IBM Technical Newsletter

File Number S360-25

Re: Form No. Y28-6601-2

This Newsletter No. Y28-6827

Date November 15, 1968

Previous Newsletter Nos. Y28-6819 (applies to -2)
Y28-6383 (applies to -2)

IBM System/360 Operating System
FORTRAN IV (E)
Program Logic Manual

This Technical Newsletter, a part of Release 17 of the IBM System/360 Operating System, provides replacement pages for IBM System/360 Operating System: FORTRAN IV (E) Program Logic Manual, Form Y28-6601-2. These replacement pages remain in effect for subsequent releases unless specifically altered. Pages to be replaced and/or added are listed below.

Pages

Cover, preface
37-38.1
91-94
159-162 (159.1 deleted, 161.1 added)
165-166

Changes to the text, and small changes to illustrations, are indicated by a vertical line to the left of the change; changed or added illustrations are denoted by the symbol • to the left of the caption.

Summary of Amendments

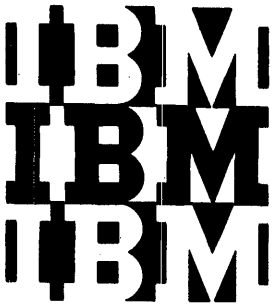
The discussion of address assignment by the compiler is clarified. Information concerning the format and organization of the communication area has been changed. Improvements in the processing of BACKSPACE statements by the FORTRAN object-time library have been added.

File this cover letter at the back of the publication to provide a record of changes.

IBM Corporation, Programming Publications, 1271 Avenue of the Americas, New York, N.Y. 10020

PRINTED IN U.S.A.

Restricted Distribution



Y28-6601-2

Program Logic

IBM System/360 Operating System

FORTRAN IV (E)

Program Logic Manual

Program Number 360S-FO-092

This publication describes the internal design of the IBM System/360 Operating System FORTRAN IV (E) compiler program. Program Logic Manuals are intended for use by IBM customer engineers involved in program maintenance, and by system programmers involved in altering the program design. Program logic information is not necessary for program operation and use; therefore, distribution of this manual is limited to persons with program maintenance or modification responsibilities.

Restricted Distribution

Form Y28-6601-2
Page Revised 11/15/68 by TNL Y28-6827

PREFACE

This manual is organized into three sections. Section 1 is an introduction and describes the overall structure of the compiler and its relationship to the operating system. Section 2 discusses the functions and logic of each phase of the compiler. Section 3 includes a series of flowcharts that show the relationship among the routines of each phase. Also provided in this section are phase routine directories.

Appendixes at the end of this publication provide information pertaining to: (1) source statement scan, (2) intermediate text formats, (3) table formats, (4) main storage allocation, etc.

Prerequisites to the use of this publication are:

IBM System/360 Operating System: Principles of Operation, Form A22-6821

IBM System/360 Operating System: Basic FORTRAN IV Language, Form C28-6629

IBM System/360 Operating System: Introduction to Control Program Logic, Program Logic Manual, Form Y28-6605

IBM System/360 Operating System: Basic FORTRAN IV (E) Programmer's Guide, Form C28-6603 (Sections "Job Processing" and "Cataloged Procedures")

Although not prerequisite, the following documents are related to this publication:

IBM System/360 Operating System: FORTRAN IV (E) Library Subprograms, Form C28-6596

IBM System/360 Operating System: Sequential Access Methods, Program Logic Manual, Form Y28-6604

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

IBM System/360 Operating System: Supervisor and Data Management Macro-Instructions, Form C28-6647

IBM System/360 Operating System: Linkage Editor, Program Logic Manual, Form Y28-6610

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

IBM System/360 Operating System: System Generation, Form C28-6554

This compiler is similar in design to the IBM System/360 Basic Programming Support FORTRAN IV Compiler.

Third Edition (September 1966)

This is a major revision of, and makes obsolete, the previous edition, Form Y28-6601-1. Changes to the text, and small changes to illustrations, are indicated by a vertical line to the left of the change; changed or added illustrations are denoted by the symbol • to the left of the caption.

The specifications contained in this publication, as amended by TNL Y28-6827, dated November 15, 1968, correspond to Release 17 of the IBM System/360 Operating System.

Changes are periodically made to the specifications herein; any such changes will be reported in subsequent revisions or Technical Newsletters.

