tests the end-of-file indicator for the secondary files

GNXR

- Selects the next input record for processing

- Reads the next record from the file last processed

- Compares records from the primary and secondary (or group of secondary) files to determine which record is to be processed next

- Processes files that are in ascending sequence in the following order:

  $P < S$ Process primary
  $P > S$ Process secondary
  $P = S$ Process all primary records of the matching group and then process the matching secondary records

- Processes files that are in descending sequence in the following order:

  $P < S$ Process secondary
  $P > S$ Process primary
  $P = S$ Same as for ascending sequence

GNS

- Reads the secondary file last processed
- Compares the records from each secondary file to select one to compare with the primary file record
- Selects the record with the matching field of the lowest value if the files are in ascending sequence
- Selects the highest record if the records are in descending sequence

CLSUBR

- Compares control level fields of the input records being processed with the same fields of the preceding records
- Sets the appropriate level indicator or indicators on when a control break occurs
- Replaces the preceding value, which is stored in a hold area, with the value of the field in the current record
- Entry is made into CLSUBR for each level (L1-L9) present within a record group
- Entry conditions are as follows:

  GR9 Points to working storage area where the control level data of the current input record is stored
  GR10 Points to hold area containing the latest control level values

- Logic of CLSUBR is repeated in the object program nine times, once for each level, L1-L9
- Operands of the MVC and CLC instructions are modified for each time the coding is repeated

INTRODUCTION

Assemble Phase 3 processes the calculation specifications compressions and generates RPG object program text. The detail specifications are processed first with the total specifications following. If there is an RPGCV or EXTCV entry, this phase makes two passes over the compression. The first pass expands all of the calculation specifications except the EXTCV (and associated KEYCV) and the specifications that are between the RPGCV and ERPGC. The second pass extends the EXTCV and RPGCV specifications. The functions of Assemble Phase 3 are

1. Decompress the calculation specifications compressions
2. Calculate all addresses required for the operation expansions
3. Select segments of an operation expansion according to the attributes of the fields involved
4. Put out the object program text to the go/punch data set

Figure 32 and Charts MA and MB illustrate the organization and operation of Assemble Phase 3. Figure 33 illustrates the input/output flow for Assemble Phase 3.

LOGIC

As the compression is decompressed and each field is analyzed, the characteristics are stored in condition codes. These condition codes are used to select the object code expansion that fits the requirements of the specification.

The method of developing all of the addresses for all possible operations for each field avoids the need for the logic to choose and calculate a subset of the addresses.

Blank-after requirements are detected and put out to a work data set to be processed by Assemble Phase 4.

SWITCHES AND INDICATORS

FIRSTOT

X'FF' The program has executed the detail end logic

SONOF

X'00' Generate code for SETOF
X'F0' Generate code for SETON

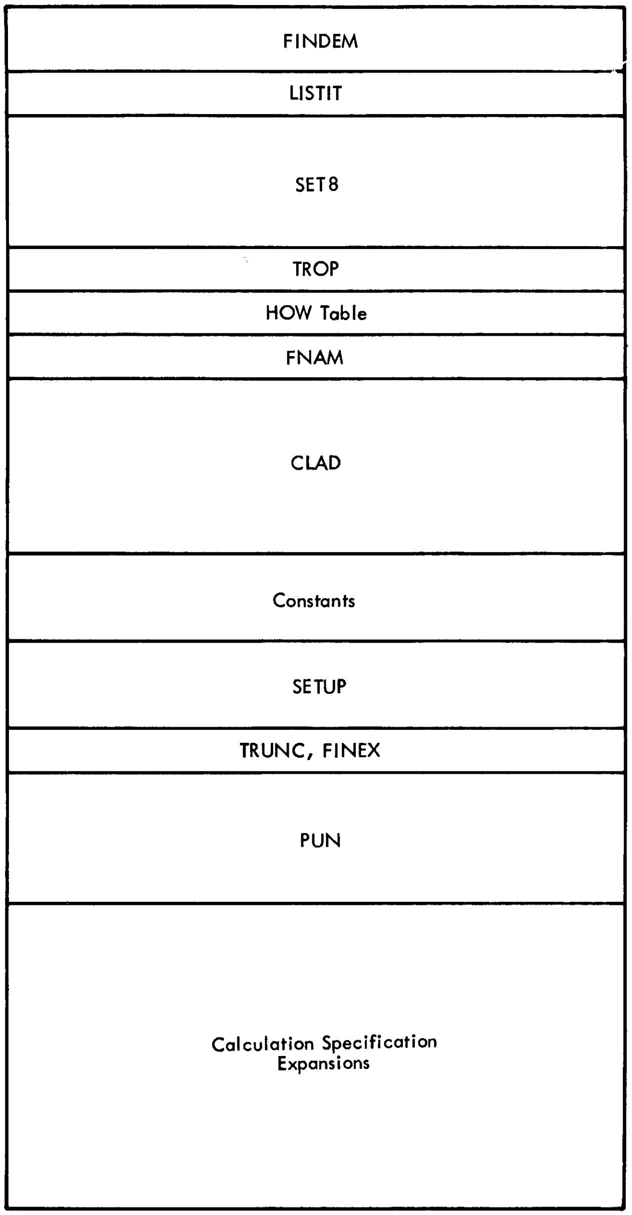


Figure 32. Assemble Phase 3 Storage Allocation Map

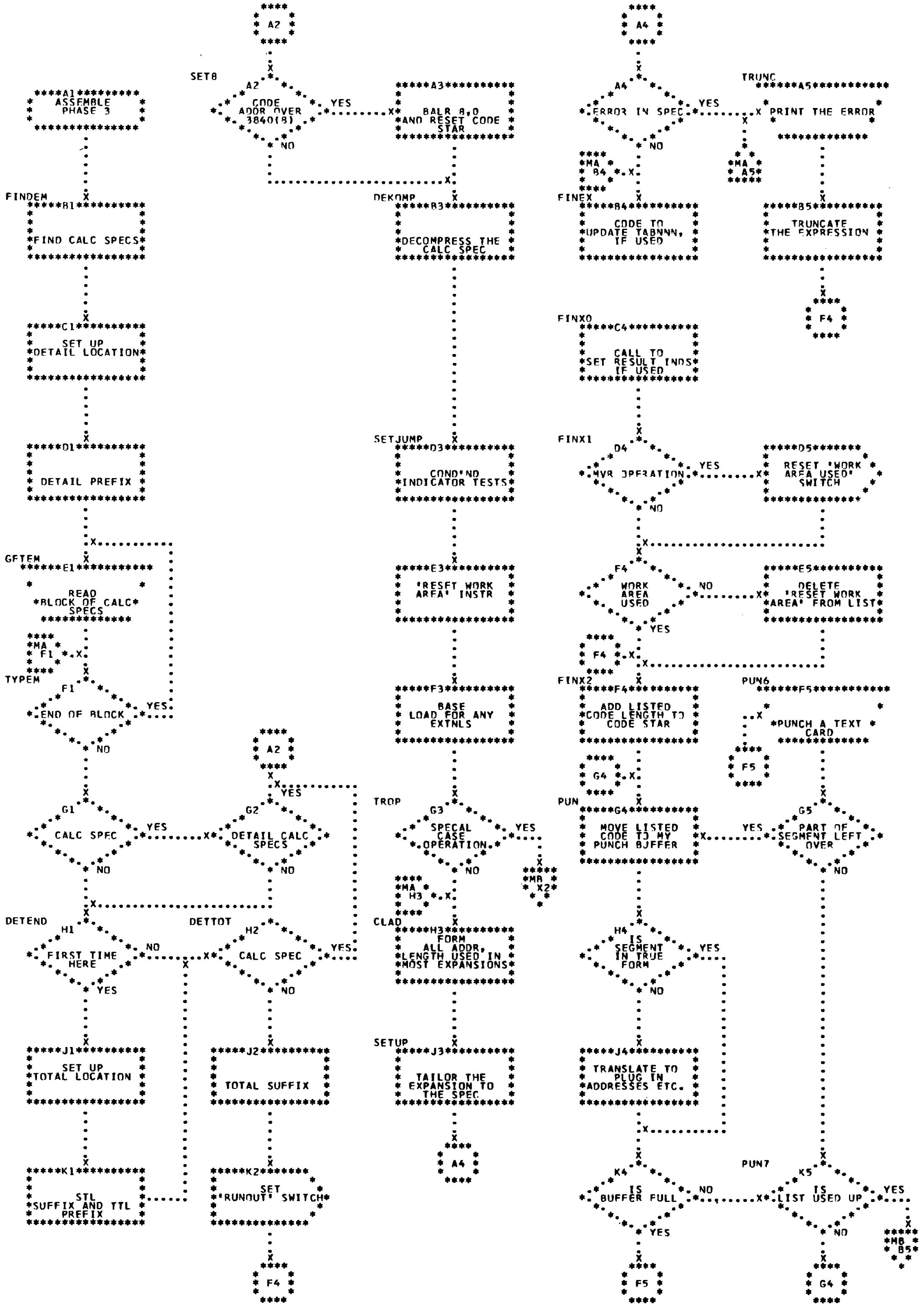


Chart MA. Assemble Phase 3

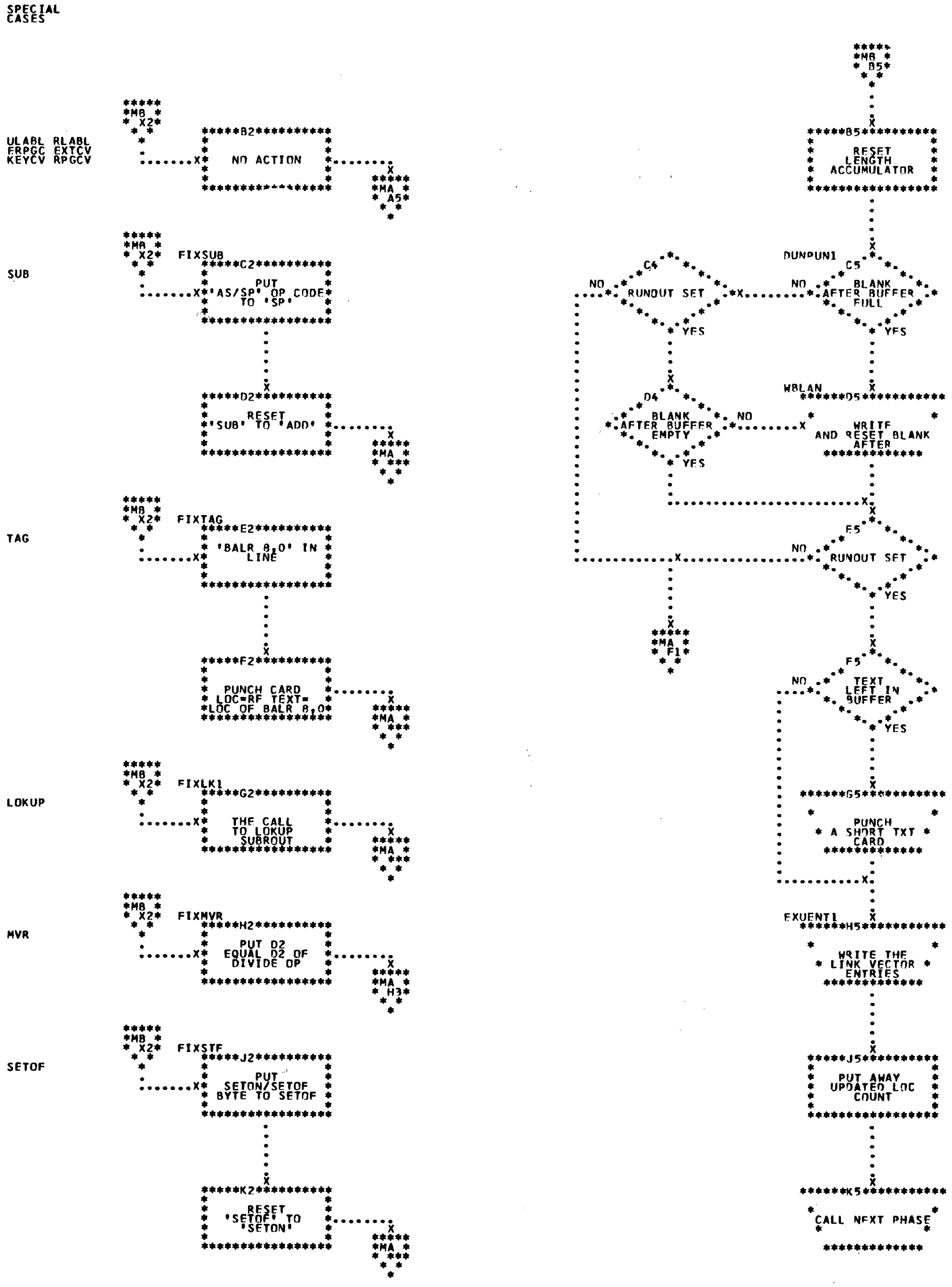


Chart MB. Assemble Phase 3

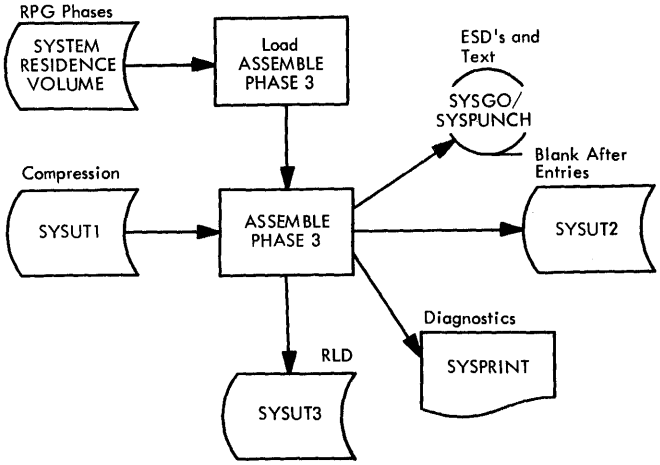


Figure 33. Assemble Phase 3 Input/Output Flow

XAPSP

X'FB'	Generate code for SUB
X'FA'	Generate code for ADD

USEW1

X'01'	Work area is used

MAIN ROUTINES

FINDEM

- Puts out the detail prefix
- Accumulates the computed length of text
- Branches to the decompression and expansion routine (SET8) for a calculation specification compression
- Otherwise, branches to FINX2 to conclude the functions of the phase

SET8

- Decompresses the calculation specification compression
- Checks and analyzes the following fields: control level, indicators, factor fields 1 and 2, operation code, result field, and resulting indicators
- Constructs the LOKUP switch
- Puts out the number of tests involved, the code for the reset, and the loads needed to get the EXTERN addresses

TROP

- Translates the operation code from hexadecimal (in compression) to an internal sequence number, for subsequent translation by a branch to the proper subroutine
- Generates the object code expansion, depending on the branch address (either a segment address or a special case address), as illustrated in the following table, which has the added information corresponding to each instruction:

xxxxxxx1	1 - branch address 0 - expansion address
xxxxxx1x	1 - calculate address 0 - do not calculate address
xxxxx1xx	Used for KEYCV 1 - operation cannot update table
xxxx1xxx	1 - resulting indicator set as part of expansion 0 - not set as part of expansion
xxx1xxxx	1 - operands must be numeric 0 - operands may be alphabetic

HOW DC	X'08'	For	SETON	Go To	SETUP
	X'01'		RPGCV*		TRUNC
	X'09'		SETOF		FIXSTF, SETUP
	X'00'		GOTO		SETUP
	X'00'		EXIT		SETUP
	X'0D'		LOKUP		FIXLK1
	X'02'		MHHZO**		CLAD, SETUP
	X'0A'		COMP		CLAD, SETUP
	X'0A'		TESTZ		CLAD, SETUP
	X'01'		TAG		FIXTAG
	X'12'		Z-ADD		CLAD, SETUP
	X'12'		Z-SUB		CLAD, SETUP
	X'12'		ADD		CLAD, SETUP
	X'11'		SUB		FIXSUB, CLAD, SETUP
	X'12'		MULT		CLAD, SETUP
	X'12'		DIV		CLAD, SETUP
	X'13'		MVR		FIXMVR, CLAD, SETUP

*Same for ERPGC,EXTCV,RLABL,ULABL,KEYCV
**Same for MOVE,MLLZO,MHLZO,MOVEL,MLHZO

The special cases that allow multiple use for sets of code or instruction modification are SUB, SETOF, LOKUP, MVR, and TAG. ERPGC, EXTCV, RLABL, RPGCV, ULABL and KEYCV all branch to truncate (TRUNC) the expansion and bypass the specification.

The table DECODE contains the addresses of the object code expansions and is used in conjunction with HOW. The table BLUN is used for converting the compressed operation code into its sequential position for use in a computed GOTO.

