SPERRY UNIVAC
COMPUTER SYSTEMS

PUBLICATIONS UPDATE

Operating System/3 (OS/3)

Supervisor Programmer Reference

UP-8241 Rev. 2–D

This Library Memo announces the release and availability of Updating Package D to "SPERRY UNIVAC Operating System/3 (OS/3) Supervisor Programmer Reference", UP-8241 Rev. 2.

This update documents the following changes for release 8.0:

- Enhancement of the OC STXIT routine.
- Restrictions to the monitor routine.

This update also includes minor technical corrections to material applicable to the supervisor prior to release 8.0.

Copies of Updating Package D are now available for requisitioning. Either the updating package only or the complete manual with the updating package may be requisitioned by your local Sperry Univac representative. To receive only the updating package, order UP-8241 Rev. 2–D. To receive the complete manual, order UP-8241 REv. 2.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists BZ, CZ and MZ	Mailing Lists A00, A01, 18, 18U, 19, 19U, 20, 20U, 21, 21U, 75, 75U, 76, and 76U (Package D to UP-8241 Rev. 2, 31 pages plus Memo)	Library Memo for UP-8241 Rev. 2–D
		RELEASE DATE: September, 1982

SPERRY UNIVAC
COMPUTER SYSTEMS

PUBLICATIONS UPDATE

Operating System/3 (OS/3)

Supervisor

Programmer Reference

UP-8241 Rev. 2–B

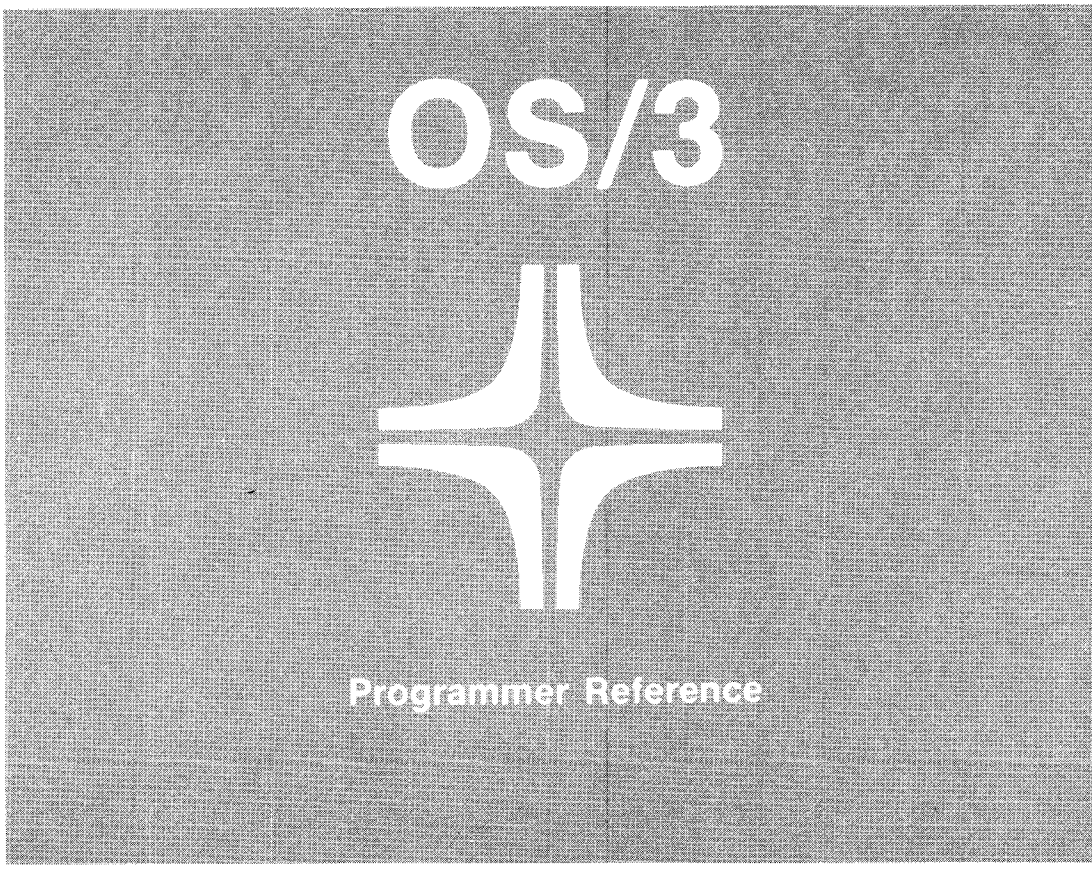
This Library Memo announces the release and availability of Updating Package B to "SPERRY UNIVAC Supervisor Programmer Reference", UP-8241 Rev. 2.