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

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

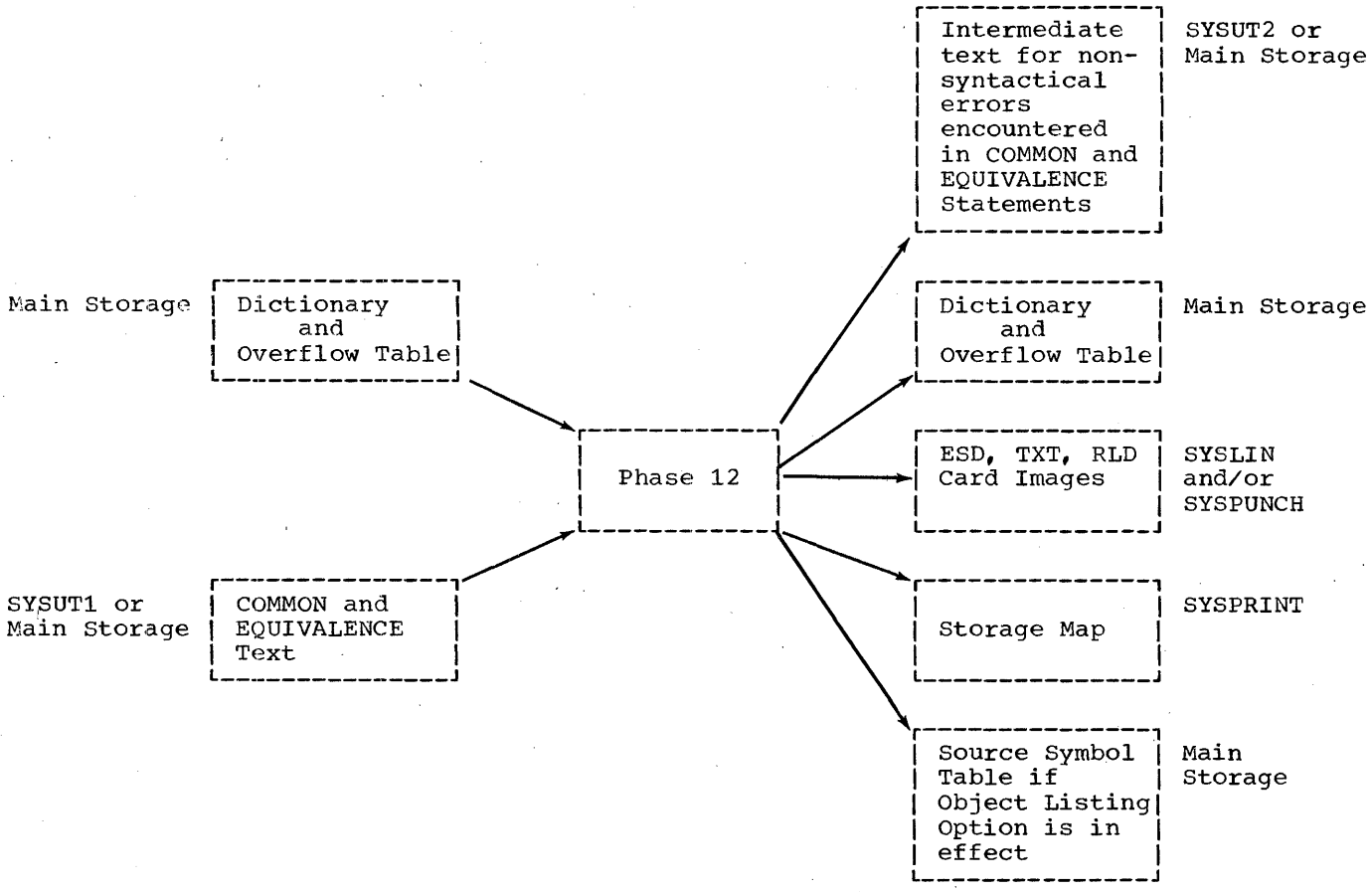


Figure 9. Phase 12 Data Flow

ADDRESS ASSIGNMENT

An effective address in IBM System/360 Operating System (a base-displacement address) is generated by adding the displacement value in an instruction to the value in an assigned base register. (A complete description of address generation can be found in the publication IBM System/360 Principles of Operation, Form A22-6821.) When addresses are assigned by Phase 12, the compiler uses a modified base-displacement format, where the high-order six bits represent the assigned base register and the low-order ten bits represent the displacement. The 2-byte address is stored in this internal format until Phase 25 converts it to IBM System/360 base-displacement form (i.e., the high-order four bits represent the base register and the low-order twelve bits represent the displacement.) All symbols in the object module generated by the compiler are referenced by this converted 2-byte address.

The base-displacement address is assigned through the use of a location counter, which is initialized and then incremented by the number of words needed in main storage to contain the variable, array, constant, address constant, or equated variable assigned an address. If more than 4096 bytes are needed, a new base register is assigned.

There are only two instances in which the location counter may be incremented when no address is assigned:

- The first occurs after the variables in COMMON are assigned addresses. A new base register is assigned to the location counter so that variables not in COMMON have different base registers than variables in COMMON.
- The second may occur before the assignment of addresses to double-precision constants that are not in COMMON. The location counter is adjusted to a doubleword boundary in order to accommodate double-precision constants.

When a variable is assigned an address, that address is placed in the chain field of the dictionary or overflow table entry for the variable.

FORMAT statements are assigned addresses during the execution of Phase 14. All phases after Phase 12 assign addresses whenever a constant or work area is defined.

EQUIVALENCE STATEMENT PROCESSING

The EQUIVALENCE text is processed by Phase 12 so that equated variables are assigned to the same address.

The following terms are used in the description of EQUIVALENCE processing:

- EQUIVALENCE group -- the variable and/or array names between a left and right parenthesis in an EQUIVALENCE statement.
- EQUIVALENCE class -- two or more EQUIVALENCE groups that have the following characteristics. If any EQUIVALENCE groups contain the same element, these groups form an EQUIVALENCE class. Further, if any other group contains an element in this class, the other group is part of this class, etc.
- Root -- the member of an EQUIVALENCE group or class from which all other variables in that group or class are referenced by means of a positive displacement.
- Displacement -- the distance, in bytes, between a variable and its root.

The root of an EQUIVALENCE group is assigned an address, and all other variables in the group are assigned addresses relative to that root.

To determine the root and the displacement of the other elements in the group from the root, the first element in the EQUIVALENCE group is established initially as the root. The displacement for the other elements (in relation to the root) is calculated by subtracting the offset of the root from the offset of the variable whose displacement is being calculated. (The offset for subscripted variables is contained in the EQUIVALENCE text created by Phase 10D. The offset for nonsubscripted variables is zero.)

If the resulting displacement is negative, the root is changed. The new root is the variable whose displacement was being calculated. Whenever a new root is assigned to an EQUIVALENCE group, the previously calculated displacements must be recalculated.

The root and the displacements in each group are entered in an EQUIVALENCE table, which is used by the storage assignment routines of Phase 12 to assign addresses to equated variables. (Refer to Appendix I for the table format.)

Note: Phase 12 generates intermediate text for nonsyntactical errors encountered in COMMON and EQUIVALENCE statements during relative address assignment. (The internal statement numbers for the error messages that are generated from this intermediate text by Phase 30 is 0000.) The amount of intermediate text for such errors depends on whether the SPACE or the PRFRM option is in effect.