SETUP

- Uses the information extracted and stored by CLAD, LISTIT, and TROP2 to select segments of an operation expansion. Stored information is as follows:

Reg	0	RLEN	Length of list buffer (computed in LISTIT)	
	1	RD2D1	SW1 in first byte and SW4, SW7 in next two bytes	
	2	RACC	SW2	These switches are all set up in CLAD where all attributes are evaluated. Their meaning is made explicit in the section named CLAD and also wherever they are used or tested. Registers 6,7,8 are also used as work registers.
	3	RDWDR	SW3	
	4	RZCC	SW5	
	5	RDRD2	SW6	
	6	RLRL2	SW8	
	7	RF2RF	SW9	
	8	RL1L2	SW10	
	9	RSEG	Contains pointer to segment address setup in TROP2	
	10	RYES	Points to INTERP	
	11	RSTEP	Step counter for number of segments to be skipped	
	12	RWAY	Holds computed GOTO branch, and used as scratch	
	13	RNO	Has address of NEWSEG	
	14	RSCAN	Points to code in segment currently used	

TRUNC

- Provides linkage between this phase and CIOEX to print the diagnostic codes assigned in this phase
- Truncates the expansion at the last calculation specification on an error

FINEX

- Does some final checking on tables, resulting indicators, MVR operations, and use of work areas
- Computes and transfers the length of the code of the expansion to the routine (PUN) that moves the expansion to the punch buffer area

PUN

- Moves the selected expansions to the punch area, using the length and the address from the list (see SETUP)
- Stores a new address and length at the same list position for continuations
- Links with CIOEX to put out the text
- Writes out the linkage vector entries for calling the detail and total lines routines

SUBROUTINES

LISTIT

- Moves the length and address of a sequence of object code into a list (LIST1), referred to during the output to the punch routine (PUN)

FNAM

- Separates and stores the attributes of the field name as follows:

  Field Name =B (BASE) DDD (Displacement).
  BETABETA = Field information byte
  LL = Field length-1 (in binary)
  X=X'E' for external field, or X'0' if not external
  D=X'B' for alphameric, X'0'-X'9' for numeric
  OV = Overflow key X'00'-X'04'

  Literal field is all the same except that XD is replaced by OD=X'OB' for alphameric, X'00'-X'09' for numeric, and X'0A' for edit word.

  Field information byte (BETABETA), starting from left, has format TTTTXYZZ where TTTT is not used and

```
 X = 0  No blank-after is wanted
 Y = 0  BDDD address reference is direct
     1  It refers to the table linkage
        field
ZZ = 00 Table is in ascending order
     01 Table is in descending order
     10 Order is not specified
```

If data base overflow is indicated, the appropriate base number is substituted in the field address. The 24-bit (unrelocated) address of the field is formed for possible use in the look-up (LOKUP) operation.

CLAD

- Loads registers with the attributes of the field name and calculates the addresses required
- Stores the condition codes and addresses for use in other routines. The list of registers and the information they contain is described under SETUP

INTRODUCTION

These phases perform additional diagnostics and generate an executable object routine in relocatable text, which assembles and puts out all records requested on the output-format specifications sheet. The data used are the compressed specifications from the file description and output-format specifications, produced previously by Enter Phases 1 and 6, and the resulting indicator information from the blank-after work file produced by Assemble Phases 2 and 3.

Figures 34, 35 and Charts NA, NB, NC, ND, NE illustrate the organization and operation of Assemble Phases 4 and 4.5. Figures 36 and 37 illustrate the input/output flow for Assemble Phases 4 and 4.5.

LOGIC

Assemble Phases 4 and 4.5 are separately loaded and executed. Phase 4 generates the instructions that perform the logical tests to determine when the record is to be put out, and that put out the line. This phase, which is in two sections, is concerned with compressed specifications labeled 0 (line). Phase 4.5 generates the instructions to assemble the output record from the component fields and to test any output indicators associated with it. This phase is concerned with compressed specifications labeled M (field).

In processing the output-format entries, the field name is tested to determine if a blank-after is indicated. For this situation, the blank-after data set is searched for an equivalent field name address, and instructions are generated to set any associated resulting indicators on or off.

Section 1 of Phase 4 builds the file description table and performs error diagnostics. This table contains information on all valid output files in the source specifications. This information consists of such entries as type, format, record length, overflow indicator, output device, and sequence number of the defining specification.

Section II of Phase 4 has the following separate functions:

1. Generates a linkage point and puts it on the RLD data set
2. Generates the instructions necessary to assemble and output all total records not conditioned by overflow
3. Generates the instructions for testing the overflow switches for all data sets
4. Generates a linkage point and puts it out on the RLD data set
5. Generates the instructions necessary to assemble and puts out all total records conditioned by overflow

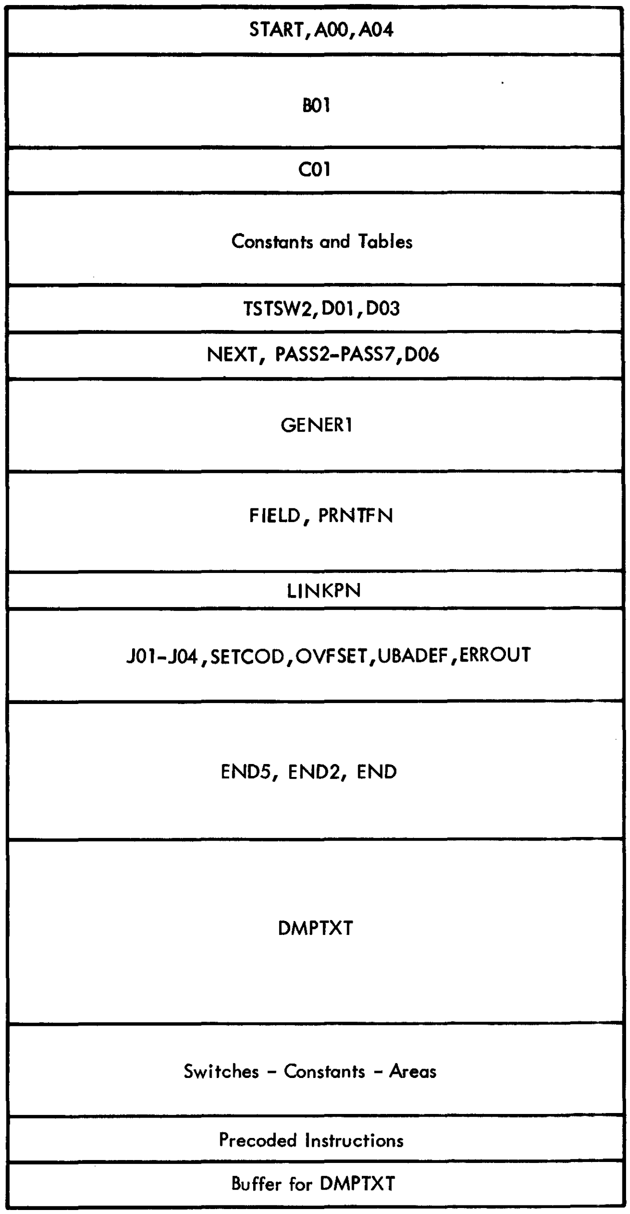


Figure 34. Assemble Phase 4 Storage Allocation Map

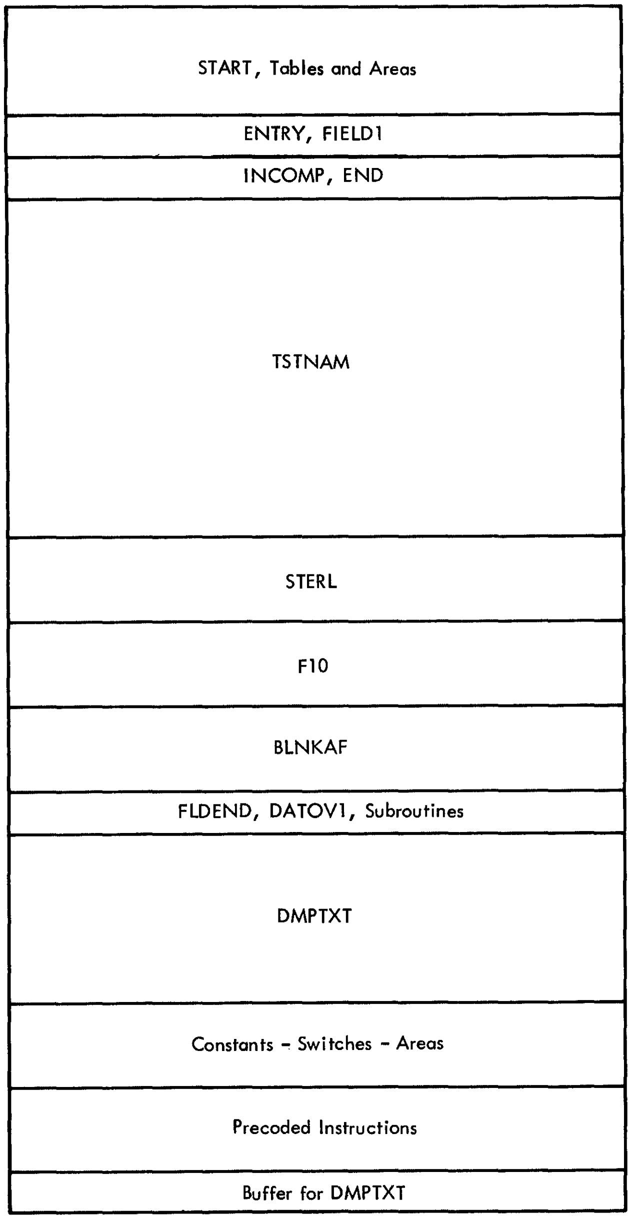


Figure 35. Assemble Phase 4.5 Storage Allocation Map

6. Generates the instructions necessary to put out all heading records and all detail records conditioned by overflow
7. Generates a linkage point and puts it out on the RLD data set
8. Generates the instructions to turn off all X-switches
9. Generates the instructions necessary to assemble and puts out all heading records and all detail records not conditioned by overflow
10. Generates the instructions to turn all overflow switches on and off
11. Places the file description table, built in section 1 of Phase 4, after the blank-after data set for communication with Phase 4.5

Phase 4.5 creates all of the object code necessary to assemble the fields for the line coding generated in Phase 4. In processing the output-format entries, the field name is tested to determine if a blank-after is indicated. If it is indicated, the data set containing the blank-after entries is searched for equal field name addresses, and instructions are generated to set on any associated blank or zero indicator. The starting address for each string of field coding is placed in the assemble 4 table. Each string of line code that is generated for the O-type compressed specifications, ends in a branch to the assemble 4 table. The addresses that have been placed in the table provide the link to the field coding for that line. The field coding, in turn, returns the program to the next sequential line to be processed.

The maximum number of unique lines of coding that can be generated is 1023.

SWITCHES AND INDICATORS

General Switch No. 1 - Phase 4

TTOVSW	xxxxxx1x	Data pass 5 completed
PROCSW	xxxxx1xx	Present line started
BALRSW	xxx1xxxx	BALR issued
LINESW	xx1xxxxx	Previous line built
ANDSW	x1xxxxxx	Process AND specification
PNCHSW	1xxxxxxx	Punch/print issued

General Switch No. 2 - Phase 4

FLDSW	xxxxxxx1	Suppress field linkage
PAS2SW	xxxxxx1x	Indicates data pass 2 completed

General Switch No. 1 - Phase 4.5

PROCSW	xxxxxxx1	Line not to be processed
FLD1SW	xxxxxx1x	First field of a line
ULABSW	xxxxx1xx	ULABL was used
SIGNSW	xxxx1xxx	Non-standard sterling sign
EDITSW	xxx1xxxx	Edit a sterling field
FTOSW	xx1xxxxx	First-time switch branch address
NOFTAB	x1xxxxxx	Not in file description table

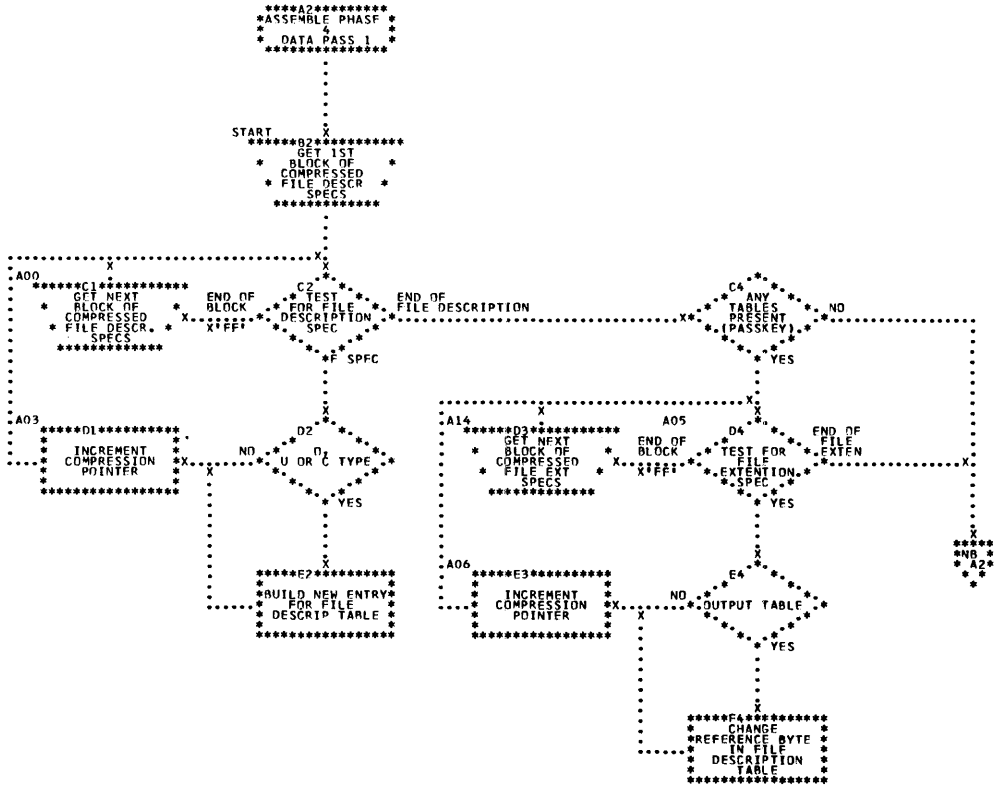

A2
ASSEMBLE PHASE 4 DATA PASS 1
START
B2
GET 1ST BLOCK OF COMPRESSED FILE DESCR SPECS
A00
C1
GET NEXT BLOCK OF COMPRESSED FILE DESCR. SPECS
END OF BLOCK X'FF'
C2
TEST FOR FILE DESCRIPTION SPEC
END OF FILE DESCRIPTION
F SPEC
A03
D1
INCREMENT COMPRESSION POINTER
NO
D2
U OR C TYPE
YES
E2
BUILD NEW ENTRY FOR FILE DESCRIP TABLE
C4
ANY TABLES PRESENT (PASSKEY)
NO
YES
A14
D3
GET NEXT BLOCK OF COMPRESSED FILE EXT SPECS
A05
END OF BLOCK X'FF'
D4
TEST FOR FILE EXTENTION SPEC
END OF FILE EXTEN
YES
NB A2
A06
E3
INCREMENT COMPRESSION POINTER
NO
E4
OUTPUT TABLE
YES
F4
CHANGE REFERENCE BYTE IN FILE DESCRIPTION TABLE

Chart NA. Assemble Phase 4

PHASE 4 - SECTION I ROUTINES

A00

- Searches the file description specification compressions for valid output files and builds table entries for them

A04

- Searches the file extension specification compressions for any table references to output files

B01

- Scans the output-format compressions to complete the file description table and to determine if any invalid files are referenced in these specifications

C01

- Makes a reverse scan of the file description table in order to diagnose file type errors

- Assigns a diagnostic code for an error that is detected

PHASE 4 - SECTION II ROUTINES

TSTSW2

- Re-initializes the compression for the next pass through the output-format specifications

D01

- Examines the next compression

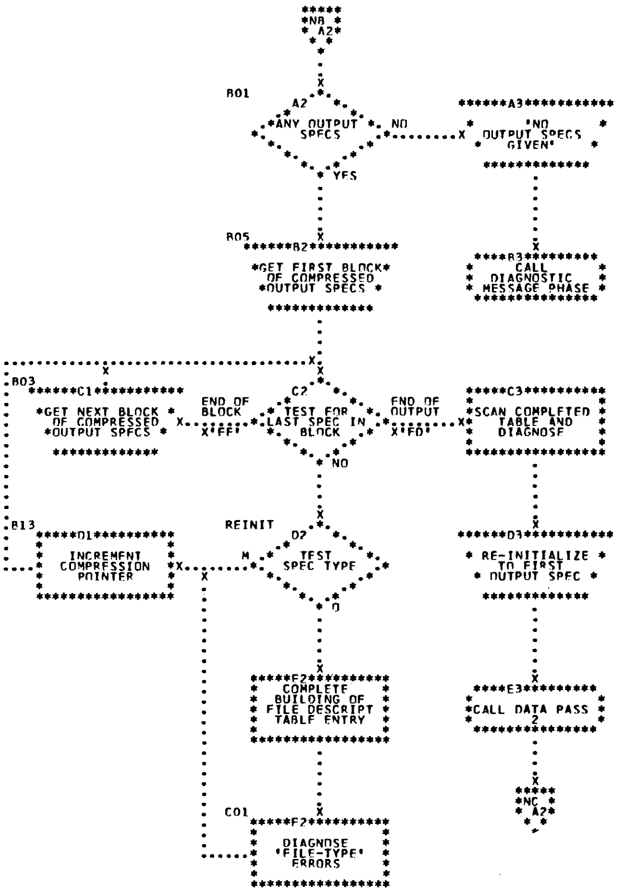

NB
A2
B01
A2
ANY OUTPUT SPECS
NO
YES
A3
'NO OUTPUT SPECS GIVEN'
B05
B2
GET FIRST BLOCK OF COMPRESSED OUTPUT SPECS
B3
CALL DIAGNOSTIC MESSAGE PHASE
B03
C1
GET NEXT BLOCK OF COMPRESSED OUTPUT SPECS
END OF BLOCK
X'FF'
C2
TEST FOR LAST SPEC IN BLOCK
END OF OUTPUT
X'F0'
NO
C3
SCAN COMPLETED TABLE AND DIAGNOSE
B13
D1
INCREMENT COMPRESSION POINTER
REINIT
D2
TEST SPEC TYPE
M
O
D3
RE-INITIALIZE TO FIRST OUTPUT SPEC
E2
COMPLETE BUILDING OF FILE DESCRIPT TABLE ENTRY
E3
CALL DATA PASS 2
NC
A2
C01
F2
DIAGNOSE 'FILE-TYPE' ERRORS