This update includes a clarification of the format of message buffers used in the WTL, WTLD, and OPR macroinstructions.

Copies of Updating Package B are now available for requisitioning. Either the updating package alone, or the complete manual with the updating package may be requisitioned by your local Sperry Univac representative. To receive the updating package alone, order UP-8241 Rev. 2–B. To receive the complete manual, order UP-8241 Rev. 2.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists BZ, CZ and MZ	Mailing Lists 18, 18U, 19, 19U, 20, 20U, 21, 21U, 75, 75U, 76 and 76U (Package B to UP-8241 Rev. 2, Covers and 14 pages plus Memo)	Library Memo RELEASE DATE: October, 1980

Supervisor



Environment: 90/25, 30, 30B, 40

SPERRY UNIVAC

This document contains the latest information available at the time of preparation. Therefore, it may contain descriptions of functions not implemented at manual distribution time. To ensure that you have the latest information regarding levels of implementation and functional availability, please consult the appropriate release documentation or contact your local Sperry Univac representative.

Sperry Univac reserves the right to modify or revise the content of this document. No contractual obligation by Sperry Univac regarding level, scope, or timing of functional implementation is either expressed or implied in this document. It is further understood that in consideration of the receipt or purchase of this document, the recipient or purchaser agrees not to reproduce or copy it by any means whatsoever, nor to permit such action by others, for any purpose without prior written permission from Sperry Univac.

Sperry Univac is a division of the Sperry Corporation.

This document was prepared by Systems Publications using the SPERRY UNIVAC UTS 400 Text Editor. It was printed and distributed by the Customer Information Distribution Center (CIDC), 555 Henderson Rd., King of Prussia, Pa., 19406.

PRINTED IN U.S.A.

SPERRY UNIVAC
Operating System/3 (OS/3)
Supervisor
Programmer Reference

This document contains the latest information available at the time of preparation. Therefore, it may contain descriptions of functions not implemented at manual distribution time. To ensure that you have the latest information regarding levels of implementation and functional availability, please contact your local Sperry Univac representative.

Sperry Univac reserves the right to modify or revise the content of this document. No contractual obligation by Sperry Univac regarding level, scope, or timing of functional implementation is either expressed or implied in this document. It is further understood that in consideration of the receipt or purchase of this document, the recipient or purchaser agrees not to reproduce or copy it by any means whatsoever, nor to permit such action by others, for any purpose without prior written permission from Sperry Univac.

Sperry Univac is a division of the Sperry Rand Corporation.

This document was prepared by Systems Publications using the SPERRY UNIVAC UTS 400 Text Editor. It was printed and distributed by the Customer Information Distribution Center (CIDC), 555 Henderson Rd., King of Prussia, Pa., 19406.

PRINTED IN U.S.A.

PAGE STATUS SUMMARY

ISSUE: Update D — UP-8241 Rev. 2
RELEASE LEVEL: 8.0 Forward

Part/Section	Page Number	Update Level
Cover/Disclaimer		B
PSS	1	D
Preface	1	Orig.
Contents	1 2 3 4 5, 6	Orig. A Orig. C Orig.
Section 1	Title Page 1, 2 3, 4 5 6	Orig. Orig. A C Orig.
Section 2	Title Page 1 thru 7 8 9 10 11, 12 13 14 15 16 17 18 19 thru 21 22 23 24 thru 26 27, 28 29 30 thru 33 34 35 36 36a 37 thru 42 43, 44 45 46, 47 48 49 thru 55 56 57 thru 59 60, 61 62 63 thru 76 77 78	Orig. Orig. D A D Orig. A D Orig. D A D Orig. D Orig. C Orig. A Orig. D Orig. A A Orig. A Orig. A B Orig. A Orig. A C Orig. A Orig.

Part/Section	Page Number	Update Level
Section 2 (cont)	79, 80 81 thru 83 84 85 thru 94 95, 96 97 98 99 100 100a thru 100c 101 thru 107 108 109 110, 111	C Orig. A Orig. D A C D C C Orig. B C Orig.
Appendix A	Title Page 1, 2 3 thru 8 9, 10 11 thru 15 16 17, 18	Orig. A Orig. D Orig. A Orig.
Appendix B	Title Page 1	Orig. B
Appendix C	Title Page 1	Orig. Orig.
Appendix D	Title Page 1 2 3 thru 16	Orig. Orig. A Orig.
Appendix E	Title Page 1 thru 6	Orig. Orig.
Appendix F	Title Page 1 2 3 thru 17	Orig. Orig. D Orig.
Glossary	1, 2 3 4, 5 6	D Orig. D Orig.
User Comment Sheet		

Part/Section	Page Number	Update Level

All the technical changes are denoted by an arrow (➔) in the margin. A downward pointing arrow (↓) next to a line indicates that technical changes begin at this line and continue until an upward pointing arrow (↑) is found. A horizontal arrow (➔) pointing to a line indicates a technical change in only that line. A horizontal arrow located between two consecutive lines indicates technical changes in both lines or deletions.