If the SPACE option is in effect, the amount of error text is limited by the size of the first internal text buffer for the SYSUT2 data set. Phase 12 does not write any of the error text onto the SYSUT2 data set; it places the text into the above buffer. (The contents of the buffer are written onto SYSUT2 by Phase 14.) If the buffer is filled before COMMON and processing is completed, Phase 12 continues such processing, but does not generate additional error text. If the buffer is not filled before COMMON and EQUIVALENCE processing is completed, Phase 12 places the displacement of the next available location within the buffer into the FTXTPTRB field in the communication area. Phase 14 starts placing its intermediate text output at the location indicated by this field.

If the PRFRM option is in effect, there is no limitation on the amount of intermediate text generated by Phase 12 for COMMON and EQUIVALENCE statement errors. Phase 12 starts placing the error text into the first text buffer in the first text buffer chain for the SYSUT2 data set. When that buffer is full, the next buffer in the chain is used, etc. When all of the COMMON and EQUIVALENCE text is processed, the displacement of the next available location within the current buffer is placed into the FTXTPTRB field in the communication area. Phase 14 starts placing its intermediate text output at the location indicated by this field.

BRANCH LIST TABLE PREPARATION

The branch list table is initialized by Phase 12 (and is completed by Phase 25). This table is used by the object module to control the branching process. (Refer to Appendix J for the table format.) Each statement number referenced in a control statement is assigned a position relative to the start of the branch table. This position is indicated to Phase 25 by a relative number, which replaces the chain field of the corresponding statement number entry in the overflow table.

FOR PRFRM COMPILATIONS

For PRFRM compilations, the compiler requires main storage for:

- Load modules (phases, interface, and performance)
- Resident tables (dictionary, overflow table, and SEGMAL)
- Internal text buffer chains
- BSAM I/O routines
- Block/deblock buffers if blocking is specified

The main storage required by any given phase of the compiler need be contiguous only for each control section within that phase. Figure 23 reflects the main storage allocation for the duration of a PRFRM compilation, when only a minimal amount of main storage (19K bytes, where K=1024) is available for compilation.

When the main storage allocated to the compiler (specified in the SIZE option) is greater than 19K bytes, the internal text buffers may be interspersed within the area occupied by the dictionary and the overflow table. In this case, there need be no relationship among the various areas required by the compiler.

Figure 23 is a schematic showing the main storage allocated; proportional sizes within the diagram do not necessarily indicate proportional amounts of main storage.

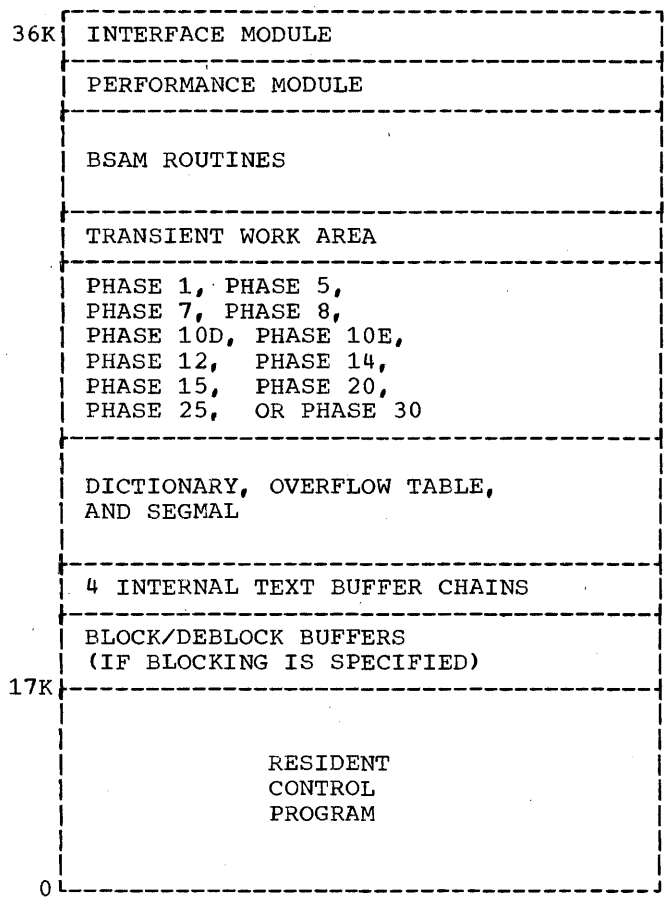


Figure 23. Main Storage Allocation for a PRFRM Compilation

APPENDIX B: COMMUNICATION AREA (FCOMM)

The communication area is a central gathering area used to communicate necessary information between the various phases of the compiler. The communication area, as a portion of the interface module, is resident throughout the compilation.

Various bits in the communication area are examined by the phases of the compiler. The status of these bits determines such things as:

- Options specified by the source programmer
- Specific action to be taken by a phase

Several entries in the communication area are equated to the addresses of other entries in the communication area used during earlier phases. Equating the entries keeps the size of the communication area to a minimum.

The communication area is assembled as a DSECT (dummy section) within each phase. This allows the phases to symbolically address the entries in the communication area without the communication area actually residing in each phase.

Table 22 indicates the format and organization of the communication area.

• Table 22. Communication Area (Part 1 of 3)