Chart NB. Assemble Phase 4

- Tests for the end of the compression block, the record type (O or M), and the record type for AND and OR specifications

- Puts out the print/punch function information if necessary

D03

- Performs the operations required to finish the previous line and start the next. These operations finish the code generation and fill in the UBAs for the last line, update the address counter to include the last code block, store filename and the print/punch function information from the compression

NEXT

- Not a routine but a list of branch instructions that guide the program through the various passes, in sequence

D06

- Resets the appropriate switches for the next specification and increases the compression pointer in order to obtain the next specification

PASS2-PASS7

- A set of six routines that determines, by testing, if a specification is to be processed by the particular pass

GENER1

- Generates the user object code for line type (O) specifications

- Includes routines to process AND and OR types, H/D/T types, and specifications with no resulting indicators

FIELD

- Sets up the linkage to the assemble 4 table that is built in Phase 4 to provide an address for a branch to the coding for the field type (M) specifications

PRNTFN

- Issues the code for moving the print/punch function information from the compression to the IORB

LINKPN

- Puts out the V-type linkage record on the RLD data set

- Issues the first object code of the block

J01 - J04

- Common linkages to the punch text code routine (DMPTXT), which issues the code for BALR and a resulting indicator test,

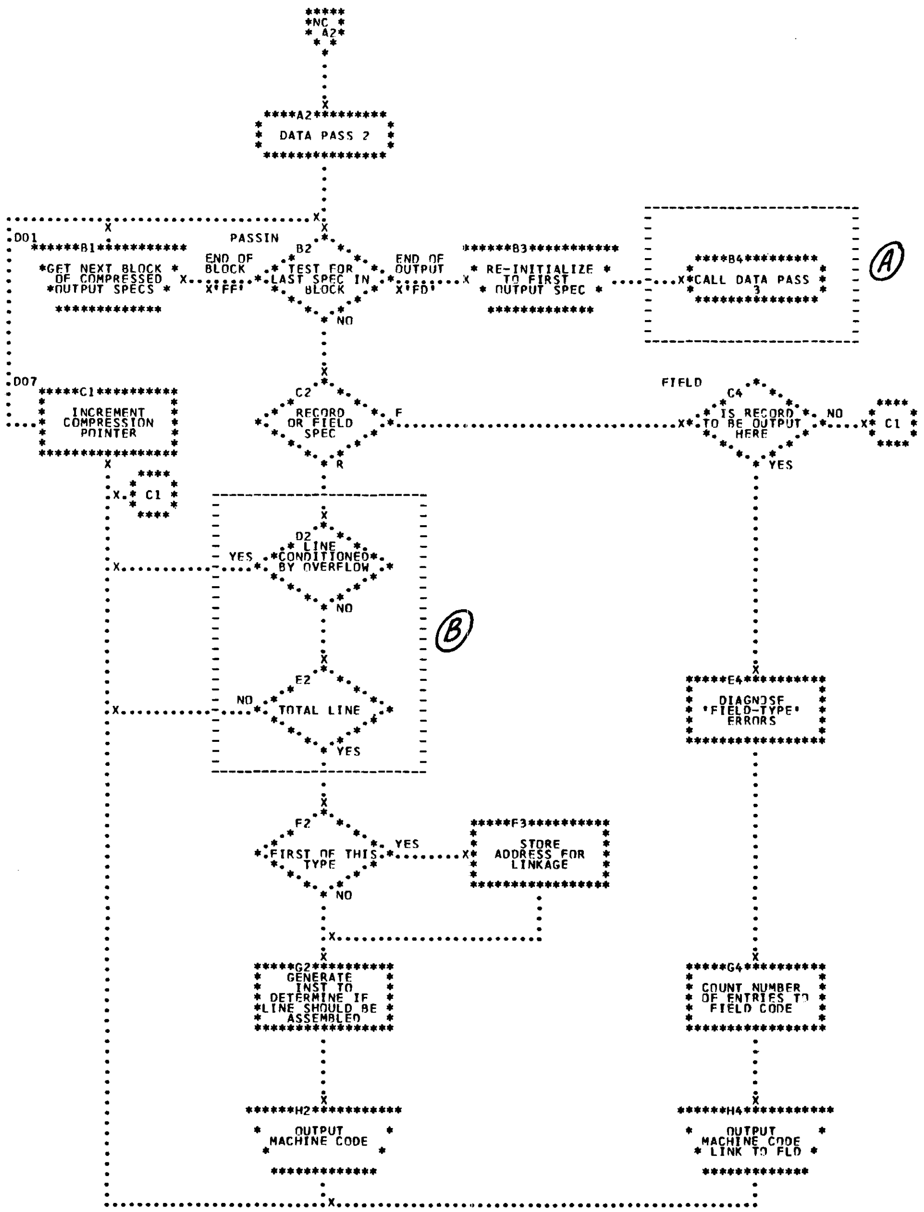

NC A2
A2 DATA PASS 2
PASSIN
D01
B1 GET NEXT BLOCK OF COMPRESSED OUTPUT SPECS
END OF BLOCK X'FF'
B2 TEST FOR LAST SPEC IN BLOCK
END OF OUTPUT X'FD'
B3 RE-INITIALIZE TO FIRST OUTPUT SPEC
B4 CALL DATA PASS 3
A
NO
D07
C1 INCREMENT COMPRESSION POINTER
C1
C2 RECORD OR FIELD SPEC
F
R
FIELD
C4 IS RECORD TO BE OUTPUT HERE
NO
C1
YES
D2 LINE CONDITIONED BY OVERFLOW
YES
NO
B
E2 TOTAL LINE
NO
YES
E4 DIAGNOSE 'FIELD-TYPE' ERRORS
F2 FIRST OF THIS TYPE
YES
F3 STORE ADDRESS FOR LINKAGE
NO
G2 GENERATE INST TO DETERMINE IF LINE SHOULD BE ASSEMBLED
G4 COUNT NUMBER OF ENTRIES TO FIELD CODE
H2 OUTPUT MACHINE CODE
H4 OUTPUT MACHINE CODE LINK TO FLD

Chart NC. Assemble Phase 4

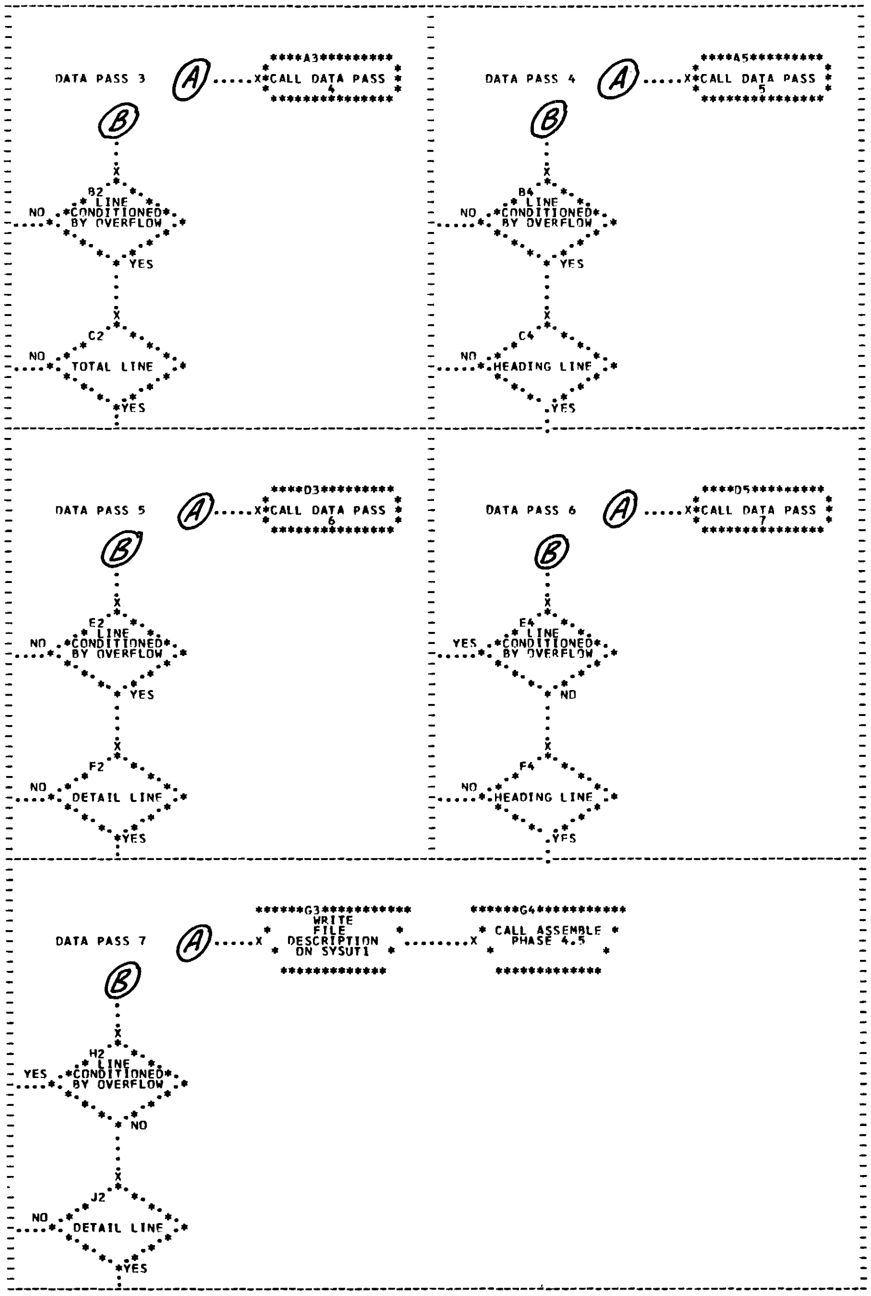


Chart ND. Assemble Phase 4

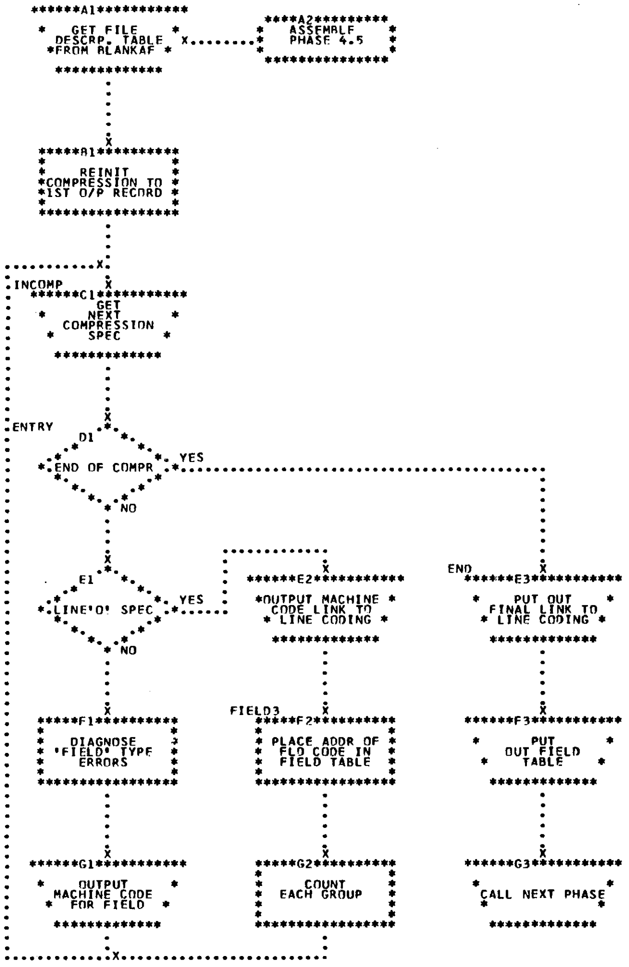

A1
GET FILE DESCRP. TABLE FROM BLANKAF
A2
ASSEMBLE PHASE 4.5
B1
REINIT COMPRESSION TO 1ST O/P RECORD
INCOMP
C1
GET NEXT COMPRESSION SPEC
ENTRY
D1
END OF COMPR
YES
NO
E1
LINE'O' SPEC
YES
NO
E2
OUTPUT MACHINE CODE LINK TO LINE CODING
END
E3
PUT OUT FINAL LINK TO LINE CODING
F1
DIAGNOSE 'FIELD' TYPE ERRORS
FIELD3
F2
PLACE ADDR OF FLD CODE IN FIELD TABLE
F3
PUT OUT FIELD TABLE
G1
OUTPUT MACHINE CODE FOR FIELD
G2
COUNT EACH GROUP
G3
CALL NEXT PHASE

Chart NE. Assemble Phase 4.5

increments the displacement register for a line, and increments the UBA counter

SETCOD

- Sets up and issues the user object code for linking to the I/O intercept routine

OVFSET

- Sets the overflow switch on/off from the X-switch setting

UBADEF

- Linkage to the UBA-defining subroutine (ADRDEF)

ERROUT

- Linkage to CIOEX to print the diagnostic codes

END5

- Executed at the end of the pass that processes a detail line conditioned by overflow
- Issues code for a printer carriage automatic skip to channel 1 and for the end of the first object code output block
- Updates the address counter to include the last code block
- Issues code for the first entry into a detail line code block
- Writes the linkage information on the RLD data set
- Issues code to load and store registers and store constants
- Resets X-switches for all print lines

END2

- Executed at the end of the pass that processes a total line not conditioned by overflow
- Issues code for testing overflow switches for print lines

END

- Executed at the end of Pass 1 to put out the file description table and the assemble 4 table on the blank-after data set which are used by Phase 4.5

DMPTXT

- Puts out the text card images for the object program
- Described in more detail in Assemble Phase 2

PHASE 4.5 ROUTINES

START

- Reinitializes the compression for

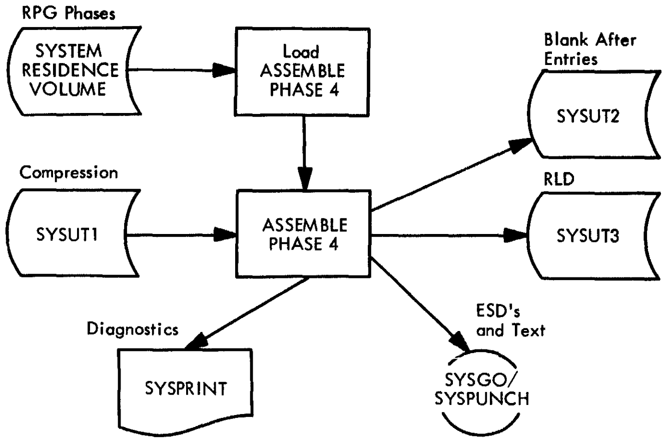


Figure 36. Assemble Phase 4 Input/Output Flow

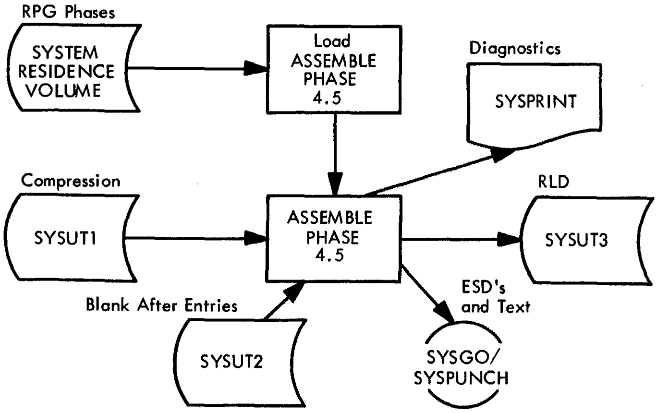


Figure 37. Assemble Phase 4.5 Input/ Output Flow

the next pass through

- Brings in the file description table and the assemble 4 table address for use in this part of the program

ENTRY

- Examines the next compression
- Checks for end-of-compression block
- Determines which path to follow for specific record type

END

- Executed at the end of Phase 4.5 to put out any remaining text, the last RLDX record, the last branch address
- Calls I/O Phase 1

TSTNAM

- Processes the M-type records by generating object code to process all fields, including edit words, floating dollar sign, status sign, numeric, alphameric, sterling, zero suppress, and constants

STERL

- Performs diagnostics on sterling fields
- Sets up the code for linking to the sterling conversion routine
- Determines the correct lengths and addresses for sterling fields

F10

- Performs diagnostics on the position of the field within the line
- Issues the code for testing the resulting indicators, conditioning a field, handling page fields, and providing linkage to the sterling conversion routine

BLNKAF

- Issues the code for a blank-after operation for a numeric or alphameric field

DATOV1

- Issues the code to load the correct value if the present field address overflow key has changed from that of the previous key

INTRODUCTION

The purpose of the I/O phases is to generate the machine code and parameters (I/O interface) necessary for the execution of the input/output of data records at object time.