Preface

This programmer reference manual is one in a series designed to be used as a quick-reference document for programmers familiar with the SPERRY UNIVAC Operating System/3 (OS/3). This particular manual describes the macro instructions used in communicating with the supervisor and in achieving optimum system utilization.

The information presented here is limited to facts; no introductory information or examples of use are provided. This type of information is presented in two other supervisor manuals: an introduction to the supervisor, UP-8058 (current version), and a supervisor user guide, UP-8075 (current version).

The information contained in this manual is presented as follows:

- Section 1. General Information

  Describes the supervisor concepts, organization, and interfaces. This section also states the general conventions for writing macro instruction statements which request the services of the supervisor.

- Section 2. Supervisor Macro Instructions

  Gives the function, format, and parameter description of each supervisor macro instruction, in alphabetic sequence by macro instruction mnemonic.

- Appendixes

  Contain formats for control blocks, canned message buffer, disc records, magnetic tape labels, tape volume and file organization, and a description of the monitor and trace facility.

- Glossary

  Briefly describes some of the principal terms used in this manual.

Contents

FIGURES

TABLES

1. General Information

SUPERVISOR OVERVIEW

The SPERRY UNIVAC Operating System/3 (OS/3) Supervisor (supervisor) operates with user programs to provide the central control needed for most efficient use of the system hardware and software. It provides the control, interface, coordination, and allocation of hardware and controls the initiation, loading, execution, and termination of user jobs.

Modularity

The supervisor consists of control modules, each of which performs a particular set of functions or services. At system generation time, a supervisor program is produced, with modules modified and combined to provide the specific combination of capabilities that meet the needs of the particular user installation.

Minimum Main Storage Requirements

The modular design of the supervisor reduces the main storage requirement to a minimum. Frequently used modules are called resident routines because they reside permanently in main storage. Infrequently used modules are called transient routines and reside on the system resident disc volume (YRES). These transient routines are loaded from disc into main storage only when needed. They are executed as an extension of the requesting program.

Multijobbing and Multitasking Capability

In the multijobbing environment, from one to seven user jobs may be executed concurrently, with each job consisting of a series of job steps (programs). The job steps are executed in a serial manner within each job. Job steps may have from 1 to 256 tasks capable of executing concurrently with other tasks within the job step.

Minimum Operator Intervention

Operator intervention is kept to a minimum. Operating in conjunction with job control, the supervisor efficiently controls multijobbing. Most error situations are handled by the supervisor and by user-supplied error routines, so that an operator usually is not required to initiate error recovery procedures.

Interrupt Handling

The supervisor is informed of processing events through interrupts. Interrupts may be allowed or held pending to avoid simultaneous interrupts and to service interrupts based on their relative priorities. Upon recognizing an interrupt, the executing task is suspended and program control is transferred to the appropriate interrupt handler. The interrupt handler analyzes the cause of the interrupt and activates the appropriate interrupt servicing routine.

STATEMENT CONVENTIONS

The conventions used to illustrate the coding formats presented in this manual are:

- Information that must be coded exactly as shown is presented in uppercase letters. Commas, equal signs, parentheses, slashes, and percent signs must be coded as shown.
- Information that can take various forms is presented in lowercase letters. Acronyms which form part of the variable name remain capitalized.
- Unless otherwise indicated, all parameters are positional parameters and must be coded in the order they appear in the coding formats.
- Keyword parameters can be coded in any sequence.
- When more than one option exists for any given parameter, the various options are listed within braces:

  { option-1 }
  { option-2 }
  { option-n }

- Information that may be included or omitted, depending upon program requirements, is enclosed in brackets:

  [,optional-information]

- Optional parameters that have a default specification are shaded:

  [, { option-1 }]
  [{ option-2 }]
  [{ default }]

- An ellipsis (a series of three periods) is used to indicate the omission of a variable number of similar parameters:

  entry-1,...,entry-n

- When an uppercase portion of a parameter is underlined, only that portion need be coded. For example:

  PR:xv

  can be coded as either P:12 or PR:12.

MACRO INSTRUCTIONS

Table 1—1 lists the supervisor macro instructions and briefly describes the service performed by each. In this table, BCW, CCB, CCW, PIOCB, DTFPF, PCA, SAT, TCA, ECB, ARGLST, DDCPF and DCFLT are declarative macro instructions; the remainder are imperative macro instructions. The macro instructions are listed here within their related functional groups. The function and format for each macro instruction and a brief description of the parameters are given in Section 2 of this manual, where the macro instructions are arranged alphabetically by acronym for quick reference.