Entry	Size	Meaning
FCOMM	DS XL4	BIT0 SOURCE [1]
		BIT1 DECK [1]
		BIT2 MAP [1]
		BIT3 ADJUST [1]
		BIT4 PRFRM [1]
		BITS 5-6 00 NOLOAD [1]
		11 LOAD [1]
		BIT7 BCD [1]
		BIT8 NAME PARAMETER EXISTED
		BITS 9-10 00 MAIN PROGRAM
		10 SUBROUTINE SUBPROGRAM
		11 FUNCTION SUBPROGRAM
		BIT11 FUNCTION NAME DEFINED
		BIT12 OBJECT MODULE CALLS AN EXTERNAL S/P
		BIT13 SPARE
		BIT14 LAST COMPILE OF THIS JOB STEP-PH 10E/1
		BIT15 ERROR ON ANY COMPILE OF A BATCH RUN
		BIT16 WARNING MESSAGES
		BIT17 ERROR MESSAGES
		BIT18 MESSAGE IN CURRENT STATEMENT-PH 10D/10E
		INPUT BUFFER TO BE PRIMED-PH 12/14
		'DIOCS' ESD TO BE GENERATED-PH 14/20
		BIT19 WARNING IN ANY COMPILE OF A BATCH RUN
		BIT20 ABORT COMPILATION
		BIT21 ALL INTERNAL TEXT IN STORAGE
		BIT22 ONE INTERNAL TEXT RECORD-PH 10D/10E
		OBJ. MOD. USES A SPILL BASE REG-PH 12/25
		BRANCH LIST TEXT NOT ALL IN STORAGE-PH. 25/30
		BIT23 OBJECT LISTING
		BIT24 OTHER THAN FIRST COMPILE
		BIT25 COMPILATION RESTARTED
		BIT26 INVALID OPTION(S) IN 'PARM' FIELD
		BIT27 'NAME' OPTION TOO LONG-TRUNCATED
		BIT28 SYNAD EXIT OUTSTANDING
		BITS 29-31 SPARE

[1]Default values for these compiler options may be specified by the user during the system generation process via the FORTRAN macro-instruction. The default values specified at system generation time are assumed if the corresponding parameters in the PARM field of the user's EXEC statement are not included.

• Table 22. Communication Area (Part 2 of 3)

Entry	Size	Meaning	
FSIZE	DS F	BYTES OF STORAGE REQUESTED FOR COMPILER[1]	
FDATE	DS CL5	YEAR (2 DIGITS), DAY (3 DIGITS)	
FLINELNG	DS X	OBJECT PROGRAM PRINT LINE LENGTH[1]	
FINDEX	DS H	DISPLACEMENT FROM FCOMM TO FDECBIN	
FMAXLINE	DS H	MAXIMUM NUMBER OF LINES ON LISTING PAGE[1]	
FCURLINE	DS H	CURRENT LINE ON LISTING PAGE[1]	
FIEJF	DS CL4	FORTRAN E INTERNAL COMPONENT CODE - IEJF	
FPHASE	DS CL4	ENTRY POINT OF PHASE IN CONTROL	
FDMRRDCD	DS X	HI-ORDER BYTE OF REREAD ITEM IN CLOSE LIST	
FDMLSTCD	DS X	HI-ORDER BYTE OF LAST ITEM IN CLOSE LIST	
FPRTCTRL	DS 2H	BRANCH TO PRINT CONTROL ROUTINE	
THE CONTENTS OF THE NEXT 4 FIELDS DEPEND ON WHETHER A SPACE OR A PRFRM COMPILATION IS BEING PERFORMED.		FOR SPACE COMPILATIONS	FOR PRFRM COMPILATIONS
FIORTN	DS 2H	B SIORTN	MVI FPRFRMDL,X'4'
FNEXT	DS 2H	B SNEXT	L 13,FPRFRMDL
	DS H	(NOT USED)	BR 13
FPRFRMDL	DS A	ZERO	ADDR. OF IEJFAPA0
FAGA0END	DS A	ADDRESS OF (END OF INTERFACE MODULE + ONE)	
FSAVADDR	DS A	ADDRESS OF CONTROL PROGRAM SAVE AREA	
FTXBFSZA	DS H	SIZE OF 'SYSUT1' INT. TEXT BUFFER	
FTXBFSZB	DS H	SIZE OF 'SYSUT2' INT. TEXT BUFFER	
FTXTPTRA	DS H	DISP. OF NEXT SYSUT1 TEXT RCD.-PH. 10D/10E,12/14	
FTXTPTRB	DS H	DISP. OF NEXT SYSUT2 TEXT RCD.-PH. 12/14	
FTXTBFA1	DS A	ADDRESS OF INTERNAL TEXT BUFFER 1 - SYSUT1	
FTXTBFA2	DS A	ADDRESS OF INTERNAL TEXT BUFFER 2 - SYSUT1	
FTXTBFB1	DS A	ADDRESS OF INTERNAL TEXT BUFFER 1 - SYSUT2	
FTXTBFB2	DS A	ADDRESS OF INTERNAL TEXT BUFFER 2 - SYSUT2	
FPRTBUF1	DS A	ADDRESS OF FIRST PRINT BUFFER	
FPRTBUF2	DS A	ADDRESS OF SECOND PRINT BUFFER	
FINITBFS	DS 4A	INITIAL TEXT BUFFER POINTERS	
FDICTNDX	DS A	ADDRESS OF DICTIONARY INDEX - PHASE 7/12	
FOVFLNDX	DS A	ADDRESS OF OVERFLOW INDEX	
FDICTBLK	DS A	DICT. BLOCK NOW BEING BUILT - PH. 10D/E	
FOVFLBLK	DS A	OVFL. BLOCK NOW BEING BUILT - PH. 10D/E	
FDICTNXT	DS A	DICT. ENTRY NEXT TO BE BUILT - PH. 10D/E	
FOVFLNXT	DS A	OVFL. ENTRY NEXT TO BE BUILT - PH. 10D/14	
FISNEX1	DS F	ISN OF FIRST EXECUTABLE-PHASE 10D/E	
FOBJPROG	DS CL6	NAME OF OBJECT PROGRAM	
FOBJREGS	DS X	BITS 0-2 SPARE	
		BIT3 EXTERNAL FUNCTION HAS BEEN CALLED	
		BITS 4-7 LOWEST INDEX REGISTER IN OBJ. PROG.	
FASFCNT	DS X	COUNT OF SF'S IN OBJECT PROGRAM	
FDOCOUNT	DS H	NUMBER OF DO STATEMENTS	
	DS H	SPARE	

[1]Default values for these compiler options may be specified by the user during the system generation process via the FORTRAN macro-instruction. The default values specified at system generation time are assumed if the corresponding parameters in the PARM field of the user's EXEC statement are not included.