The I/O interface, illustrated in Figure 38, consists of

1. The data control blocks (DCB) and data event control blocks (DECB) required for input/output operations under the control of the operating system
2. The master routine (IORPG) and the interpretive routines necessary to process and execute the input/output requests from the RPG object program
3. An input/output request block (IORB) to act as the interface between the RPG object program and the RPG I/O interpretive routines
4. Pointer tables for DCBs, work areas, interpretive routines, and IORBs

The logic required to create the I/O interface is contained in two phases that are separately loaded and executed. Figures 39 and 40 and Charts OA and OC illustrate the organization and operation of I/O Phases 1,2. Figures 41 and 42 illustrate the input/output flow for the two I/O phases.

LOGIC (PHASE 1)

The functions of Phase 1 are

1. Build and put out the DCBs and the work areas
2. Locate and put out the line counter tables
3. Build and put out the IORBs
4. Put out the linkage vector entries

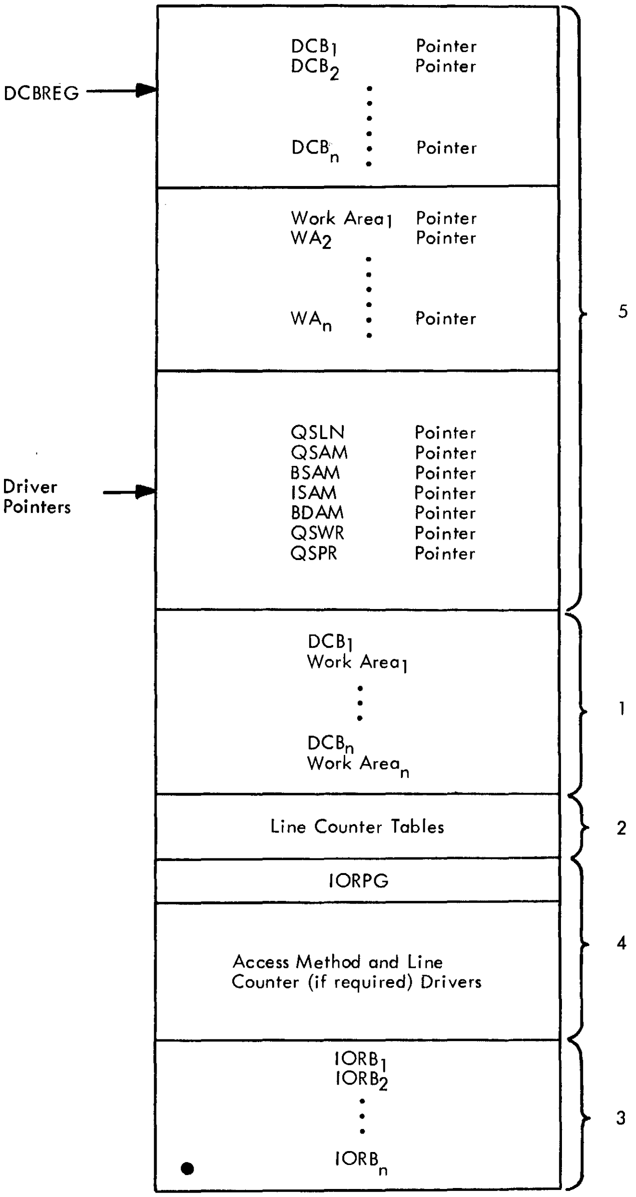
The DCBs created correspond to the access method used in processing the file. The access methods used are BSAM for combined files, ISAM for indexed-sequential files, BDAM for files with a direct organization and QSAM for sequential files other than combined files.

The record work area for each file is equal in length to the longest logical record plus additional bytes as required for variable length files and control characters.

LOGIC (PHASE 2)

The functions of Phase 2 are

1. Input the DCBs, work areas, and IORB pointer tables
2. Put out the master and interpretive access method routines



NOTE: Map is shown as located in core
Numbers represent sequence of generation

Figure 38. Organization of I/O Interface

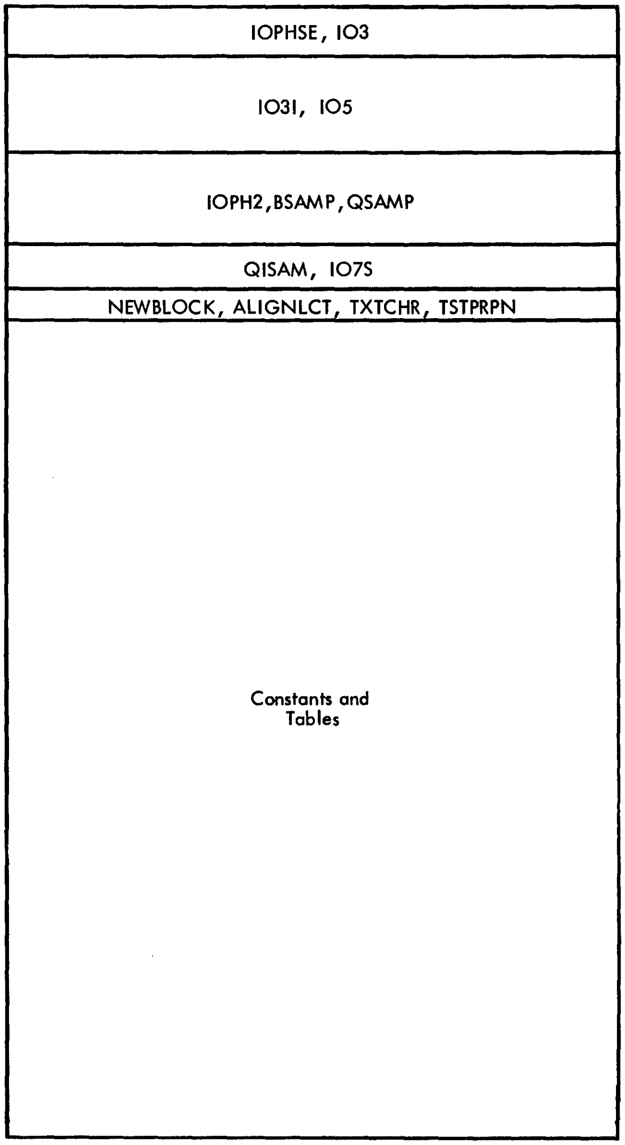


Figure 39. I/O Phase 1 Storage Allocation Map

3. Put out the pointer tables

The pointer tables consist of the DCB addresses at object time and the addresses of the IORPG access method routines and the line counter subroutine.

PHASE 1 MAIN ROUTINES

<u>IOPHSE</u>

- Saves registers

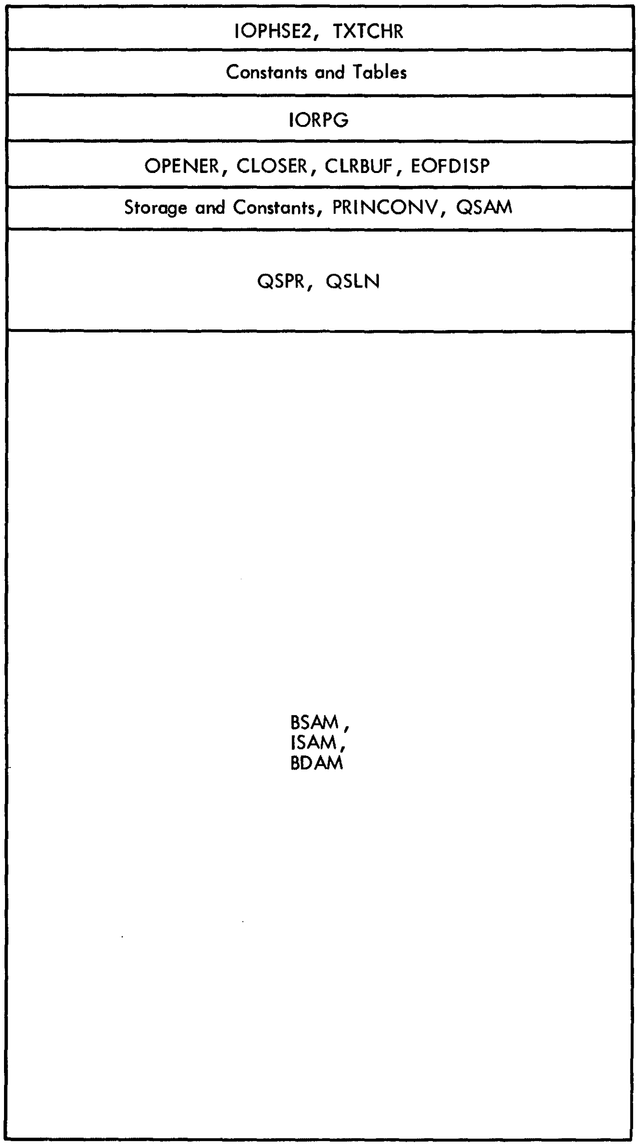

IOPHSE2, TXTCHR
Constants and Tables
IORPG
OPENER, CLOSER, CLRBUF, EOFDISP
Storage and Constants, PRINCONV, QSAM
QSPR, QSLN
BSAM,
ISAM,
BDAM

Figure 40. I/O Phase 2 Storage Allocation Map

- Initializes compression
- Aligns the location counter
- Initializes pointer vectors

<u>IO3</u>

- Locates line counter tables
- Puts out line counter tables as text

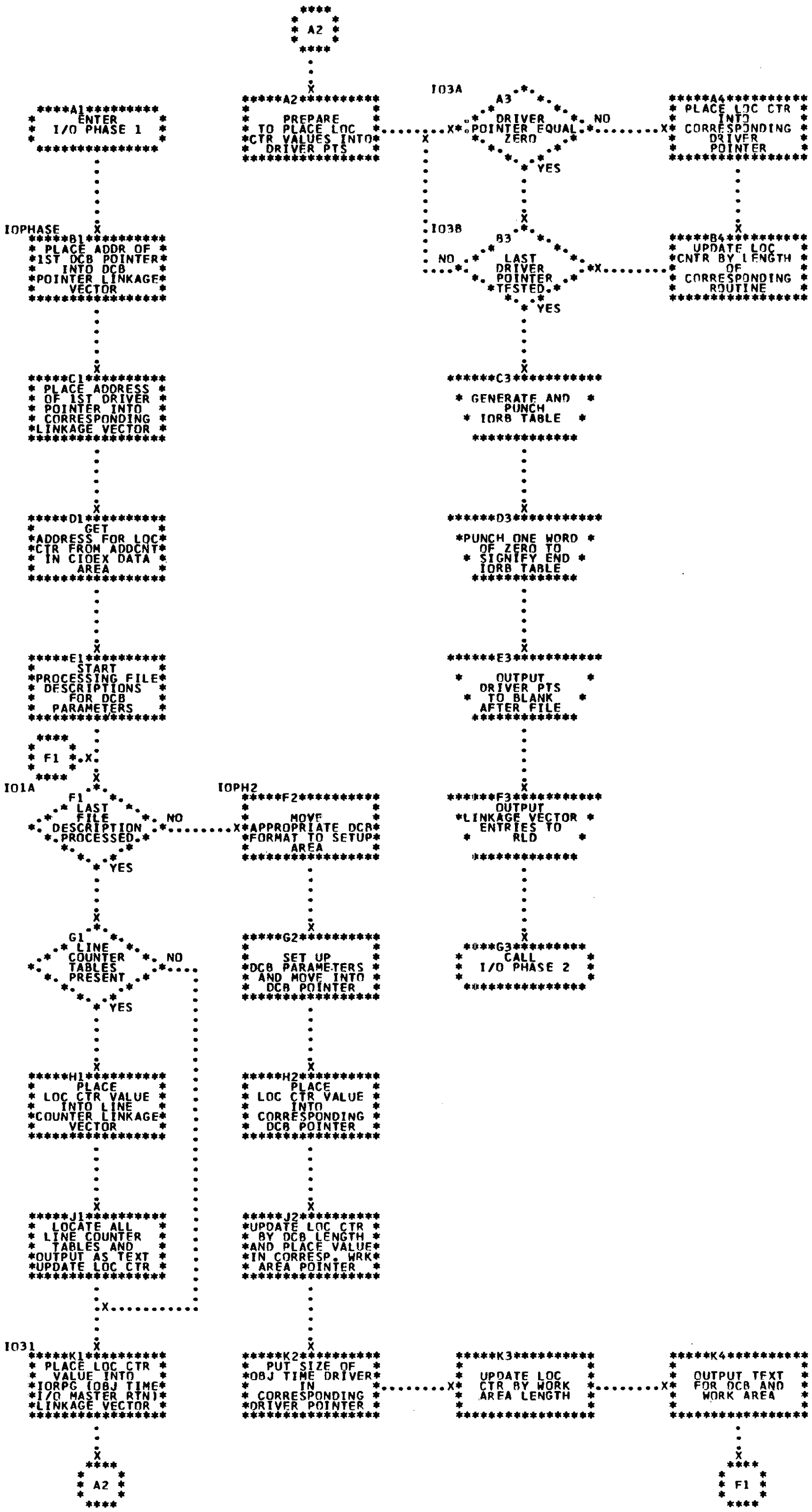

A2
A1
ENTER I/O PHASE 1
A2
PREPARE TO PLACE LOC CTR VALUES INTO DRIVER PTS
IO3A
A3
DRIVER POINTER EQUAL ZERO
NO
YES
A4
PLACE LOC CTR INTO CORRESPONDING DRIVER POINTER
IOPHASE
B1
PLACE ADDR OF 1ST DCB POINTER INTO DCB POINTER LINKAGE VECTOR
IO3B
B3
LAST DRIVER POINTER TESTED
NO
YES
B4
UPDATE LOC CNTR BY LENGTH OF CORRESPONDING ROUTINE
C1
PLACE ADDRESS OF 1ST DRIVER POINTER INTO CORRESPONDING LINKAGE VECTOR
C3
GENERATE AND PUNCH IORB TABLE
D1
GET ADDRESS FOR LOC CTR FROM ADDCNT IN CIOEX DATA AREA
D3
PUNCH ONE WORD OF ZERO TO SIGNIFY END IORB TABLE
E1
START PROCESSING FILE DESCRIPTIONS FOR DCB PARAMETERS
E3
OUTPUT DRIVER PTS TO BLANK AFTER FILE
F1
IO1A
F1
LAST FILE DESCRIPTION PROCESSED
NO
YES
IOPH2
F2
MOVE APPROPRIATE DCB FORMAT TO SETUP AREA
F3
OUTPUT LINKAGE VECTOR ENTRIES TO RLD
G1
LINE COUNTER TABLES PRESENT
NO
YES
G2
SET UP DCB PARAMETERS AND MOVE INTO DCB POINTER
G3
CALL I/O PHASE 2
H1
PLACE LOC CTR VALUE INTO LINE COUNTER LINKAGE VECTOR
H2
PLACE LOC CTR VALUE INTO CORRESPONDING DCB POINTER
J1
LOCATE ALL LINE COUNTER TABLES AND OUTPUT AS TEXT UPDATE LOC CTR
J2
UPDATE LOC CTR BY DCB LENGTH AND PLACE VALUE IN CORRESP. WRK AREA POINTER
IO31
K1
PLACE LOC CTR VALUE INTO IORPG (OBJ TIME I/O MASTER RTN) LINKAGE VECTOR
K2
PUT SIZE OF OBJ TIME DRIVER IN CORRESPONDING DRIVER POINTER
K3
UPDATE LOC CTR BY WORK AREA LENGTH
K4
OUTPUT TEXT FOR DCB AND WORK AREA
A2
F1

Chart OA. I/O Phase 1

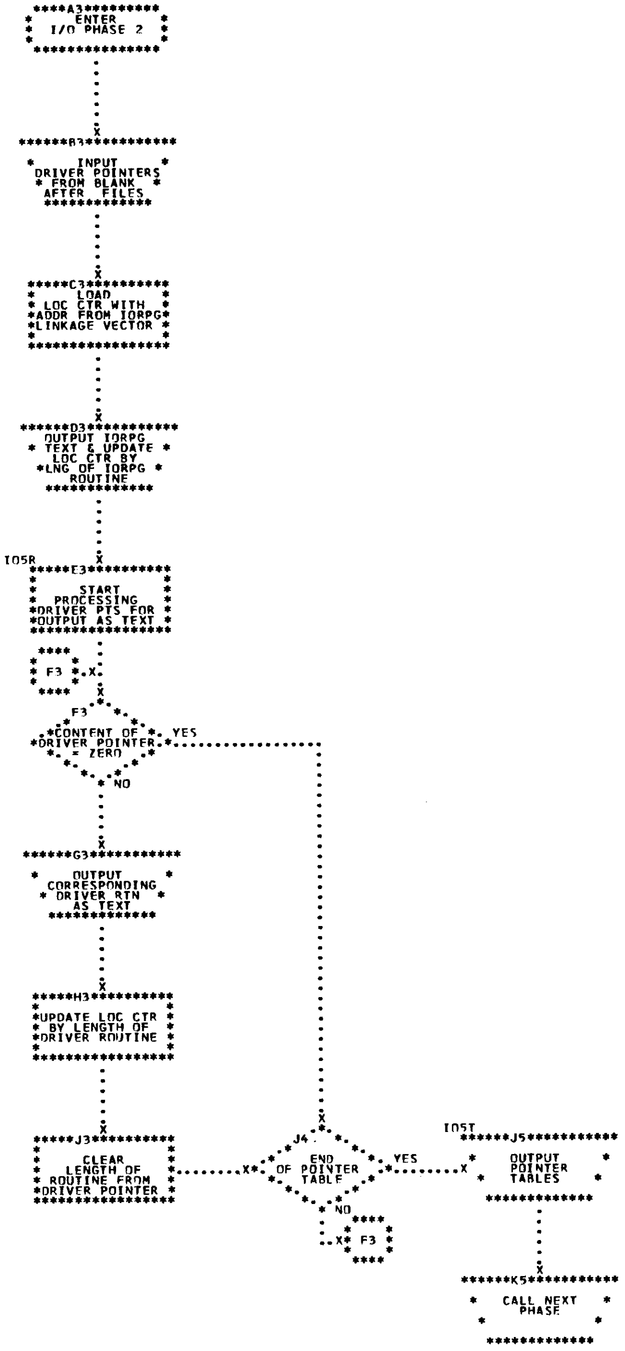

A3
ENTER I/O PHASE 2
B3
INPUT DRIVER POINTERS FROM BLANK AFTER FILES
C3
LOAD LOC CTR WITH ADDR FROM IORPG LINKAGE VECTOR
D3
OUTPUT IORPG TEXT & UPDATE LOC CTR BY LNG OF IORPG ROUTINE
IO5R
E3
START PROCESSING DRIVER PTS FOR OUTPUT AS TEXT
F3
F3
CONTENT OF DRIVER POINTER = ZERO
YES
NO
G3
OUTPUT CORRESPONDING DRIVER RTN AS TEXT
H3
UPDATE LOC CTR BY LENGTH OF DRIVER ROUTINE
J3
CLEAR LENGTH OF ROUTINE FROM DRIVER POINTER
J4
END OF POINTER TABLE
YES
NO
F3
IO5T
J5
OUTPUT POINTER TABLES
K5
CALL NEXT PHASE