Table 1—1. Functional Listing of Supervisor Macro Instructions (Part 1 of 3)

PHYSICAL INPUT/OUTPUT CONTROL SYSTEM

Physical Input/Output Control

BCW	Generates buffer control word.
CCB	Generates command control block.
CCW	Generates channel command word.
EXCP	Executes channel program.
PIOCB	Generates physical input/output control block.
RDFCB	Reads file control block.

Input/Output Synchronization

WAIT	Waits for one or all input/output requests to complete.
WAITM	Waits for one of several input/output requests to complete.

SPACE MANAGEMENT

Disc

ALLOC	Assigns space to a new disc file or to an existing file.
EXTEND	Assigns additional space to an existing disc file.
OBTAIN	Accesses VTOC user block.
RENAME	Renames a disc file.
SCRTCH	Deallocates one or more disc files.

Diskette

ALLOC	Assigns space to a new diskette file.
OBTAIN	Obtains diskette label information.
SCRTCH	Deallocates a diskette file.

SYSTEM ACCESS TECHNIQUE (SAT)

Disc SAT

CLOSE	Closes a disc file.
DTFPF	Defines a partitioned file.
GET	Retrieves next logical block.
OPEN	Opens a disc file.
PCA	Defines a partition.
PUT	Outputs a logical block.
READE	Searches track by key, equal.
READH	Searches track by key, equal or higher.
SEEK	Accesses a physical block.
WAITF	Waits for block transfer.

Tape SAT

CLOSE	Closes a tape file.
CNTRL	Controls tape unit functions.
GET	Gets next logical block.
OPEN	Opens a tape file.
PUT	Outputs next logical block.
SAT	Defines a magnetic tape file.
TCA	Defines a tape control appendage.
WAITF	Waits for block transfer.

Table 1—1. Functional Listing of Supervisor Macro Instructions (Part 2 of 3)

MULTITASKING

Task Management

ATTACH	Creates and activates an additional task.
AWAKE	Reactivates an existing nonactive task.
CHAP	Changes the priority of a task.
DETACH	Terminates a task normally.
ECB	Generates an event control block.
TYIELD	Deactivates a task.

Task Synchronization

POST	Activates the waiting task.
WAIT	Waits for a task request to complete.
WAITM	Waits for one of several task requests to complete.
TPAUSE	Deactivates one or more tasks other than the issuing task.
TGO	Reactivates one or more tasks other than the issuing task.

PROGRAM MANAGEMENT

Program Loader

FETCH	Loads a program phase and branches.
LOAD	Loads a program phase and returns control.
LOADI	Locates a program phase and stores its phase header in a work area.
LOADR	Loads a program phase, relocates address-constants, and returns control.

Job and Task Termination

CANCEL	Terminates a job abnormally.
EOJ	Terminates a job step normally.

Timer Services

GETIME	Obtains current date and time.
SETIME	Sets timer interrupt for the requesting task.

Program Linkage

ARGLST	Generates an argument list.
CALL/VCALL	Calls a program.
RETURN	Restores registers and returns.
SAVE	Saves register contents.

Island Code Linkage

EXIT	Exits from island code routine.
STXIT	Links to island code routine.

System Information Control

GETCOM	Retrieves data from communication region.
GETINF	Retrieves data from system control blocks.
PUTCOM	Places data into communication region.

Table 1—1. Functional Listing of Supervisor Macro Instructions (Part 3 of 3)

PROGRAM MANAGEMENT (cont)	
Control Stream Reader	
GETCS	Retrieves embedded data file submitted in job control stream.
SETCS	Resets pointer to embedded data file.
DIAGNOSTIC AND DEBUGGING	
Storage Displays	
DUMP	Prints the job main storage contents and terminates the job step.
SNAP/SNAPF	Prints portions of main storage and returns control.
Checkpoint Facility	
CHKPT	Records a checkpoint.
DCFLT	Generates a file list table.
DCPCLS	Closes a disc checkpoint file.
DCPOPN	Opens a disc checkpoint file.
DDCPF	Defines a disc checkpoint file.
Monitor and Trace	
// **OPTION TRACE**	Monitors from start of job. (This is a job control statement, not a macro instruction.)
MESSAGE DISPLAY, LOGGING, AND OPERATOR COMMUNICATION	
GETMSG	Retrieves message from canned message file.
WTL	Writes a message into system log file.
WTLD	Writes a message into system log file after displaying on system console.
OPR	Displays a message to operator on system console.
OTHER SERVICES	
Spooling	
BRKPT	Creates a breakpoint in a spool output file.

2. Supervisor Macro Instructions

ALLOC

(disc space management)

Function:

Assigns space to a new disc file or to an existing disc file. Allocation is performed on a volume-by-volume basis.