• Table 22. Communication Area (Part 3 of 3)

Entry	Size	Meaning
FCOMSIZE	EQU FDICTBLK	SIZE OF OBJECT PROGRAM COMMON - PH. 12/30
FALSIZE	EQU FDICTBLK+2	SIZE OF OBJ. PROG. ARGUMENT LIST - PH. 15/20
FBLSIZE	EQU FOVFLBLK	SIZE OF OBJ. PROG. BRANCH LIST - PH. 12/30
FBLSTRT	EQU FOVFLBLK+2	ADDR. OF OBJ. PROG. BRANCH LIST - PH. 12/30
FASFDOBL	EQU FOVFLNXT+2	ADDRESS OF ASF/DO BRANCH LIST - PH. 20/30
FBVSTRT	EQU FDICTNXT	ADDR. OF OBJ. PROG. BASE VAL. LIST - PH. 12/30
FOBJSTRT	EQU FDICTNXT+2	STARTING ADDR. OF OBJECT PROGRAM - PH. 12/30
FLOCCTR	EQU FISNEX1	LOCATION COUNTER FOR OBJ. PROG. - PH. 12/30
FFNCADDR	EQU FDICTBLK+2	ADDRESS OF RESULT (FUNCTION S/P) - PH. 14/15
FIBCOM	EQU FOVFLNXT	ADDRESS OF IBCOM - PHASE 20/25
FOBJERR	EQU FDICTBLK+2	ADDR. OF OBJ. PROG. ERROR RTNE. - PH. 20/25
FDECKSEQ	EQU FDICTNDX	OBJECT PROGRAM DECK SEQUENCE NUMBER - PH. 12/30
FESDSEQ	EQU FDICTNDX+2	OBJECT PROGRAM ESD SEQUENCE NUMBER - PH. 12/20
FENDSTOR	EQU FDICTNDX+2	END-OF-DATA STORAGE ADDRESS - PH. 25/30
FALSTRT	DS F	DSRN ARGUMENT LIST ADDRESS
FDATEMP	DS F	ADDRESS OF DIRECT ACCESS I/O TEMPORARY AREA
FDEFILCT	DS F	'DEFINE FILE' DSRN COUNT - PH. 10D/20
FDIOCS	EQU FDEFILCT	ADDRESS OF DIOCS - PH. 20/25
FPATCH	DS 2H	BRANCH TO PATCH ROUTINE IN INTERFACE MODULE
FPTCHTBL	DS A	ADDRESS OF PATCH TABLE
FPTCHPTR	DS A	PATCH TABLE ENTRY NEXT TO BE POSTED
FSORSYM1	DS A	ADDRESS OF SORSYM TABLE
FSORSYM2	DS A	SORSYM TABLE ENTRY NEXT TO BE BUILT
FSYNADGR	DS 2A	DECB AND DCB POINTERS AT SYNAD EXIT

[1]Default values for these compiler options may be specified by the user during the system generation process via the FORTRAN macro-instruction. The default values specified at system generation time are assumed if the corresponding parameters in the PARM field of the user's EXEC statement are not included.

I/O Device Manipulation Routines

The I/O device manipulation routines of IHCFCOME implement the BACKSPACE, REWIND, and END FILE source statements. These routines receive control from within the load module via calling sequences that are generated by the compiler when these statements are encountered.

Note: Backspace, rewind, and end file requests are honored only for sequential data sets and are ignored for direct access data sets. However, these statements are device independent and can be used for sequential data sets on either sequential or direct access devices.

The implementation of BACKSPACE, REWIND, and END FILE statements is straightforward. The I/O device manipulation routines submit the appropriate control request to IHCFIOSH, the I/O interface module. After the request is executed, control is returned to the calling routine within the load module.

Write-to-Operator Routines

The write-to-operator routines of IHCFCOME implement the STOP and PAUSE source statements. These routines receive control from within the load module via calling sequences generated by the compiler upon recognition of the STOP and PAUSE statements.

STOP: A write-to-operator (WTO) macro-instruction is issued to display the message associated with the STOP statement on the console. Load module execution is then terminated by passing control to the program termination routine of IHCFCOME.

PAUSE: A write-to-operator-with-reply (WTOR) macro-instruction is issued to display the message associated with the PAUSE statement on the console and to enable the operator's reply to be transmitted. A WAIT macro-instruction is then issued to determine when the operator's reply has been transmitted. After the reply has been received, control is returned to the calling routine within the load module.

Utility Routines

The utility routines of IHCFCOME perform the following functions:

- Process object-time error messages
- Process arithmetic-type program interruptions
- Terminate load module execution

PROCESSING OF ERROR MESSAGES: The error message processing routine (IBFERR) receives control from various FORTRAN library subprograms when they detect object-time errors.

Error message processing consists of initializing the data set upon which the message is to be written and also of writing the message. Control is then passed to the termination routine of IHCFCOME.

PROCESSING OF ARITHMETIC INTERRUPTIONS: The arithmetic-interrupt routine (IBFINT) of IHCFCOME initially receives control from within the load module via a compiler-generated calling sequence. The call is placed at the start of the executable coding of the load module so that the interrupt routine can set up the program interrupt mask. Subsequent entries into the interrupt routine are made through arithmetic-type interruptions.

The interrupt routine sets up the program interrupt mask by means of a SPIE macro-instruction. This instruction specifies the type of arithmetic interruptions that are to cause control to be passed to the interrupt routine, and the location within the routine to which control is to be passed if the specified interruptions occur. After the mask has been set, control is returned to the calling routine within the load module.

In processing an arithmetic interruption, the first step taken by the interrupt routine is to determine its type. If exponential overflow or underflow has occurred, the appropriate indicators, which are referenced by OVERFL (a library subprogram), are set. If any type of divide check caused the interruption, the indicator referenced by DVCHK (also a library subprogram) is set.

Regardless of the type of interruption that caused control to be given to the interrupt routine, the old program PSW is written out for diagnostic purposes.