Chart OB. I/O Phase 2

IO3I

- Generates IORB
- Puts out IORB tables
- Puts out driver pointers

IOPH2

- Inserts parameters into DCB skeleton
- Reserves work area
- Determines the access method(s) required
- Controls access method processors

BSAMP

- Basic sequential access method processor (BSAM) for combined files only

QSAMP

- Queued sequential access method processor (QSAM) for other than combined files

QISAM

- Indexed-sequential access method processor (ISAM)

BDAM

- Basic direct access method processor (BDAM)

PHASE 2 MAIN ROUTINES

IOPHSE2

- Inputs driver pointers
- Puts out the object time I/O processor (IORB)
- Puts out the pointer tables
- Calls the next phase

SUBROUTINES

TXTCHR

- Puts out text card images

NEWBLOCK

- Gets a block of compression

ALIGNLCT

- Aligns the location counter on a double word boundary

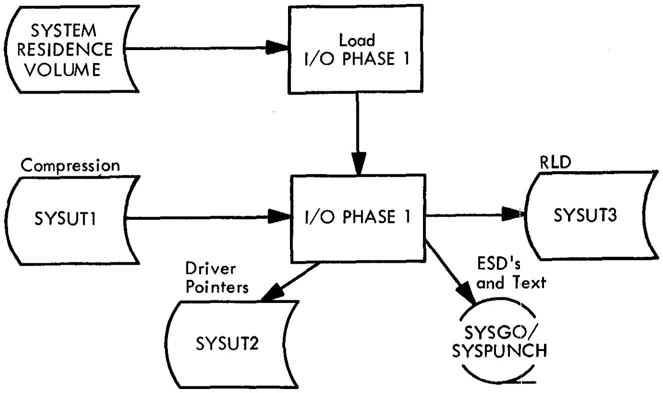


Figure 41. I/O Phase 1 Input/Output Flow

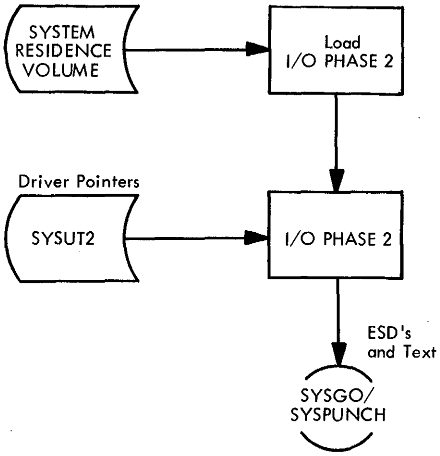


Figure 42. I/O Phase 2 Input/Output Flow

TSTPRPN

- Tests device entry in compression for a printer or punch indication and sets corresponding bits in IORB

PRECODED ROUTINES

IORPG

- Start of object module execution
- IORB first pass processor
- Sets up parameters for data set OPEN
- Branches to OPEN routine

I20

- Determines access method address
- Branches to access method processor

OPENER

- Opens data sets
- Clears work areas

CLOSER

- Closes all data sets

CLRBUF

- Clears an area to blanks
- Register 5 contains address of 1st character
- Register 12 contains the field length
- Registers 6 and 14 are entry and exit registers

PRINCONV

- Extracts printer commands from IORB
- Converts commands to control characters
- Stores control characters for use by the write and line counter routines

EOFDISP

- The end of data routine that sets the end-of-file flag in the IORB

QSAM

- GET/PUT for sequential files

BSAM

- Combined files

ISAM

- Indexed-sequential files

BDAM

- Direct access files

QSLN

- Line counter routine (appended to QSAM)

QSWR

- Write routine (appended to QSAM)

QSPR

- Print routine (appended to QSAM)

INTRODUCTION

The function of the diagnostic phases is printing the diagnostic code messages that correspond to the codes assigned during all of the previous phases.

Figure 43 and Chart PA illustrate the input/output flow and the operation of the diagnostic phases.

LOGIC

A record of all of the codes that have been assigned is in the CIOEX data area. This record is a 256-bit position field in which the bit that corresponds to each numbered code assigned is turned on. Thus, this record serves as an index from which the appropriate messages for printing are selected.

Five separate core loads are required to store the 256 messages. Some codes are provided but undefined and, if used, cause the message, NO ERROR MESSAGE ASSIGNED FOR THIS NOTE, to be printed. The program logic is loaded concurrently with each set of messages.

Each set (a program plus messages) uses CNTRG2 to keep track of the bit to be analyzed in the error byte and CNTREG to control the number of bytes analyzed for each program. When all of the bits in the bytes applicable to the program in core have been scanned, the next set in numerical sequence is brought in. In this manner the linkage phase is called when all messages have been printed.

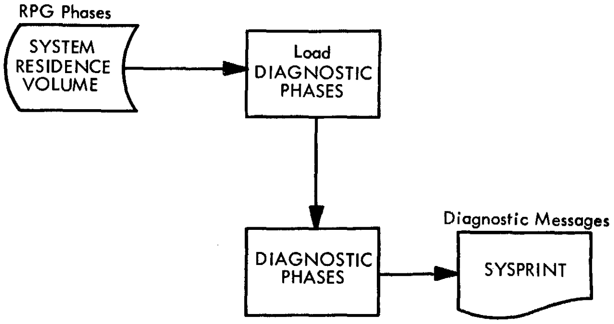


Figure 43. Diagnostic Phases Input/Output Flow

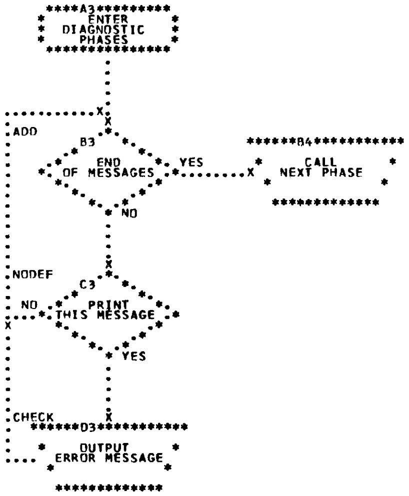


Chart PA. Diagnostic Phases

INTRODUCTION

The linkage phase has several functions:

1. Put out all of the subroutines to be used by the RPG object program (sterling conversion, table look up, test zone, set indicator, sign check)
2. Create linkage for the execution, at object time, of the routines compiled by the previous phases
3. Put out the linkage routine and linkage vector
4. Put out the memory map
5. Put out the RLD card images
6. Put out the END card image

LOGIC

The subroutines listed in 1 above are loaded in object code form with the logic of the phase. The logical functions of these routines are described briefly under the heading Linkage Phase Precoded Routines.

As the routines to accomplish the functions specified in the RPG source program are developed, the locations of these routines are stored as V-type (Appendix F) records on the work data set. The linkage phase uses those records to create a predetermined order of execution and the linkage vector. The linkage vector is a table which provides the absolute addresses of the routines to be executed at object time.

The memory map consists of a printout of the name of each routine used by the RPG object program and the absolute hexadecimal address of the routine at object time. Figure 44 is a sample of a memory map print-out. Table 6 is a summary of the various entries that can appear in the memory map. The addresses are not relocated; thus the relocation factor from LINKEDIT must be added to obtain the routine address. The relocation factor of the RPG object program can be obtained by adding X'20' to the starting address of the corresponding request block shown on the ABEND dump. The request block to be used is listed under the ACTIVE RBS section of the ABEND dump and has the name previously given to the object program load module.

The work data set that contains the V-type entries also contains R- (relocation dictionary) type records (Appendix F). Card images are created to provide this information to the Operating System at object time.

Figure 45 illustrates a storage allocation map for the logic of the linkage phase plus the precoded subroutines stored for output.

Figure 46 illustrates the input/output flow for the linkage phase. Chart ZA illustrates the operation of the linkage phase.

MAIN ROUTINES

ASSEM6

- Puts out (on the go/punch data set) each subroutine (object code form) that has been called for by the other phases. Possible routines are as follows:

1. Sterling conversion - input and output
2. Table look-up
3. Punch
4. Test zone and decimal
5. Set indicator
6. Sign check

The linkage routine (object code form) is output (on the go/punch data set) along with the linkage vector. The linkage vector is a table of absolute addresses of all of the routines used in the object program.

This routine also prints the memory map (Figure 44).

DR

- Processes the remaining records of the RLD data set; that is, the R- type records (Appendix F)

- Puts out as many card images to the go/punch data set as are necessary to contain all of the RLD entries

- Puts out the END card image to the go/punch data set

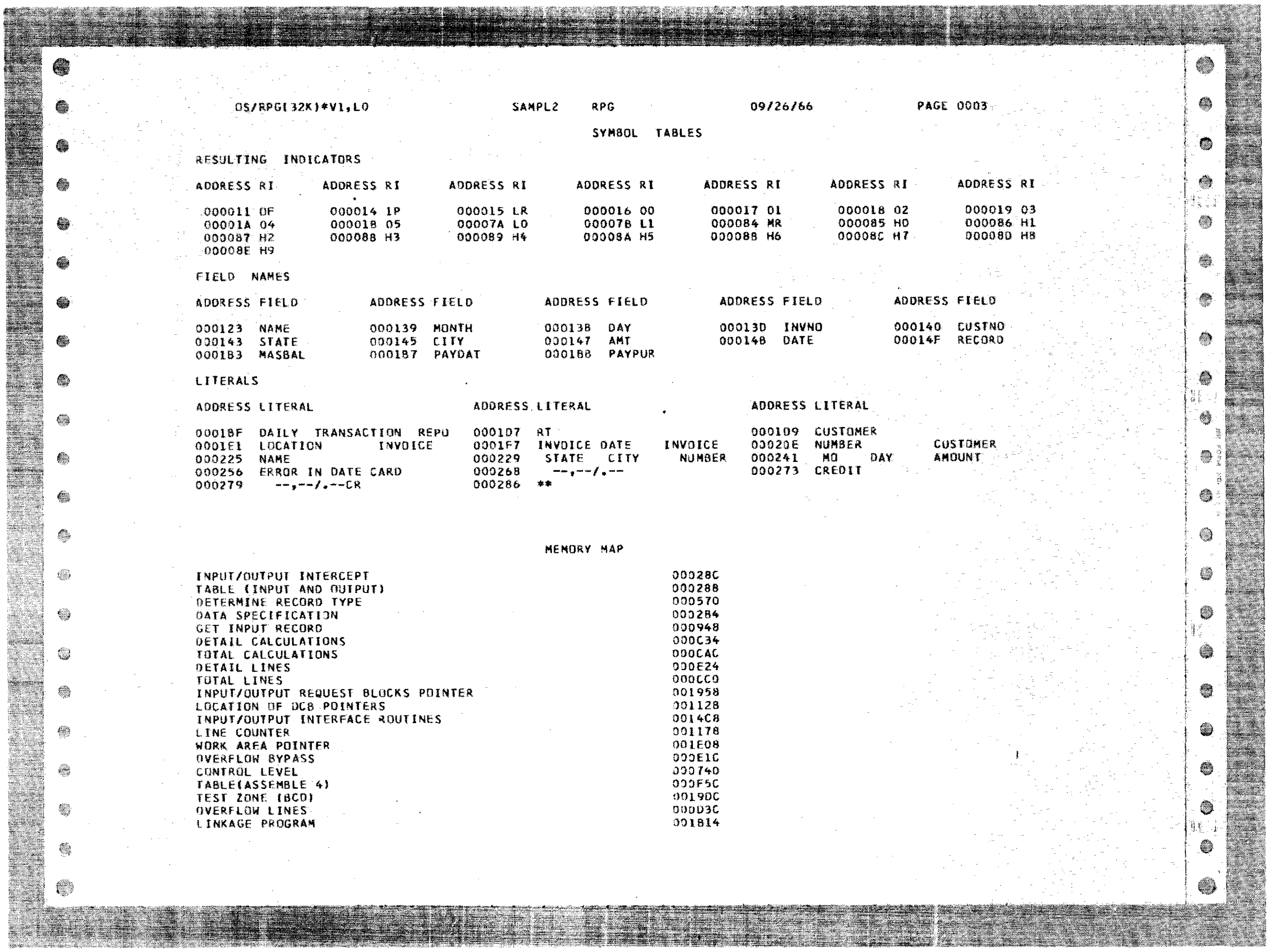
- Calls the terminal phase

SUBROUTINES

The following subroutines exit via register 15 unless stated otherwise.

PLINK

- Used by the major portion of the phase to prepare output for the go/punch data set



```
      OS/RPG(32K)*V1,LO                    SAMPL2    RPG                    09/26/66                PAGE 0003
                                                        SYMBOL  TABLES
  RESULTING  INDICATORS

  ADDRESS RI        ADDRESS RI        ADDRESS RI        ADDRESS RI        ADDRESS RI        ADDRESS RI        ADDRESS RI

   000011 0F         000014 1P         000015 LR         000016 00         000017 01         000018 02         000019 03
   00001A 04         000018 05         00007A L0         00007B L1         000084 MR         000085 H0         000086 H1
   000087 H2         000088 H3         000089 H4         00008A H5         00008B H6         00008C H7         00008D H8
   00008E H9

  FIELD  NAMES

  ADDRESS FIELD            ADDRESS FIELD            ADDRESS FIELD            ADDRESS FIELD            ADDRESS FIELD

  000123  NAME             000139  MONTH            00013B  DAY              00013D  INVNO            000140  CUSTNO
  000143  STATE            000145  CITY             000147  AMT              000148  DATE             00014F  RECORD
  0001B3  MASBAL           000187  PAYDAT           0001B8  PAYPUR

  LITERALS

  ADDRESS LITERAL                         ADDRESS LITERAL                             ADDRESS LITERAL

  00018F  DAILY  TRANSACTION  REPU        000107  RT                                  000109  CUSTOMER
  0001E1  LOCATION        INVOICE         0001F7  INVOICE DATE     INVOICE            00020E  NUMBER           CUSTOMER
  000225  NAME                            000229   STATE   CITY      NUMBER           000241   MO    DAY     AMOUNT
  000256  ERROR IN DATE CARD              000268    --,--/.--                         000273  CREDIT
  000279    --,--/.--CR                   000286  **



                                                   MEMORY MAP

  INPUT/OUTPUT INTERCEPT                                               00028C
  TABLE (INPUT AND OUTPUT)                                             000288
  DETERMINE RECORD TYPE                                                000570
  DATA SPECIFICATION                                                   000284
  GET INPUT RECORD                                                     000948
  DETAIL CALCULATIONS                                                  000C34
  TOTAL CALCULATIONS                                                   000CAC
  DETAIL LINES                                                         000E24
  TOTAL LINES                                                          000CC0
  INPUT/OUTPUT REQUEST BLOCKS POINTER                                  001958
  LOCATION OF DCB POINTERS                                             001128
  INPUT/OUTPUT INTERFACE ROUTINES                                      0014C8
  LINE COUNTER                                                         001178
  WORK AREA POINTER                                                    001E08
  OVERFLOW BYPASS                                                      000E1C
  CONTROL LEVEL                                                        000740
  TABLE(ASSEMBLE 4)                                                    000F5C
  TEST ZONE (BCD)                                                      0019DC
  OVERFLOW LINES                                                       000D3C
  LINKAGE PROGRAM                                                      001814
```

Figure 44. Sample Print-Out of a Memory Map

- Stores information for punch output in an area designated TEX, the lower limit of which is in register 9 and the upper limit, register 3

<u>HEXIT</u>

- Unpacks and converts, byte-by-byte a given absolute address, stored in register 14, to printable hexadecimal form and replaces it in register 14

<u>OUT</u>

- Stores and restores registers 6 and 7
- Branches to the interface program

PRECODED ROUTINES

<u>BEGIN1</u>

- Linkage routine for the RPG object program
- Computes the displacement factor and adds it to the linkage factor and to the entries in the linkage vector
- Establishes the linkage to each of the routines contained in the object program
- The routines are as follows:

  1. Open
  2. Table

Table 6. Summary of Entries Appearing on a Memory Map

Name of Routine	Generated by Phase:	*Comments
Input/Output Intercept	Assemble Phase 1	Determines if disks (DASD) are used and establishes linkage to the RAF support or the input/output routine.
Record Address File	Assemble Phase 1	Determines the disk addresses to be used.
Chaining	Assemble Phase 1	Provides chaining capability for disk records.
Table (Input and Output)	Assemble Phase 1	Reads input and creates a user table.
Sterling Input Conversion	Linkage Phase	
Sterling Output Conversion	Linkage Phase	
Determine Record Type	Assemble Phase 2	
Data Specification	Assemble Phase 2	
Get Input Record	Assemble Phase 2	
Detail Calculations	Assemble Phase 3	
Total Calculations	Assemble Phase 3	
Detail Lines	Assemble Phase 4	Not conditioned by overflow.
Total Lines	Assemble Phase 4	Not conditioned by overflow.
Input/Output Request Blocks Pointer	I/O Phase 1	Unrelocated address points to first IORB. [File No. (1-10)-1] 32=Logical Record Address
Location of DCB Table Pointers	I/O Phase 2	Unrelocated address points to list of DCB pointers (1-10).
Input/Output Interface Routines	I/O Phase 2	Main input/output routine.
Table Look-Up	Linkage Phase	Table look-up routine.
Testz Routine	Linkage Phase	Routine for Calculation specifications.
Line Counter	I/O Phase 2	Line Counter routine.
Line Counter Table	I/O Phase 1	
Work Area Pointer	Linkage Phase	A scratch work area, internal to RPG.
Overflow Bypass	Assemble Phase 4	Address points to first instruction not involved in overflow coding of Output-Format specifications.
Set Indicator Routine	Linkage Phase	
Sign Check Routine	Linkage Phase	
Control Level	Assemble Phase 2	
Table (Assemble 4)	Assemble Phase 4	A linkage table which permits interchange between lines and fields of Output-Format specifications.
Test Zone (BCD)	Assemble Phase 2	RPG routine to test Input specifications.
Overflow Lines	Assemble Phase 4	Address points to the first instruction for lines conditioned by overflow.
Alternating Sequence	Linkage Phase	
Linkage Program	Linkage Phase	

*NOTE: The address (unrelocated) that appears on the Memory Map points to the first byte of the routine or table unless otherwise clarified in the comments.