Format:

LABEL	ΔOPERATIONΔ	OPERAND
[symbol]	ALLOC	{FCB-name / filename-addr / (1)} [,{error-addr / (r)}] [,{vol-seq-no,OLD,NOFCB / (0)}]

Parameters:

FCB-name
Specifies the symbolic address of the FCB.

filename-addr
Specifies the symbolic address of an 8-byte area in main storage in which you have stored the file name (as listed in the LFD job control statement for the file). NOFCB must be entered as positional parameter 5.

(1)
Specifies that register 1 has been preloaded with the address of the FCB or the address of the file name if NOFCB has been entered as positional parameter 5.

error-addr
Specifies the symbolic address of an error routine that receives control if an error occurs.

(r)
Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 2 is omitted, the calling task is abnormally terminated if an error occurs.

vol-seq-no
Specifies the volume number of a multivolume file to be allocated.

If positional parameter 3 is omitted, the value 1 is assumed.

OLD
Specifies that an old file is extended with information contained in the specified FCB and ERB rather than with the EXTEND macro instruction.

If positional parameter 4 is omitted, a new file is assumed.

ALLOC

(disc space management)

NOFCB

Specifies that positional parameter 1 refers to a file name instead of an FCB. In this case, space management issues an RDFCB macro instruction to read the FCB from the job run library file into the transient area.

If positional parameter 5 is omitted, it is assumed that positional parameter 1 refers to an FCB and that you have issued an RDFCB macro instruction for this file.

(0)

Indicates that register 0 has been preloaded with the information for positional parameters 3, 4, and 5.

Bit	
22	1 = NOFCB (See positional parameter 5.)
23	1 = OLD (See positional parameter 4.)
24—31	Volume sequence number

ALLOC

(diskette space management)

Function:

Assigns space on the diskette to a new file. The increments of allocation are sectors (128 bytes per sector). After ensuring the request is valid, the allocate routine locates space on the diskette by reading the index track. It calculates the available space on the volume from file labels 8 through 26. The routine then writes a file label on the index track to allocate space to the file.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	ALLOC	{FCB-name / filename-addr / (1)} [, {error-addr / (r)}] [, {vol-seq-no,,NOFCB / (0)}]

Parameters:

FCB-name
Specifies the symbolic address of the file control block.

filename-addr
Specifies the symbolic address of an 8-byte area in main storage in which you have stored the file name (as listed on the LFD job control card of the file). NOFCB must be entered as positional parameter 5.

(1)
Indicates that register 1 has been preloaded with the address of the file control block, or the address of the file name if NOFCB has been entered as positional parameter 5.

error-addr
Specifies the symbolic address to which control is transferred if an error is encountered.

(r)
Indicates that a register (other than 0 or 1) has been preloaded with the error address.

If positional parameter 2 is omitted, the calling task will be abnormally terminated if an error occurs.

vol-seq-no
Specifies the volume number of a multivolume file to be allocated.

If positional parameter 3 is omitted, the value 1 is assumed.

,
This parameter (positional parameter 4) is not applicable, but a comma must be entered in this position if NOFCB is specified.

ALLOC

(diskette space management)

NOFCB

Specifies that positional parameter 1 refers to a file name instead of an FCB. In this case, space management will issue an RDFCB macro instruction to read the FCB from the run library into the transient area.

If positional parameter 5 is omitted, it is assumed that positional parameter 1 refers to an FCB and that you have issued an RDFCB macro instruction for this file.

(0)

Indicates that register 0 has been preloaded with the information for positional parameters 3, 4, and 5.

Bit	
22	1 = NOFCB
23	0 = new allocation
24—31	Volume sequence number.

ARGLST

Function:

Generates an argument list (list of parameters) in the format required by the CALL/VCALL macro instruction.

This is a declarative macro instruction and must not appear in a sequence of executable code.

Format:

LABEL	Δ OPERATION Δ	OPERAND
symbol	**ARGLST**	**param-1,...,param-n**

Label:

symbol
Specifies the symbolic address of the generated parameter list. This name can be used in the CALL/VCALL macro instruction to refer to the parameter list.

Parameters:

param-1,...,param-n
Specifies one or more parameters to be included in the parameter list generated by this macro.

ATTACH

Function:

Creates and activates a task desiring control of the processor. It generates an additional TCB and enters the task on the switch list.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	ATTACH	{ECB-name / (1)}, {entry-point-name / (0)} [, {error-addr / (r)}] [,n]

Parameters:

ECB-name
Specifies the symbolic address of the ECB used to identify and control this task.

(1)
Specifies that register 1 has been preloaded with the address of the ECB.

entry-point-name
Specifies the symbolic address of the point in the program at which this task receives control. The coding to be executed for the task must be in main storage when the ATTACH macro instruction is issued.