After the interruption has been processed, control is returned to the interrupted routine at the point of interruption.

PROGRAM TERMINATION: The load module termination routine (IBEXIT) of IHCFCOME receives control from various library subprograms (e.g., DUMP and EXIT) and from other IHCFCOME routines (e.g., the routine that processes the STOP statement).

This routine terminates execution of the load module by the following means:

- Calling the appropriate I/O interface(s) to check (via the CHECK macro-instruction) outstanding write requests
- Issuing a SPIE macro-instruction with no parameters to indicate that the FORTRAN object module no longer desires to give special treatment to program interruptions and does not want maskable interruptions to occur.
- Returning to the operating system supervisor.

IHCFIOSH

IHCFIOSH, the object-time FORTRAN sequential access input/output data management interface, receives I/O requests from IHCFCOME and submits them to the appropriate BSAM (basic sequential access method) routines and/or open and close routines for execution.

Chart E3 illustrates the overall logic and the relationship among the routines of IHCFIOSH. Table 37, the IHCFIOSH routine directory, lists the routines used in IHCFIOSH and their functions.

BLOCKS AND TABLE USED

IHCFIOSH uses the following blocks and table during its processing of sequential access input/output requests: (1) unit blocks, and (2) unit assignment table. The unit blocks are used to indicate I/O activity for each unit number (i.e., data set reference number) and to indicate the type of operation requested. In addition, the unit blocks contain skeletons of the data event control blocks (DECB) and the data control blocks (DCB) that are required for I/O operations. The unit assignment table is used as an index to the unit blocks.

Unit Blocks

The first reference to each unit number (data set reference number) by an input/output operation within the FORTRAN load module causes IHCFIOSH to construct a unit block for each unit number. The main storage for the unit blocks is obtained by IHCFIOSH via the GETMAIN macro-instruction. The addresses of the unit blocks are placed in the unit assignment table as the unit blocks are constructed. All subsequent references to the unit numbers are then made through the unit assignment table.

ABYTE	BBYTE	CBYTE	DBYTE	4 bytes
Address of Buffer 1				4 bytes
Address of Buffer 2				4 bytes
Current buffer pointer*				4 bytes
Record offset (RECPTR)*				4 bytes
Address of last DECB				4 bytes
Mask for alternating buffers				4 bytes
DECB1 skeleton section				20 bytes
Not used			LIVECNT1	4 bytes
DECB2 skeleton section				20 bytes
Work space		Not used	LIVECNT2	4 bytes
DCB skeleton section				88 bytes

Housekeeping Section

*Used only for variable length and/or blocked records.

• Figure 90. Format of a Unit Block for a Sequential Access Data Set

Figure 90 illustrates the format of a unit block for a unit that is defined as a sequential access data set.

Each unit block is divided into four sections: a housekeeping section, two DECB skeleton sections, and a DCB skeleton section.

HOUSEKEEPING SECTION: The housekeeping section is maintained by IHCFIOSH. The information contained in it is used to indicate data set type, to keep track of I/O buffer locations, and to keep track of addresses internal to the I/O buffers to enable the processing of blocked records. The fields of this section are:

- ABYTE. This field, containing the data set type passed to IHCFIOSH by IHCFCOME, can be set to one of the following:

  F0 - Input data set requiring a format
  FF - Output data set requiring a format
  00 - Input data set not requiring a format
  0F - Output data set not requiring a format

- BBYTE. This field contains bits that are set and examined by IHCFIOSH during its processing. The bits and their meanings, when on, follow:

  0 - Exit to IHCFCOME on I/O error
  1 - I/O error occurred
  2 - Current buffer indicator
  3 - Not used
  4 - End-of-current buffer indicator
  5 - Blocked data set indicator
  6 - Variable record format switch
  7 - Not used

- CBYTE. This field also contains bits that are set and examined by IHCFIOSH. The bits and their meanings, when on, follow:

  0 - Data control block opened
  1 - Data control block not TCLOSEd
  2 - Data control block not previously opened
  3 - Buffer pool attached
  4 - Data set not previously rewound
  5 - Not used
  6 - Concatenation occurring -- reissue READ
  7 - Data set is DUMMY

- DBYTE. This field contains bits that are set and examined by IHCFIOSH during the processing of an input/output operation involving a backspace request. The bits and their meanings, when on, follow:

  0 - A physical BACKSPACE has occurred
  1 - Previous operation was BACKSPACE
  2 - Not used
  3 - End-of-file routine should retain buffers
  4 - Not used
  5 - Not used
  6 - END FILE followed by BACKSPACE
  7 - Not used

- Address of Buffer 1 and Address of Buffer 2. These fields contain pointers to the two I/O buffers obtained during the opening of the data control block for this data set.

- Current Buffer Pointer. This field contains a pointer to the I/O buffer currently being used.

- Record Offset (RECPTR). This field contains a pointer to the current logical record within the current buffer.

- Address of Last DECB. This field contains a pointer to the last DECB used.

- Mask for Alternating Buffers. This field contains the bits which enable an Exclusive Or operation to alternate the current buffer pointer.

DECB SKELETON SECTIONS (DECB1 AND DECB2): The DECB (data event control block) skeleton sections are blocks of main storage within the unit block. They have the same form as the DECB constructed by the control program for an L form of an S-type READ or WRITE macro-instruction (refer to the publication IBM System/360 Operating System: Supervisor and Data Management Macro-Instructions, Form C28-6647). The various fields of the DECB skeleton are filled in by IHCFIOSH; the completed block is referred to when IHCFIOSH issues a read/write request to BSAM. The read/write field is filled in at open time. For each I/O operation, IHCFIOSH supplies IHCFCOME with: (1) an indication of the type of operation (read or write), and (2) the length of and a pointer to the I/O buffer to be used for the operation.

- LIVECNT1 AND LIVECNT2. These fields indicate whether any input/output operation performed for the data set is unchecked. (A value of 1 indicates that a previous read or write has not been checked; a value of 0 indicates that all previous read and write operations for the data set have been checked.)