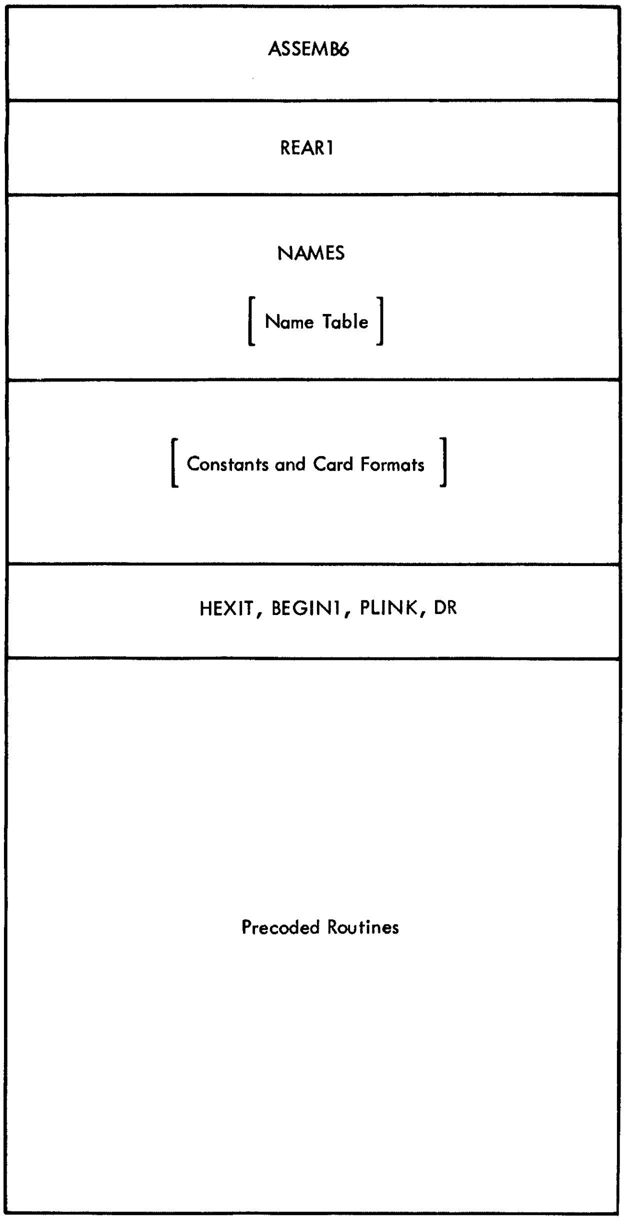


Figure 45. Linkage Phase Storage Allocation Map

3. Detail lines
4. Get input
5. Total calculation
6. Detail calculation
7. Data specification
8. Close

STEROT

- Sterling output conversion routine

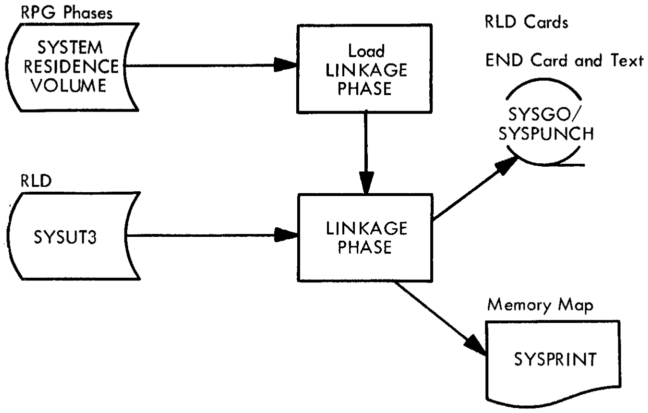


Figure 46. Linkage Phase Input/Output Flow

- Converts a packed pence field with a maximum of eight positions to a pounds-pence-decimal pence field configuration

- Maximum of three decimal positions can be specified

STERLIN

- Sterling input conversion routine

- Converts a shilling-pound-pence-decimal pence field to a packed all-pence field

SETIND

- Indicator turn-on routine

- Turns on and off indicators representing plus, minus, and zero

TESTZ

- Test zone routine

- Turns on the plus indicator for a twelve punch, the minus indicator for an eleven punch, and the zero indicator for any other punch detected

SIGNCHK

- Sign check routine for a numeric field

- Considers the sign to be minus when bits 4-7 contain D, B, 6; otherwise the sign is considered to be plus

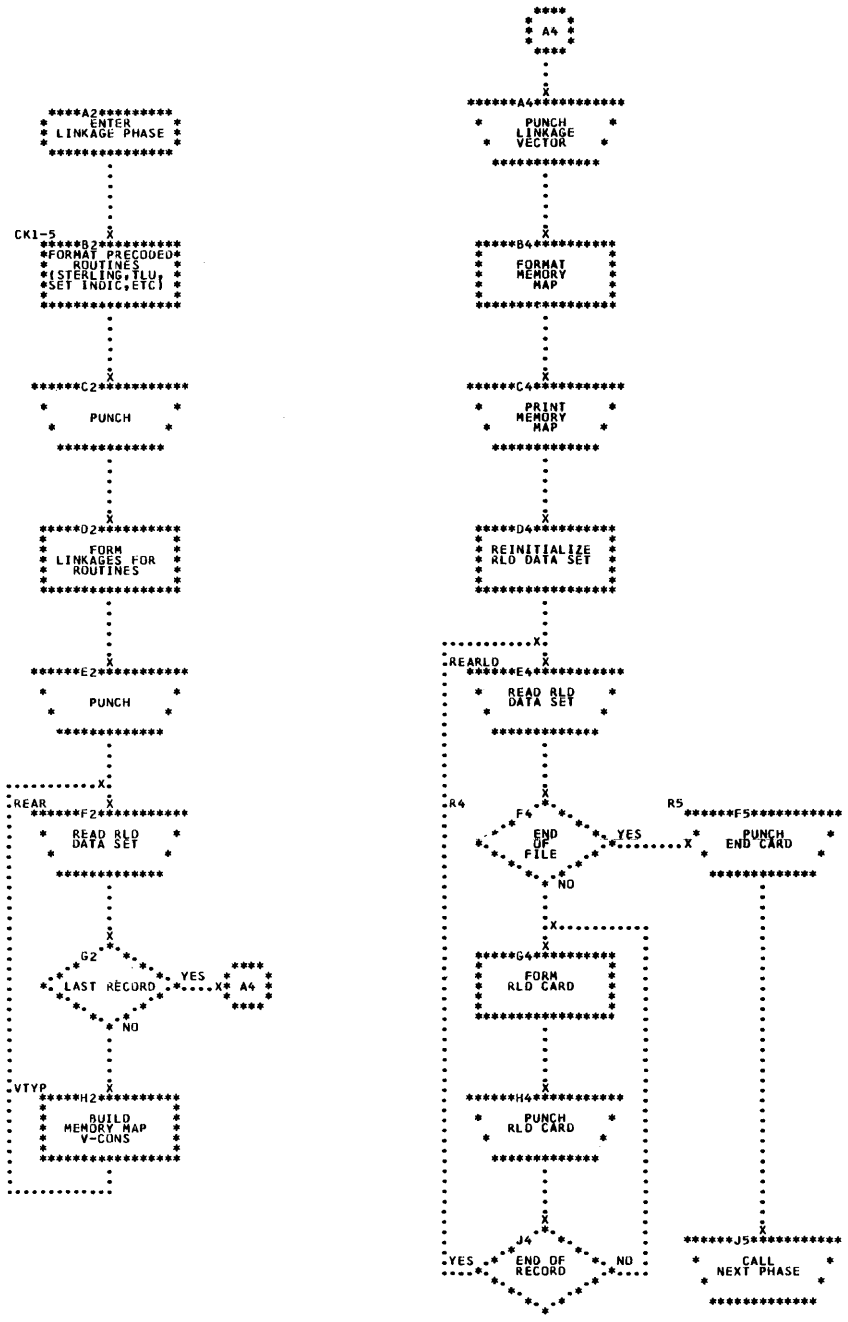


Chart ZA. Linkage Phase

TERMINAL PHASE

INTRODUCTION

As the last logical element of the RPG compiler program, Terminal Phase closes the compiler data sets (SYSIN, SYSPRINT, SYSGO/SYSPUNCH, SYSUT1, SYSUT2, SYSUT3). A separate phase is provided for this function because of the large amount of work area required by the parallel CLOSE of six data sets.

This phase sets a switch in the CIOEX data area that signals to the resident phase that a return to the supervisor is requested.

Figure 47 and Chart ZM illustrate the organization and operation of Terminal Phase. Figure 48 illustrates the input/output flow for this phase.

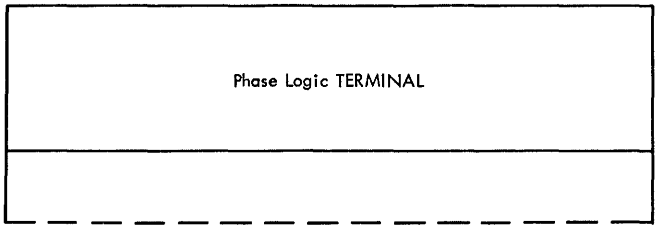


Figure 47. Terminal Phase Storage Allocation Map

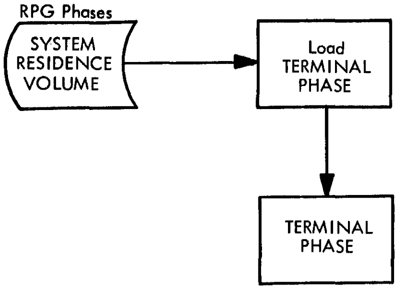


Figure 48. Terminal Phase Input/Output Flow

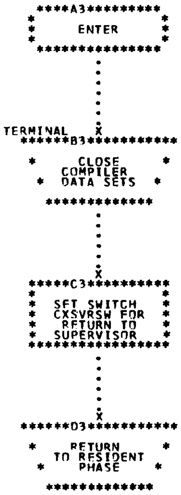


Chart ZM. Terminal Phase

Each entry in the File Name Table is 12 bytes long and the table can contain 10 entries. If the number exceeds 10 (overflows), Enter Phase 1 treats the additional entries as comments and prints an error. Each entry in the File Name Table contains four parts: filename (bytes 1 - 8), reference byte (byte 9), generated unpacked sequence number (bytes 10-11), and type or usage of the file (byte 12).

Before an entry is placed in the table, the table is searched to determine if the entry is already present. If the entry is not in the table it is added and the reference byte is created as a blank. If the entry is in the table the reference byte remains as is, but a diagnostic is printed indicating multidefined file.

TABLE ENTRY FORMAT

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	x	x	x	x	x	x	x	r	s	s	t	ℓ	ℓ	ℓ	ℓ

x - filename

r - reference byte

Type Specification	Reference Byte Encountered as:				Filename Not in Table
	Blank	E	L	R	
File Description	Leave as Blank	N/A	N/A	N/A	Create as a blank
File Extension	Change to E	Leave as E	N/A	N/A	Make Diagnostic Message
Line Counter	Change to L	Change to L	Leave as L	N/A	Make Diagnostic Message
Input	Change to R	Change to R	N/A	Leave as R	Make Diagnostic Message

s - generated sequence number in unpacked format

t - type byte, the type or usage of file

Type Byte	File Type	*File Designation	Type of File Designation	Method of Processing	
I	Input	P, S	Seq'l or Indexed Seq'l	Sequential	Without Chaining
I	Combined	P, S	Sequential	Sequential	Field
E	Input	P, S	Seq'l or Indexed Seq'l	Sequential	Without Chaining
E	Combined	P, S	Sequential	Sequential	Field
U	Input	P, S	Indexed Sequential	Random (or	Without Chaining
				Between Limits)	Field
U	Update	P, S	Indexed Sequential	Random	
V	Input	P, S	Indexed Sequential	Random (or	With Chaining
				Between Limits)	Field
V	Update	P, S	Indexed Sequential	Random	
Y	Input	P, S	Direct Organization	Random	Without Chaining
Y	Update	P, S	Direct Organization	Random	Field
Z	Input	P, S	Direct Organization	Random	With Chaining
Z	Update	P, S	Direct Organization	Random	Field
O	Output	Blank	Sequential	Sequential	Without Line Counter
P	Output	Blank	Sequential	Sequential	With Line Counter
R	Input	R	Sequential	Sequential	With Extension Code
T	Input	T	Sequential	Sequential	With Extension Code

*P - Primary, S - Secondary, R - Record Address, T - Table, C - Chained

Type Byte	File Type	*File Designation	Type of Designation	Method of Processing	
C	Input	C	Indexed Sequential	Random	Without Chaining Field
C	Update	C	Indexed Sequential	Randon	
D	Input	C	Indexed Sequential	Random	With Chaining Field
D	Update	C	Indexed Sequential	Random	
W	Input	C	Direct Organization	Random	Without Chaining Field
W	Update	C	Direct Organization	Random	
X	Input	C	Direct Organization	Random	With Chaining Field
X	Update	C	Direct Organization	Random	

*P - Primary, S - Secondary, R - Record Address, T - Table, C - Chained.

ℓ - maximum record length (unpacked format)

Each field name entry is 13 bytes long and the table can contain 70 entries (16K system). Each entry in the table consists of three parts: field name, reference, and field length information.

Before an entry is placed in the table, the table must be searched to determine if the entry is already present. If the entry is found, the logical AND of the table name gamma byte (byte 13) and the search name field mask is formed.

If the result is non-zero, the table name is referenced as multidefined, a diagnostic is listed and the specification associated with the search entry is dropped. If the result is zero, the field length attributes of the search entry and the table entry are compared. If the field length attributes (bytes 11-12) are unequal, the reference byte (byte 7) is changed to an M, to indicate multidefined. If the field length attributes are equal (length from table equal length from search or length from search is blank), the reference byte is changed to an R and processing continues.

If the entry is not in the table and the table has not overflowed, the entry is added. If the name is specified in a Result Field, the reference byte is created as a blank. If the name is specified in a Factor 1 or Factor 2, the reference byte is created as U. When the number of entries exceeds the allotted area (overflows), the sequence number of the specification causing the overflow is saved in the CIOEX Data Area and X'FD' is placed in the first byte following the last table entry.