(0)
Specifies that register 0 has been preloaded with the address of the entry point.

error-addr
Specifies the symbolic address of an error routine that receives control if an error occurs.

(r)
Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

'f positional parameter 3 is omitted, the task is abnormally terminated if an error occurs.

n
Specifies a 1-byte value to be added to the switch list priority of the originating task. This increases the dispatching priority value and results in a lesser priority for the task. (The higher the priority number, the lower the priority.) The result is assigned as the switch list priority of the new task unless it exceeds the limit of this system, in which case the highest number (lowest priority) in the system is used. The use of this parameter always results in a lesser priority. You cannot attach a task with a priority higher than that of the primary task.

If positional parameter 4 is omitted, the new task is created at the same priority as the originating task.

AWAKE

Function:

Reactivates a task deactivated by a TYIELD macro instruction. It clears the TYIELD bit within the wait bytes of the TCB regardless of whether or not the task is idle, thereby activating the task to receive control of the processor from the switcher.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	AWAKE	[{ECB-name / (1)}]

Parameters:

ECB-name
Specifies the symbolic address of the ECB of the task that is reactivated.

(1)
Specifies that register 1 has been preloaded with the address of the ECB.

If omitted, or if this macro instruction is executed with a zero address in register 1, the primary task is reactivated.

BCW

Function:

Generates a BCW that provides the hardware parameter interface to the integrated disc adapter, integrated peripheral channel, and the multiplexer channel for use by the PIOCS routines. Also, the BCW macro instruction provides you with a limited device independent interface across selector channel devices. In this case, the PIOCS routines, using the information provided in the BCW, construct a CCW chain. The BCW formats are shown in Figures A—1, A—2, and A—3.

The BCW of formatting commands sent to the SPERRY UNIVAC 8411 and SPERRY UNIVAC 8414 Disc Subsystems must specify a single record.

This is a declarative macro instruction and must not appear in a sequence of executable code.

Format:

LABEL	Δ OPERATION Δ	OPERAND
symbol	**BCW**	**[device-cmd-code] [,data-addr] [,data-flag] [,data-byte-count] [,repl-addr] [,repl-flag] [,repl-byte-count] [,control-flag]**

Label:

symbol
Specifies the symbolic address of the BCW. This symbol is used to refer to the BCW.

Parameters:

device-cmd-code
Specifies the actual device command code that directs the operation of the I/O device. (For a complete description of the command codes for a particular device, refer to the appropriate subsystem programmer reference manual.)

If positional parameter 1 is omitted, 16 bytes containing zeros are reserved for the BCW and the assembly listing contains an error note.

data-addr
Specifies the symbolic address of the data being transferred. This is the active buffer for the system console and the line adapters. For the integrated punch with the optional read feature, it is the address of the punch output buffer. This parameter is required if data is being transferred to or from main storage.

If positional parameter 2 is omitted, the data address field in the BCW is set to zeros and the assembly listing contains an error note.

data-flag
Specifies the flag byte associated with the address of the active buffer. This is written in the form X'xx' as follows:

For the integrated disc adapter:

X'40' Perform a search operation on an entire cylinder rather than a track.

X'80' Do not transfer data.

For the integrated peripheral channel:

X'20' Do not transfer data. (This entry can also be used for the multiplexer channel.)

X'80' Perform a replacement operation. If this entry is used, positional parameters 5, 6, and 7 are also required.

If positional parameter 3 is omitted, X'00' is assumed, indicating normal operation as specified by the device command code, data address, and data byte count fields in the BCW.

data-byte-count

Specifies the number of bytes to be transferred or the number of sectors to be transferred for a sectored IDA device.

If positional parameter 4 is omitted, zero is assumed. For a search on the integrated disc adapter, this indicates that the maximum number of bytes or sectors are to be transferred; and for a read or a write, this indicates no data is to be transferred. For the integrated peripheral channel, this indicates that the maximum number of bytes are to be transferred.

repl-addr

Positional parameters 5, 6, 7, and 8 apply only to the integrated peripheral channel.

Specifies the symbolic address of the second buffer area. This is the replacement buffer for the system console and the line adapters. For the integrated punch with the optional read feature, it is the address of the input buffer.

When the byte count decrements to zero during a data transfer operation, this address replaces the data address specified in positional parameter 2.

Positional parameter 3 (data-flag) must be X'80'.

repl-flag

Specifies the flag byte associated with the address of the replacement buffer. When the byte count decrements to zero during a data transfer operation, this flag byte replaces the data-flag specified in positional parameter 3. To continue the replacement operation, this entry must be X'80'.

Positional parameter 3 (data-flag) must be X'80'.

repl-byte-count

Specifies the number of replacement bytes to be transferred. When the byte count decrements to zero during a data transfer operation, this byte count replaces the data byte count specified in positional parameter 4.