- Work Space. This field is used to align the logical record length of a variable record segment on a fullword boundary.

DCB SKELETON SECTION: The DCB (data control block) skeleton section is a block of main storage within the unit block. It is of the same form as the DCB constructed by the control program for a DCB macro-instruction under BSAM (refer to the Supervisor and Data Management Macro-Instructions publication). The various fields of the DCB skeleton are filled in by the control program when the DCB for the data set is opened (refer to the publication IBM System/360 Operating System: Concepts and Facilities, Form C28-6535). Standard default values may also be inserted in the DCB skeleton by IHCFIOSH. See "Unit Assignment Table" for a discussion of the insertion of default values into the DCB skeleton.

Unit Assignment Table

The unit assignment table (IHCUATBL) resides on the FORTRAN system library (SYS1.FORTLIB). Its size depends on the maximum number of units that can be referred to during execution of any FORTRAN load module. This number (≤ 99) is specified by the user during the system generation process via the FORTLIB macro-instruction.

The unit assignment table is designed to be used by both IHCFIOSH and IHCDIOSE. It is included once, by the linkage editor, in the FORTRAN load module as a result of an external reference to it within IHCFIOSH and/or IHCDIOSE.

The unit assignment table contains a 16-byte entry for each of the unit numbers that can be referred to by the user. These entries differ in format depending on whether the unit has been defined as a sequential access or a direct access data set.

Figure 91 illustrates the format of the unit assignment table.

Unit number (DSRN) being used for current operation	[1] n x 16	4 bytes
Unit number (DSRN) of error output device	not used	4 bytes
UBLOCK01 field		4 bytes
DSRN01 default values		8 bytes
LIST01 field		4 bytes
. . .	. . .	
UBLOCKn field[2]		4 bytes
DSRNn default values[3]		8 bytes
LISTn field[4]		4 bytes

[1]n is the maximum number of units that can be referred to by the FORTRAN load module. The size of the unit table is equal to (8 + n x 16) bytes.

[2]The UBLOCKn field contains either a pointer to the unit block constructed for unit number n if the unit is being used at object-time, or a value of 1 if the unit is not being used.

[3]The default values for the various unit numbers are specified by the user and are assembled into the unit assignment table entries during the system generation process. The default values are used only by IHCFIOSH; they are ignored by IHCDIOSE.

[4]If the unit is defined as a direct access data set, the LISTn field contains a pointer to the parameter list that defines the direct access data set. Otherwise, this field contains a value of 1.

Figure 91. Unit Assignment Table Format

Because IHCFIOSH deals only with sequential access data sets, the remainder of the discussion on the unit assignment table is devoted to unit assignment table entries for sequential access data sets. If IHCFIOSH encounters a reference to a direct access data set, it is considered an error, and control is passed to the load module termination routine of IHCFCOME.

The pointers to the unit blocks created for sequential data sets are inserted into the unit assignment table entries by IHCFIOSH when the unit blocks are constructed.

Note: Default values are standard values that IHCFIOSH inserts into the appropriate fields (e.g., BUFNO) of the DCB skeleton section of the unit blocks if the user either:

- Causes the load module to be executed via a cataloged procedure, or
- Fails, in stating his own procedure for execution, to include in the DCB parameter of his DD statements those subparameters (e.g., BUFNO) he is permitted to include (refer to the publication IBM System/360 Operating System: Basic FORTRAN IV (E) Programmer's Guide).

Control is returned to IHCFIOSH during data control block opening so that it can determine whether the user has included the subparameters in the DCB parameter of his DD statements. IHCFIOSH examines the DCB skeleton fields corresponding to user-permitted subparameters, and upon encountering a null field (indicating that the user has not specified the subparameter), inserts the standard value (i.e., the default value) for the subparameter into the DCB skeleton. (If the user has included these subparameters in his DD statement, the control program routine performing data control block opening inserts the subparameter values, before giving control to IHCFIOSH, into the DCB skeleton fields reserved for those values.)

BUFFERING

All input/output operations are double buffered. (The double buffering scheme can be overriden by the user if he specifies in a DD statement: BUFNO=1.) This implies that during data control block opening, two buffers will be obtained. The addresses of these buffers are given alternately to IHCFCOME as pointers to:

- Buffers to be filled (in the case of output)
- Information that has been read in and is to be processed (in the case of input)

COMMUNICATION WITH THE CONTROL PROGRAM

In requesting services of the control program, IHCFIOSH uses L and E forms of S-type macro-instructions (refer to the publication IBM System/360 Operating System: Control Program Services).

Note: The write section checks to see if the data set being written on is a 1403 printer. If it is, the carriage control character is changed to machine code, and three buffers, instead of the normal two, are used when writing on the printer.

ERROR PROCESSING: If an end-of-data set or an I/O error is encountered during reading or writing, the control program returns control to the location within IHCFIOSH that was specified during data set initialization. In the case of an I/O error, IHCFIOSH sets a switch to indicate that the error has occurred. Control is then returned to the control program. The control program completes its processing and returns control to IHCFIOSH, which interrogates the switch, finds it to be set, and passes control to the I/O error routine of IHCFCOME.

In the case of an end-of-data set, IHCFIOSH simply passes control to the end-of-data set routine of IHCFCOME.

Chart E4 illustrates the execution-time I/O recovery procedure for any I/O errors detected by the I/O supervisor.

Device Manipulation

The device manipulation section of IHCFIOSH processes backspace, rewind, and write end-of-data set requests.

BACKSPACE: IHCFIOSH processes the backspace request by issuing the appropriate number of BSP (physical backspace) macro-instructions (0, 1, 2, or 3) and adjusting the RECPTR in the unit blocks to point to the preceding logical record. The number of BSP's issued depends on the number of buffers used, the previous input/output operation, and the position of RECPTR prior to the backspace.