TABLE ENTRY FORMAT

1	2	3	4	5	6	7	8	9	10	11	12	13	(13 byte field)
x	x	x	x	x	x	r	b	b	β	L	D	γ	

x - field name

r - reference byte
b = defined but unreferenced
U= referenced but undefined
R = referenced
M= multidefined

b - blank (X'40'): field address will be placed here by the assign phases

β - field information byte

Bit	Value	Meaning
0 - 3	0000	Always zero
4	0	No blank-after reference
	1	Blank-after reference
5	0	BDDD references field directly
	1	BDDD references table linkage field
6 - 7	00	Table is in ascending order
	01	Table is in descending order
	10	Table is not specified as ascending or descending
	11	Not assigned

L - field length in binary

Source		L = Length Value of Field in Compression	Number of Bytes To be Reserved
Input Specification			
Shilling Numeric	BSI	L = TO-FROM ⌿ 1	[(L ⌿ 1)/2] ⌿ 1
	IBM	L = TO-FROM	[(L ⌿ 1)/2] ⌿ 1
Numeric (unpacked)		L = TO-FROM	[(L ⌿ 1)/2] ⌿ 1
Numeric (packed)		L = 2 (TO-FROM)	[(L ⌿ 1)/2] ⌿ 1
Alphameric		L = TO-FROM	L ⌿ 1

[x] = integer portion of x
L ⌿ 1 = number of digits or characters in the field

FIELD NAME TABLE (CONTINUED)

Source	L = Length Value of Field in Compression	Number of Bytes To Be Reserved
Calculation Specification		
Numeric	L = field length - 1	$[(L + 1)/2] + 1$
Alphameric	L = field length - 1	$L + 1$
File Extension Table		
Numeric (unpacked	L = length of table entry - 1	$[(L + 1)/2] + 1$
Numeric (packed)	L = 2 (length of table entry - 1)	$[(L + 1)/2] + 1$
Alphameric	L = length of table entry - 1	$L + 1$
File Description Specification		
RAF with Conversion	L = length of record address field - 1	$L + 1$
Output Specification		
PAGE (n)	L = 3	$[(L + 1)/2] + 1$

$[x]$= integer portion of x
$L + 1$ = number of digits or characters in the field

D - decimal position (unpacked): X'40' = alphameric
X'F0' - X'F9' = numeric

γ - field type byte

Gamma Byte Value	Field Type	Field Mask*	Meaning
10000000	All other field names (normal)	00110010	The zeros in the masks correspond to the allowed combinations of field types. For example, the mask for Factor 1 of EXTCV, RPGCV is 11011111. This indicates that the field name associated with this mask can be used only to identify a conversion routine. An attempt to use the field name in any other manner would result in the name's being referenced as multidefined.
01000000	Result Field of a KEYCV following an EXTCV	00111011	
00100000	Factor 1 of EXTCV, RPGCV	11011111	
00010000	Factor 2 of EXTCV, EXIT	11101111	
00001000	Factor 2, Result Field of a LOKUP	01110011	
00000100	RLABL	00110011	
00000010	TAG, GOTO	11111101	
00000001	ULABL	01111110	

* When a match is found between a search name and a table name, the logical AND of the table name gamma byte and the search name field mask is formed. If the result is zero, the search name is processed; if the result is nonzero, the table name is referenced as multidefined, and the specification associated with the search name is dropped.

Entries are made into this table from the Calculation Specifications (Factor l and Factor 2) and from the Output-Format Specifications (Constant or Edit word). This table overlays the File Name Table area. There are three types of literal entries possible: numeric literals, alphameric literals, and edit words.

The table entries vary in length. Each entry consists of eight information bytes plus the number of bytes in the actual literal. The number of entries the table can contain depends on the size of the literals entered. The shortest possible entry is ll bytes (literal of three bytes) ; the longest entry is 33 bytes (literal of 25 bytes). The table area is initially filled with X'FE's and terminated by 33 X'FF's.

Before entering a literal in the table, the table must first be searched to determine if the entry is already present or if the table area is already full. A full table is detected by reaching one of the 33 X'FF's at the end of the area without first detecting an X'FE'. If the entry is not present, and the table is not full, it is added. The length values which are placed in the table must be calculated by the phase. During the Enter Phases, if the entry is not present but the table is full, an X'FD' is placed immediately after the end of the last entry, and the literal is not entered. The specification sequence number causing the overflow is saved in the CIOEX Data Area. However, if an X'FD' is found in the table, then a previous specification has caused an overflow. In this case no action is taken. Literals not placed in the table during the Enter Phases because of overflow are handled in Assign Phase 2.

The three types of literal entries are:

Numeric

Entered from a Calculation Specification only. A Factor 1 or Factor 2 may contain a numeric literal with a maximum length of ten unpacked digits. This is packed into the literal table giving a maximum length of six bytes. The minimum length when packed is three bytes. If it is less than three it will be padded. The decimal position is noted in table entry bytes 2 and n+5. The decimal is removed before packing. The sign (if any) must be moved to the units position of the literal before packing.

Alphameric

This may be entered either from the Calculation or the Output-Format Specifications. In the Calculation Specifications it is contained in Factor l or Factor 2 and has a maximum length of eight characters. In the Output-Format Specifications it is contained in the Constant or Edit Word Field and has a maximum length of 24 characters. In both cases the minimum length is three. Entries less than this length are padded. Two adjacent apostrophes indicated on the specifications sheet are reduced to a single apostrophe before the literal is entered in the table.

Edit Words

Entered from the Output-Format Specifications only. Since an extra "fill" character is supplied for use in the EDIT instruction, an edit word may have a maximum length of 25 characters. The minimum length, including the "fill" character, is three bytes. Entries less than this length are padded.

The entry in the literal table differs from the entry on the specification sheet in the following ways:

1. A "fill" character of X'5C'for asterisk protection or X'40' is placed at the start.
2. All blanks or floating dollar signs are replaced by X'20'.
3. A zero or asterisk denoting the end of zero suppression is replaced by X'21'.
4. A dollar sign appearing in column 46 is replaced by X'40'.
5. An ampersand is replaced by X'40' (blank)
6. Two adjacent apostrophes are replaced by an apostrophe.
7. All other characters are entered without change.

After the last source specification is processed, Assign Phase 1 will perform the assigning of machine addresses to the literals in the table. After an address has been computed, its base and displacement will be placed in the table entry in bytes n+ 1 and n+ 2. Byte n+ 5 is changed X'00' - X'09' for numeric entries, X' 0B' for alphameric entries, and X' 0A' for edit words. Byte n+ 6 is filled with the appropriate overflow address factor.

If table overflow occurrs, Assign Phase 2 also performs this function on the table or tables that it creates. (Assign Phase 2 continues to rebuild the table until all literals are accommodated.)

LITERAL TABLE (CONTINUED)

TABLE ENTRY FORMAT

1	2	3	n	n+ 1	n+ 2	n+ 3	n+ 4	n+ 5	n+ 6
N	D	x	x	X'40'	X'40'	X'00'	n	D	X'00'

N (1) - Length of actual literal + 2 (binary). Gives displacement by byte n+ 1.

D (2) - Decimal position or type numeric X'F0' - X'F9'
alphameric X'40'
edit word X'AA'

x(3 thru n) - Actual literal - minimum length is 3 bytes; maximum is 25 bytes. (May contain 1 or 2 pad characters to insure minimum length of 3.)

X'40'(n+ 1; n+ 2) - Always blank . Overlayed with an address during ASSIGN phase.

X'00'(n+ 3) - β - byte. Always X'00' for literal table entry.

n(n+ 4) - Number of positions in actual literal - l (binary) does not include pad characters (if any).

D(n+ 5) - Repeat of information in byte 2.

X'00'(n+ 6) - Always X'00'. Replaced by overflow address factor during Assign Phase I, 2 if storage address is above 20,480.

Each entry consists of 5 bytes and the table can contain 130 entries. Each entry consists of three parts; the resulting indicator, a reference byte, and the address assigned (initialized with X'00' and replaced by an address during the Assign Phase).

This table is pre-loaded with all valid indicators; thus overflow can never occur. If the indicator from a specification is found in the table, the table reference byte is changed to indicate "referenced." This byte is also checked to see if the indicator has been previously defined.

If an indicator is specified but is not found in the Resulting Indicator Table, the indicator is in error; a diagnostic is noted and the indicator is replaced by the L0 indicator.

TABLE ENTRY FORMAT

1	2	3	4	5
i	i	r	a	a

i - resulting indicator

r - reference byte:

The reference byte of the resulting indicator entry is initialized as follows:

Indicator	Reference Byte (Hexadecimal)
MR	00
LR	03
L1-L9	00
H1-H9	03
1P	07
L0	07
OA-OG	00
OV	00
00	07
01-99	00

The reference byte of an indicator will be set to reflect its usage in a specification as follows:

Bits	Value
0-3	0000
4	1 - RLABL Usage 0 - Otherwise
5	0 - Initialize OFF 1 - Initialize ON
6	0 - Not referenced 1 - Referenced
7	0 - Undefined 1 - Defined

a - assigned address

APPENDIX E. COMPRESSIONS

FILE DESCRIPTION COMPRESSION

$FNNNNNNNNTDESF_FBBBLLLPA_1A_1AOK_LK_LK_LK_LCD_1D_1D_1S_DS_DS_DL_BN_LN_LN_LN_LN_LN_LL_LD_L\gamma_LN_EN_EN_EN_EN_EN_EL_ED_E\gamma_E$

- F - Type of Specification
- N - Filename
- T - File Type (I, O, U or C)
- D - File Designation (P, S, C, R, T, or blank)
- E - End of File (E or blank)
- S - Sequence (A, D, or blank)
- F_F - File Format (F or V)
- B - Block Length (in binary format)
- L - Record Length (in binary format)
- P - Mode of Processing (L, R, or blank)
- A_1 - Length of Record Address Field (in packed format) or Overflow Indicator (in unpacked format)
- A - Record Address Type (I, K, or blank)
- O - Type of File Organization (I, D, or blank)
- K_L - Key Location
- C - Extension Code (E, L, or blank)
- D_1 - Device
- S_D - Not Used in OS/RPG
- L_B - Labels (S, E or blank)
- N_L - Name of Label Exit (X'00' if no Label Exit)
- L_L - Binary length of Label Exit address constant (always equal to X'03', one less than actual length)
- D_L - Number of Decimal Positions (always equal to X'40' since Label Exit is always alphameric).
- γ_L - Gamma Byte (always equal to X'10' since Label Exit is always field type "Factor 2 of EXTCV, EXIT")
- N_E - Not Used in OS/RPG
- L_E - Not Used in OS/RPG
- D_E - Not Used in OS/RPG
- γ_E - Not Used in OS/RPG

Minimum Compression Length - 55 bytes

Maximum Compression Length - 55 bytes

LINE COUNTER COMPRESSION

$LFFL_1L_1L_2L_2L_3L_3L_4L_4L_5L_5L_6L_6L_7L_7L_8L_8L_9L_9L_{10}L_{10}L_{11}L_{11}L_{12}L_{12}$

- L - Type of specification
- F - Filename generated (in binary) Sequence Number (from File Name Table)
- $L_1 \ldots L_{12}$ - Line Number of respective Channel (in binary)

Minimum Compression Length - 27 Bytes

Maximum Compression Length - 27 Bytes

FILE EXTENSION COMPRESSION

Chaining File

$ESSNFFTT\alpha NNNNNNLD\gamma$

E - Type of Specification

S - Record Sequence (unpacked format) from columns 15 - 16 of Input Specifications, i.e., AA or 01

N - Number of the Chaining Field (unpacked format) from column 62 of Input Specifications

F - From Filename generated unpacked Sequence Number (from File Name Table)

T - To Filename generated unpacked Sequence Number (from File Name Table)

α - α equal to C indicates conversion routine present
α equal to blank indicates conversion routine not present

N - Name of conversion routine

L - Binary Length of conversion address constant (always equal to X'03', one less than actual length)

D - Number of Decimal Positions (always equal to X'40' since conversion field is always alphameric)

γ - Gamma Byte (always equal to X'20' since conversion is always field type "Factor 1 of EXTCV, RPGCV")

Minimum Compression Length - 9 Bytes

Maximum Compression Length - 18 Bytes

Record Address File

$RFFTTNNNNNNLD\gamma\alpha N_cN_cN_cN_cN_cN_cL_cD_c\gamma_c$

R - Type of Specification

F - From Filename generated unpacked Sequence Number (from File Name Table)

T - To Filename Generated unpacked Sequence Number (from File Name Table)

N - Name of Record Address Field (since the RAF is not described on the input specifications, the entries in the RAF will always be made available in a field called CONTD)

L - Binary Length of Record Address Field CONTD (always equal to one less than actual length as specified in columns 29-30 of the File Description specification)

D - Number of Decimal Positions (always equal to X'40' since Record Address Field CONTD is always alphameric)

γ - Gamma Byte (always equal to X'04' since Record Address Field CONTD is always field type RLABL

α - α equal to C indicates conversion routine present
α equal to blank indicates conversion routine not present

N_c - Name of conversion routine

L_c - Binary Length of conversion address constant (always equal to X'03', one less than actual length)

D_c - Number of Decimal Positions (always equal to X'40' since conversion field is always alphameric)

γ_c - Gamma Byte (always equal to X'20' since conversion is always field type "Factor 1 of EXTCV, RPGCV"

Minimum Compression Length - 15 Bytes

Maximum Compression Length - 24 Bytes

FILE EXTENSION COMPRESSION (CONTINUED)

Table File

$$\mathrm{TFFOOBBBEEEEN_1N_1N_1N_1N_1N_1\beta_1L_1D_1\gamma_1P_1S_1N_2N_2N_2N_2N_2N_2\beta_2L_2D_2\gamma_2P_2S_2}$$

T - Type of Specification

F - From Filename generated unpacked Sequence Number (from File Name Table)

O - To Filename generated unpacked Sequence Number (from File Name Table) or blank

B - Number of Table Entries Per Record (unpacked format)

E - Number of Entries Per Table (unpacked format)

N_1 - Name of Table 1

β_1 - Beta Identification Byte for Table 1:

1. Beta Byte equals X'04' if sequence of table is ascending
2. Beta Byte equals X'05' if sequence of table is descending
3. Beta Byte equals X'06' if sequence is neither ascending or descending

L_1 - Binary Length of Table 1 Entry (always equal to one less than length as specified in columns 40-42 of the File Extension specification)

D_1 - Number of Decimal Positions for Table 1 Entry (zero through nine if numeric or blank if alphameric)

γ_1 - Gamma Byte (always equal to X'08' since Table 1 is always field type "Factor 2, Result Field of a LOKUP")

P_1 - Packed or unpacked Table 1 Entry (P or blank)

S_1 - Sequence of Table 1 Entry (A, D, or blank):

S_1 equal to T indicates Table 2 Entry present

S_1 equal to blank indicates Table 2 Entry not present

N_2 - Name of Table 2

β_2 - Beta Identification Byte for Table 2:

1. Beta Byte equals X'04' if sequence of table is ascending
2. Beta Byte equals X'05' if sequence of table is descending
3. Beta Byte equals X'06' if sequence is neither ascending nor descending

L_2 - Binary Length of Table 2 Entry (always equal to one less than length as specified in columns 52-54 of the File Extension specification)

D_2 - Number of Decimal Positions for Table 2 Entry (zero through nine if numeric or blank if alphameric)

γ_2 - Gamma Byte (always equal to X'08' since Table 2 is always field type "Factor 2, Result Field of a LOKUP")

P_2 - Packed or unpacked Table 2 Entry (P or blank)

S_2 - Sequence of Table 2 Entry (A, D, or blank)

Minimum Compression Length - 25 Bytes

Maximum Compression Length - 37 Bytes

INPUT COMPRESSION

Record Type

IαBBFFRRSPPPTCPPPTCPPPTC

I - Type of specification
α - Indicates presence of fields via bit values

Bit	Value
0	0 - no Stacker Select field 1 - Stacker Select field
1-2	00 - OR record type 01 - AND record type 10 - alpha record type 11 - numeric record type
3	0 - numeric mandatory record type 1 - numeric optional record type

Bit	Value
4	0 - numeric 1 or more record type 1 - numeric 1 only record type
5	0 - no Filename field 1 - Filename field
6-7	00 - no record codes 01 - 1 record code 10 - 2 record codes 11 - 3 record codes

B - Input record sequence (Alpha or Numeric)

F - File Name Generated Sequence Number (from File Name Table)

R - Resulting Indicator

S - Stacker Select Number

P - Position of character (in packed format)

T - Type of character test

0 1 - negative test
0 - positive test

1 1 - character test
0 - not character test

2 1 - zone test
0 - not zone test

3 1 - digit test
0 - not digit test

C - Character used for code test

Minimum Compression Length - 7 bytes

Maximum Compression Length - 24 bytes

INPUT COMPRESSION (CONTINUED)

Field Type

DαFFFLDγANNNNNNLMRRPIIZZSSSS

D - Type of specification

α - Indicates presence of Fields via Bit Values

Bit	Value
0	0 - no Control Level 1 - Control Level
1	0 - no Matching Field 1 - Matching Field
2	0 - no Field-Record Relation 1 - Field-Record Relation
3	0 - no Plus used 1 - Plus used

Bit	Value
4	0 - no Minus used 1 - Minus used
5	0 - no Blank or Zero used 1 - Blank or Zero used
6	0 - no Sterling Field 1 - Sterling Field
7	0 - no Chaining Field 1 - Chaining Field

F - From position of field (in packed format)

L - Field Length: binary number calculated as follows:

Field is Shilling:	IBM Format	L = (TO-FROM)
Field is Shilling:	BSI Format	L = (TO-FROM + 1)
Field is packed numeric:		L = 2 (TO-FROM)
Otherwise:		L = (TO-FROM)

NOTE: L has a value one less than the number of Characters or digits in the field. The number of bytes occupied by the field can be calculated as:

$$\text{Bytes} = \left[\frac{L+1}{2}\right] + 1 \qquad \text{Numeric}$$

$$= L + 1 \qquad \text{Alphabetic}$$

D - Decimal Position (Unpacked Number)