Positional parameter 3 (data-flag) must be X'80'.

control-flag

Specifies the control flag for communication devices associated with the line adapters of the integrated peripheral channel. Details of this parameter and its use will be supplied later.

BRKPT

Function:

The BRKPT macro instruction creates a breakpoint in a printer or punch spool file. It closes and reopens the subfile as it is being generated by the spooler. Each segment created at this breakpoint is considered a logical subfile so that output to the physical device can be started prior to job step termination.

If this macro instruction is included in a program executing in a system that does not have the spooling capability, the macro instruction is ignored.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	BRKPT	{ filename CCB-name (1) }

Parameters:

filename
Specifies the symbolic address of the DTF macro instruction in the program which defines the file in which a breakpoint is to be created. Use this parameter if you are using data management macro instructions to define and access the file.

CCB-name
Specifies the symbolic address of the CCB associated with the file in which a breakpoint is to be created. Use this parameter if you are using PIOCS macro instructions to define and access the file.

(1)
Specifies that register 1 has already been loaded with the address of the DTF or CCB associated with the file in which a breakpoint is to be created.

CALL/VCALL

Function:

The CALL and VCALL macro instructions pass control from a program to a specified entry point in another program. They are written in the calling program to establish linkage with a called program. CALL is used to establish a direct linkage with a program already in main storage. It loads an A-type address constant, and branches. VCALL is used to establish a V-CON-type linkage with a program not necessarily in main storage. It loads a V-type address constant, and branches. No SVCs are generated by either macro instruction.

The CALL or VCALL entry point need not have a manually coded EXTRN. All other labels used on these calls, which appear outside the assembly, must have manually coded EXTRNs.

You can use positional parameter 2 of these macro instructions to pass parameters from the calling program to the called program. In this case, you can enter the parameters themselves, enclosed in parentheses; the macro expansion will generate a parameter list in the required format. Or, you can enter the address of a parameter list defined elsewhere in your program in the format required by the macro.

Another convenient method is to use the ARGLST macro instruction to generate this list for you. You then enter the symbolic address of the macro call as positional parameter 2 of the CALL or VCALL macro instruction.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	{CALL / VCALL}	{entry-point / (15)} [,{(param-1,...,param-n) / list-address / (1)}]

Parameters:

entry-point
Specifies the symbolic address of the entry point in the called program to which program control is to be given.

(15)
Indicates that register 15 has been preloaded with the address of the called program.

(param-1,...,param-n)
Specifies one or more parameters to be passed to the called program. These parameters are written enclosed in parentheses and are included in the CALL or VCALL macro expansion in the same sequence as those entered on the call line. Each parameter is considered as one full word and is aligned on a full-word boundary. The three low order bytes of each generated word contain the address of a parameter. To mark the end of the parameter list, the sign bit of the last parameter in the list is set to 1. The address loaded in register 1, prior to control being transferred to the called program, is the address of the first parameter in the list.

CALL/VCALL

The parameter entries can represent actual values. However, for compatibility with higher-level languages, this parameter is usually used to pass address constants to the called program.

list-address

Specifies the symbolic address of a user-defined parameter list. You can define the list in the required format, or you can use the ARGLST macro instruction to generate the list for you.

(1)

Indicates that register 1 has been preloaded with the address of the parameter list.

If positional parameter 2 is omitted, no parameters are assumed.

CANCEL

Function:

Causes abnormal termination of a job. It terminates the current job step, prevents execution of any remaining job steps for that job, detaches all subtasks, delinks all outstanding I/Os, and waits for all outstanding system functions to complete. CANCEL also clears all I/O locks for the task and displays an abnormal termination message on the system console indicating which job is being terminated and the error code defining the error. Unless NODUMP is entered as positional parameter 2, CANCEL provides a diagnostic storage dump of the job region similar to that produced by the DUMP macro.

CANCEL is used to terminate a job abnormally when error conditions prevent further processing. The abnormal job termination function may be requested by you through the CANCEL macro, by the operator through the CANCEL command, or as a result of a system-detected error.

If an error occurs during the execution of a macro instruction, control is passed to the error routine if an error address was specified or, if none was specified, to the abnormal termination island code if it is present. The use of island code permits you to take additional action prior to terminating the task or job step that is in error.

A number of error conditions may exist when the cancel routine is entered; however, the error code displayed in the diagnostic storage dump always represents the original cause of entry to the abnormal termination function. A printout is produced if:

- the CANCEL macro instruction does not specify NODUMP,
- the DUMP, JOBDUMP, or SYSDUMP option was specified via job control; and
- a printer was assigned to the job or is available.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	CANCEL	[{error-code / (0) / 0}] [,NODUMP]

Parameters:

error-code
Specifies a 1- to 3-digit hexadecimal error code to be displayed on the system console and included in the diagnostic storage dump.