For unformatted records, the processing of a backspace request also includes examining the SDW (segment descriptor word) of each record segment in order to locate the first segment of a spanned record (i.e., a logical record which causes more than one physical input/output operation to be performed). Control is then returned to IHCFCOMH.

REWIND: IHCFIOSH processes the rewind request by issuing a CLOSE macro-instruction, using the REREAD option. This option has the same effect as a rewind. Control is then returned to IHCFCOME.

WRITE END-OF-DATA SET: IHCFIOSH processes this request by issuing a CLOSE macro-instruction, type = T. It then frees the I/O buffers by issuing a FREEPOOL macro-instruction, and returns control to IHCFCOME.

Closing

The closing section of IHCFIOSH examines the entries in the unit assignment table to determine which data control blocks are open. In addition, this section ensures that all write operations for a data set are completed before the data control block for that data set is closed. This is done by issuing a CHECK macro-instruction for all double-buffered output data sets. Control is then returned to IHCFCOME.

Note: If a 1403 printer is being used, a WRITE from the last print buffer is issued to ensure that the last line of output is written.

IHCDIOSE

IHCDIOSE, the object-time FORTRAN direct access input/output data management interface, receives I/O requests from IHCFCOME and submits them to the appropriate BDAM (basic direct access method) routines and/or open and close routines for execution. (For the first I/O request involving a non-existent data set, the appropriate BSAM routines must be executed prior to linking to the BDAM routines. The BSAM routines format and create a new data set consisting of blank records.)

IHCDIOSE receives control from: (1) the initialization section of the FORTRAN load module if a DEFINE FILE statement is included in the source module, and (2) IHCFCOME whenever a READ, WRITE, or FIND direct access statement is encountered in the load module.

Charts E5 and E6 illustrate the overall logic and the relationship among the routines of IHCDIOSE. Table 38, the IHCDIOSE routine directory, lists the routines used in IHCDIOSE and their functions.

BLOCKS AND TABLE USED

IHCDIOSE uses the following blocks and table during its processing of direct access input/output requests: (1) unit blocks, and (2) unit assignment table. The unit blocks are used to indicate I/O activity for each unit number (i.e., data set reference number) and to indicate the type of operation requested. In addition, each unit block contains skeletons of the

data event control blocks (DECB) and the data control block (DCB) that are required for I/O operations. The unit assignment table is used as an index to the unit blocks.

Unit Blocks

The first reference to each unit number (i.e., data set reference number) by a direct access input/output operation within the FORTRAN load module causes IHCDIOSE to construct a unit block for each of the referenced unit numbers. The main storage for the unit blocks is obtained by IHCDIOSE via the GETMAIN macro-instruction. The addresses of the unit blocks are inserted into the corresponding unit assignment table entries as the unit blocks are constructed. Subsequent references to the unit numbers are then made through the unit assignment table.

Figure 93 illustrates the format of a unit block for a unit that has been defined as a direct access data set.

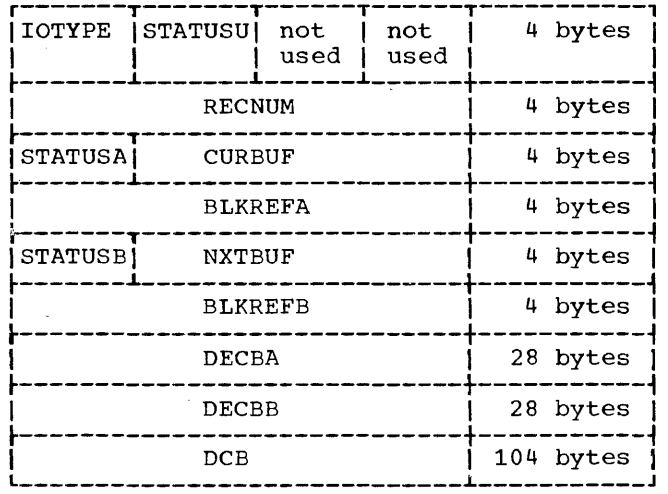


Figure 93. Format of a Unit Block for a Direct Access Data Set

The meanings of the various unit block fields are outlined below.

IOTYPE: This field, containing the data set type passed to IHCDIOSE by IHCFCOME, can be set to one of the following:

- F0 - Input data set requiring a format
- FF - Output data set requiring a format
- 00 - Input data set not requiring a format
- 0F - Output data set not requiring a format

STATUSU: This field specifies the status of the associated unit number. The bits and their meanings, when on, follow:

- 0 - Not used
- 1 - Error occurred
- 2 - Two buffers are being used
- 3 - DCB for data set open
- 4-5 - 10 - U-form specified in DEFINE FILE statement
  01 - E-form specified in DEFINE FILE statement
  11 - L-form specified in DEFINE FILE statement
- 6-7 - Not used

Note: IHCDIOSE references only bits 1, 2, and 3.

RECNUM: This field contains the number of records in the data set as specified in the parameter list for the data set in a DEFINE FILE statement. It is filled in by the file initialization section after the data control block for the data set is opened.

STATUSA: This field specifies the status of the buffer currently being used. The bits and their meanings, when on, follow:

- 0 - READ macro-instruction has been issued
- 1 - WRITE macro-instruction has been issued
- 2 - CHECK macro-instruction has been issued
- 3-7 - Not used

CURBUF: This field contains the address of the DECB skeleton currently being used. It is initialized to contain the address of the DECBA skeleton by the file initialization section of IHCDIOSE after the data control block for the data set is opened.

BLKREFA: This field contains an integer that indicates either the relative position within the data set of the record to be read, or the relative position within the data set at which the record is to be written. It is filled in by either the read or write section of IHCDIOSE prior to any reading or writing. In addition, the address of this field is inserted into the DECBA skeleton by the file initialization section of IHCDIOSE after the data control block for the data set is opened.

STATUSB: This field specifies the status of the next buffer to be used if two buffers are obtained for this data set during data control block opening. The bits and their meanings are the same as described for the STATUSA field. However, if only one buffer is obtained during data control block opening, this field is not used.