γ - Field Type Byte: Constant X'80'

A - Decimal Positions and Packed Indication

N - Field Name

L - Control Level Number

M - Matching or Chaining Field Number

R - Record-Field Relation

P - Plus Indicator

I - Minus Indicator

Z - Zero or Blank Indicator

S - Sterling Field (unpacked Number)

Minimum Compression Length - 15 Bytes

Maximum Compression Length - 29 Bytes

CALCULATION COMPRESSION

CαBBIIIIIIIIIFFFFFFFFFFFFOOTTTTTTTTTTTTRRRRRRLDγHSSMMZZ

C - Type of specification

α - Indicates presence of fields via bit values

Bit	Value
0	0 - no Control Level 1 - Control Level
1	0 - no indicators 1 - indicators (if 1 indicator is used all are placed in the compression)
2	0 - no Factor 1 1 - Factor 1
3	0 - Factor 1 is a Field Name (in this case reserve 6 bytes in the compression) 1 - Factor 1 is a Literal (then reserve 12 bytes in the compression)

Bit	Value
4	0 - Factor 2 is a Field Name (in this case reserve 6 bytes in the compression) 1 - Factor 2 is a Literal (reserve 12 bytes in the compression)
5	0 - no Plus indicator 1 - Plus indicator
6	0 - no Minus indicator 1 - Minus indicator
7	0 - no Zero Indicator 1 - Zero indicator

B - Control Level

I - Indicators

F - Factor 1

O - Operation Code

T - Factor 2

R - Result field

L - Length of field in binary

D - Decimal Positions (unpacked number)

γ - Byte value indicates field type:

Byte Value	
10000000	All other field names (normal)
01000000	Result Field of a KEYCV, IDCV following an EXTCV
00100000	Factor 1 of EXTCV or RPGCV
00010000	Factor 2 of EXTCV or EXIT
00001000	Factor 2, Result Field of a LOKUP
00000100	RLABL
00000010	TAG, GOTO
00000001	ULABL

H - Half Adjust

S - Plus-High indicator

M - Minus-Low indicator

Z - Zero-Equal indicator

Minimum compression length - 20 bytes
Maximum compression length - 50 bytes

OUTPUT-FORMAT COMPRESSION

Record Type

OαΔ SBADDKKFFFFFFFFIIIIIIIII

O - Type of Specification

α - Indicates presence of fields

Bit	Value
1	0 - not heading line 1 - heading line
2	0 - not detail line 1 - detail line
3	0 - not total line 1 - total line
4	0 - line not conditioned by overflow 1 - line conditioned by overflow

Bit	Value
5	0 - File Name present 1 - no File Name and an AND line type
6	0 - File Name present 1 - no File Name and an OR line type
7-8	00 - no Resulting Indicators 01 - one Resulting Indicator 10 - two Resulting Indicators 11 - three Resulting Indicators

Δ - Length of compressed specification (binary)

S - Stacker Select Number

B - Space Before

A - Space After

D - Skip Before

K - Skip After

F - File Name (omit for AND/OR type)

I - Output Resulting Indicators (3 bytes each)

Minimum Compression Length - 6 bytes
Maximum Compression Length - 27 bytes

Field Type

MαΔ F.FFFFFEEENTLLL. . . LLATBBBBBIIIIIIIIICCSSSS

M - Type of specification

α - Indicates presence of fields:

Bit	Value
1-2	00 - no Resulting Indicators 01 - one Resulting Indicator 10 - two Resulting Indicators 11 - three Resulting Indicators
3	0 - no Field Name 1 - Field Name
4	0 - No Zero suppression 1 - Zero suppression

Bit	Value
5	0 - Blank After 1 - no Blank After
6	0 - no Literal 1 - Literal
7	0 - no Sterling Field 1 - Sterling Field
8	0 - no Packed Output Field 1 - Packed Output Field

Δ - Length of compressed specification (binary)

F - Field Name (omit for constants)

E - End Position (binary) (first bit of first byte is 1 for PAGE)

N - Literal Entry Length + 2 (binary)

OUTPUT- FORMAT COMPRESSION (CONTINUED)

T - Literal Type (X'AA' or X'40')

L - Literal Field*

A - Actual Literal Length - 1 (binary)*

T - Repeat of Literal Type*

B - Edit Word Information*

I - Resulting Indicators (3 bytes each)

C - Space for Blank After Resulting Indicator (2 blanks if present)

S - Sterling Field

Minimum Compression Length - 12 Bytes
Maximum Compression Length -62 Bytes

*Literal Field and Associated Entries:

1. Constant

   $L_1L_2L_3 \ldots L_n$ - Literal data (maximum of 24 bytes)

   A - Literal Length is $0 \leq A \leq 23$ ($A = n - 1$ for $n > 3$)

   T - X'40', indicates a Literal

2. Edit Word

   L_0 - * for asterisk protection; otherwise blank

   $L_1L_2L_3 \ldots L_n$ - Edit Word (maximum of 25 bytes)

   A - Edit Word Length (minus 1) including L_0 : $1 \leq A \leq 25$ ($A = n$ for $n > 2$)

   T - X'AA' indicates an edit word

   $B_1B_2B_3B_4B_5$ - Edit Word information:

   B_1 - Edit Word body length

   B_2 - flags for:

Bit	Value
1-4	Not used
5	Fixed dollar
6	Floating dollar
7	Sign status is -
8	Sign status is CR

   B_3 - displacement of sign status from L_0

   B_4 - number of digit positions in body

   B_5 - Displacement of zero suppression end

RESULTING INDICATORS, FIELD NAMES AND LITERALS IN COMPRESSIONS

The resulting indicators, field names, and literals contained in the compressed specifications will be replaced with the following information:

Resulting Indicators will be overlaid by:

BD DD

B = Base

DDD = Displacement

Field Names will be overlaid by:

BD DD ββ LL XD OV

B = Base

DDD = Displacement

ββ = Field Information Byte

LL = Field Length-1 (Binary)

X = X'E' for External Field, X'0' for Nonexternal Field

D = B for Alpha Field, X'0' - X'9' for Numeric Field

OV = Overflow Key, X'00' - X'04'

Literals will be overlaid beginning in the second byte by:

BD DD ββ LL OD OV

B = Base

DDD = Displacement

ββ = Field Information Byte

LL = Field Length - 1 (Binary)

OD = X'0B' for Alphameric Literal; X'00' - X'09" for Numeric Literal; and X'0A' for Edit Word

OV = Overflow Key, X'00' - X'04'

Literals in the calculation compression are always stored in a 12 byte field. When the literal is overlaid, the 12th byte of this field is set to X'00'. This is for Assign 2 to key on to determine when a literal has been overlaid with an address.

The Field Information Byte, ββ , has the following form:

Bit	Value
0-3	0000 - Assignment Key for Assign 2
4	0 - no blank-after 1 - blank-after
5	0 - BDDD references field directly 1 - BDDD references Table Linkage Field
6-7	00 - Table is in ascending order 01 - Table is in descending order 10 - Table is not specified as ascending or descending 11 - not assigned

Addressing, BDDD and OV, have the following meaning:

B	03-07	specifies base register used (binary)
DDD	000-FFF	specifies displacement used (binary)
OV	00-04	specifies contents of base register

00	no overflow	-	16,384
01	1st overflow	-	20,480
02	2nd overflow	-	24,576
03	3rd overflow	-	28,672
04	4th overflow	-	32,768

Plus the Program Relocation Factor

Linkage records are output by the Assign and Assemble Phases for later use by the Linkage Phase.

- R type records are output by the Assign and Assemble Phases and are used by the Linkage Phase to output RLD cards.
- V type records are output by Assemble Phases and are used to print out the memory map of object routine addresses.

R TYPE RECORD FORMAT

RAAANNLAAANNL.........FD

R - 1 byte, X'09' signifies RLD type record.

AAA - 3 bytes; absolute address of the address constant to be relocated.

NN - 1 byte; if the address constant to be relocated is an external type, NN will be the ID number from the ESD; otherwise NN will be the program number.

L - 1 byte; the length of the address constant.

X'FD' - 1 byte; signifies the last RLD entry in a record.

One or more RLD entries may be contained per record.

V TYPE RECORD FORMAT

VDDAAAADDAAAA..........FD

V = 1 byte, X'E5' signifies routine address record.

DD = 2 bytes, displacement (binary) of address in Linkage Vector.

AAAA = 4 bytes, routine address.

APPENDIX G. BLANK AFTER ENTRY FORMAT

The Blank After entries are written out by Assemble Phases 2 and 3 whenever a Field Name (in compression) is encountered that has indicators to be set on or off when the field is blanked. The format of the Blank After entries is as follows:

$$\overbrace{\text{BD}\quad\text{DD}\quad\beta\beta\quad\text{LL}\quad\text{XD}\quad\text{OV}}^{1}\qquad\overbrace{\text{BD}\quad\text{DD}}^{2}\qquad\overbrace{\text{BD}\quad\text{DD}}^{3}\qquad\overbrace{\text{BD}\quad\text{DD}}^{4}$$

where:

1 6 bytes, the field address and information bytes placed in compression by the Assign Phases.

2 2 bytes, the address of the indicator to be set on if the content of the field is plus (or high).

3 2 bytes, the address of the indicator to be set on if the content of the field is minus (or low).

4 2 bytes, the address of the indicator to be set on if the content of the field is zero or blank (or equal).

NOTE: The addresses of the indicators are expected in the above order and any unused indicators will be filled with null characters (X'00'). Addresses for the indicators are put into compression by the Assign Phases.

Displacements in Decimal From GR3	284	288	289	290
Condition That Turned H0 On	*	Resulting Byte Combinations Set (Hexadecimal)		
Initialized on or on due to programmer request	N/A	00	00	00
Invalid Chaining request	(A)	02	N/A	N/A
Undefined record type	(B)	10	N/A	N/A
Collating sequence error (matching records)	N/A	04	N/A	N/A
Record sequence error (predetermined sequence)	N/A	08	N/A	N/A
I/O Error - Combined Files	(D)	12	N/A	N/A
I/O Error - Direct Access File	(D)	16	(G)*	(G)*
I/O Error - Indexed-Sequential File (Random processing)	(D)	20	N/A	(E)*
I/O Error - Indexed-Sequential File (Sequential or between limits processing)	(C)	24	(F)*	(F)*
I/O Error - Sequential File	(C)	28	N/A	N/A

*

(A) = Chaining Identifier address
(B) = Address of IORB
(C) = DCB address
(D) = DECB address
(E) = Exceptional condition code from DECB (ISAM)
(F) = Two-byte exceptional condition code from DCB
(G) = Two-byte exceptional condition code from DECB (DAM)

GLOSSARY

Absolute address. Machine address. A pattern of characters that identifies a unique storage location (without modification).

Alpha (α) byte. Byte in the compression record that represents the presence or absence of fields (via bit values).

Alternate collating sequence. External subroutine (ALTSEQ) used to translate the sequence of a matching field to the collating sequence of the System/360.

Attribute. A characteristic; e.g., attributes of data include record length, record format, data set name, associated device type and volume identification, use, creation date.

BSI (British Standards Institution). Format for representing sterling fields; differs from the IBM format.

Beta (β) byte. Byte that represents field information (via bit values) in the field name and literal tables. Represents the sequence (via bit values) for table files in the compression.

Betabeta (ββ) byte. Two bytes that represent field information in the data that overlays the resulting indicators, field names and literals in the compression.

Chaining request block (CHB). Area within Assemble Phase 2 that contains identifying information and is generated and put out for each chaining field.

Compression. Technique used by this compiler to store the most data in the least amount of space. All unused information is deleted from the source specifications and the result is placed into a reserved area of core storage in form of records. Records are written on a work data set to be brought back in for processing in later phases of the program.

Concatenated data set. A collection of logically connected data sets.

Data set. The major unit of data storage and retrieval in the operating system, consisting of a collection of data in one of several prescribed arrangements and described by control information that the system has access to.

Decompression. Method used by the assemble phases to analyze compression created from specifications input and processed during enter phases.

Delta (Δ) byte. Byte that represents the length (binary) of the compressed output-format specification.

Driver (input/output, CIOEX, file, record). Routine maintained in core storage; contains table of data (constants, key addresses, switches, etc.) necessary to direct the program to certain routines.

Embedded blanks. Blank positions falling between characters of a name, e.g., DATE CRD.

ESD (external symbol dictionary). Control information associated with an object or load module which identifies the external symbols in the module.

Expansion. Method used by the compiler to convert compressed specification data into output text for the object program.

FET (file environment table). Table generated during Assemble Phase 2 and 2.5 for each input file in the object program. Contains identifying information obtained from file description specifications.

File group. Term used by Assemble Phase 2 and 2.5 to describe processing record type form of the input specification compressions.

FPB (file processing block). Area within Assemble Phase 2.5 generated and output for each primary, secondary or chained input file. Contains identifying information.

Gamma (γ) byte. Byte that represents field type definition. Appears in the compressions and in the field name table.

Intercept. Input/output interface for RAF, and chaining files; put out with the object program by Assemble Phase 1.

Interface. Control program that links the problem program with the operating system. Both the control program for the RPG compiler (CIOEX) and the control program for the generated object program are referred to as interface.

Internal sequence number. Number assigned by the compiler to each accepted specification as the compression is built. Used as a cross-reference throughout the phases of the program.

Inverted print. Numeric literals and edit words use the European conventions of punctuation (commas for decimal points and vice versa) in printed output.

IORB. Symbolic name of an area in the input/output phase (I/O Phases 1,2) that contains information in coded form about the input/output operation to be performed.

Linkage editor. A program that produces a load module by transforming object modules into a format that is acceptable to fetch, combining separately produced object modules and previously processed load modules into a single load module, resolving symbolic cross references among them, replacing, deleting, and adding control sections automatically on request, and providing overlay facilities for modules requesting them.

Linkage vector. Table output as part of the linkage program. Contains the addresses at execution time of all the routines used by the object program.

Load Module. The output of the linkage editor; a program in a format suitable for loading into main storage for execution. (Also referred to as object program.)

LOKUP (look up). Operation that procures specific information from one of the tables that is contained in core storage at object program execution time.

Multidefined. Describes a file defined more than once. Specifies the same filename in multiple specification entries. A multidefined file is created by a field name specified more than once but with different attributes.

Note point logic. Type of program logic designed to record (NOTE) the position on a work data set of the last block read or written. Subsequently the work data set is repostioned (POINT) to a specific block that was NOTEd.

Object module. The output of a single execution of an assembler or compiler which constitutes input to linkage editor. (Also referred to as object program.)

Parameter list or table. Table or list of variables that provides a specific function at object time; used by generalized routines output with the object program. The compiler builds these lists or tables from information obtained from specifications.

Precoded routines. Routines output with the object program. Stored within the compiler phases in object code form; output when requested by the specifications.

Preprocessed specifications. Calculation specifications for 18 possible operation codes partially processed, but not compressed, during Enter Phase 4. These preprocessed specifications are completely processed and compressed by Enter Phase 5.

RAF (record address file). Auxiliary file that contains the record keys (low and high) or a key to each record that is to be randomly processed. The object program depletes the RAF by processing all the data between the limits described by the keys.

Record group. Term used by Assemble Phase 2 to describe processing the field type form of the input specification compression.

Reenterable. Characteristic of a routine in main storage that allows the same copy of the routine to be executed concurrently on behalf of two or more routines or tasks.

RLD (relocation dictionary). Contains the addresses within the object program and the symbol in the ESD (for an external type) of address constants used in the generated object program; identified in RLD card images, and put out with the object program.

Service routines. Routines (CIOEX) that provide the logic for the input/output linkage to the operating system.

Symbol table. Printed listing of the entries in the field name, resulting indicator, and literal tables that were made during compilation.

System residence volume. The volume on which the nucleus of the operating system and the highest level index of the catalog are located.

Text. Instructions and data generated for the object module; written in card image format to be punched and/or input to a link edit job step.

TLF (table linkage field). Control field created for each table in the object program. Subfields (length, number of entries, address, etc.) are used by table operations at object time.

Transfer vector. Table used by Assemble Phase 2 to direct the program to the subroutines used by the phase. List of branch instructions.

UBA (undefined branch address). Branch instruction of the format B 0(0), which is put out in the object text stream. The address is inserted in a later step in the same phase.

Undefined. Table entry referenced but not defined.

Unreferenced. Table entry defined but not referenced.

Volatile. Characteristic of registers whose contents will not be preserved by CIOEX routines.

INDEX

READER'S COMMENT FORM

IBM System/360 Operating System
Report Program Generator

Form Y26-3704-0

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. Comments and suggestions become the property of IBM.

	Yes	No
• Does this publication meet your needs?	□	□
• Did you find the material:		
Easy to read and understand?	□	□
Organized for convenient use?	□	□
Complete?	□	□
Well illustrated?	□	□
Written for your technical level?	□	□

- What is your occupation? ________________________________
- How do you use this publication?

  As an introduction to the subject? □

  For advanced knowledge of the subject? □

  For information about operating procedures? □

  As an instructor in a class? □

  As a student in a class? □

  As a reference manual? □

  Other ____
- Please give specific page and line references with your comments when appropriate.

COMMENTS

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

fold fold

FIRST CLASS
PERMIT NO. 387
ROCHESTER, MINN.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Systems Development Division
Development Laboratory
Rochester, Minnesota 55901

Attention: Programming Publications, Dept. 425

fold fold

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

Y26-3704-0

IBM S/360 Printed in U.S.A. Y26-3704-0

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]