(0)
Specifies that register 0 has been preloaded with the error code.

NODUMP
Specifies no dump regardless of the dump options specified in the OPTION job control statement.

CCB

Function:

Generates a CCB that serves as a link between the PIOCB and the BCW or the CCW. There must be at least one CCB macro instruction for each type of I/O peripheral device to be controlled by physical I/O macro instructions. An active CCB pertains to one I/O request at a time; therefore, each outstanding I/O request must have a unique CCB. The CCB format is shown in Figure A—5.

This is a declarative macro instruction and must not appear in a sequence of executable code.

The CCB is used to communicate with the PIOCS routines executing the I/O operations. The generated CCB forms the logical connection between the PIOCB and the CCW or the BCW. The PIOCB references the actual peripheral device and the CCW or the BCW defines and controls the function of the particular device and its data transfer. The CCB also specifies user options pertinent to the I/O request in the event of an error, and reflects the status of the request. When the related I/O interrupt occurs, the PIOCS also stores status information pertinent to the interruption in the associated CCB.

Format:

LABEL	Δ OPERATION Δ	OPERAND
symbol	CCB	PIOCB-name,{BCW-name / CCW-name} [,{PUB-entry-number / 0}] [,{error-option / X'00'}]

Label:

symbol
Specifies the symbolic address of the CCB. This symbol is used to refer to the CCB.

Parameters:

PIOCB-name
Specifies the symbolic address of an associated PIOCB generated by the PIOCB macro instruction. (The address furnished is modified by this macro instruction to be the address of the PUB address within the PIOCB.)

BCW-name
Specifies the symbolic address of a BCW.

CCW-name
Specifies the symbolic address of a CCW, or a list of CCWs if command chaining is used.

When you use data management macro instructions, the BCWs and CCWs are generated automatically. When using PIOCS macro instructions, you must specify each BCW and CCW according to the I/O function desired.

CCB

PUB-entry-number

May be 0, 2, 4, 6, 8, 10, 12, or 14 indicating one of eight 2-byte fields in the PIOCB containing the absolute address of the PUB for the device involved in the I/O operation. (Zero indicates the first entry, 2 the second, 4 the third, etc.)

error-option

Specifies error acceptance options elected at assembly time. This is written in the form X'xx' as follows:

X'00'	Indicates that no error conditions are acceptable.
X'02'	Reserved for system use.
X'04'	Reserved for system use.
X'08'	Indicates system access CCB. Device independence can be achieved by furnishing a BCW for integrated peripheral.
X'10'	Indicates a diagnostic request.
X'20'	Indicates that following the normal error recovery attempts by the supervisor, those errors classified as unique are acceptable.
X'40'	Indicates that all unrecoverable error conditions are acceptable following the normal error recovery attempts by the supervisor.
X'80'	Indicates user has own error code. No recovery is attempted by the supervisor, and device status and sense are communicated to user in the CCB.

CCW

Function:

Generates a CCW that provides the hardware parameter interface to the selector channels for use by the PIOCS routines. The CCW format is shown in Figure A—6.

The supervisor can only handle command chains on selector devices through two levels of transfer in channel (TIC) within command chain. This limitation is due to the lack of hardware address relocation on CCWs and the need to have a software function perform the absolutizing and relativizing of CCW addresses.

This is a declarative macro instruction and must not appear in a sequence of executable code.

Format:

LABEL	Δ OPERATION Δ	OPERAND
symbol	**CCW**	**[device-cmd-code] [,data-addr] [,flag] [,data-byte-count]**

Label:

symbol
Specifies the symbolic address of the CCW. This symbol is used to refer to the CCW.

Parameters:

device-cmd-code
Specifies the actual device command code that directs the operation of the I/O device. (For a complete description of the command codes for a particular device, refer to the appropriate subsystem programmer reference manual.)

If positional parameter 1 is omitted, eight bytes containing zeros are reserved for the CCW and the assembly listing contains an error note.

data-addr
Specifies the symbolic address of the data being transferred. This parameter is required if data is being transferred to or from main storage.

If positional parameter 2 is omitted, the data address field in the CCW is set to zeros and the assembly listing contains an error note.

flag
Specifies the flag byte associated with the address of the buffer. This is written in the form X'xx' as follows:

X'20'	Indicates incorrect data length to be suppressed.
X'40'	Indicates command chaining.
X'60'	Indicates both of above.

If positional parameter 3 is omitted, X'00' is assumed, indicating normal operation as specified by the device command code, data address, and data byte count fields in the CCW.

data-byte-count
Specifies the number of bytes to be transferred.

If positional parameter 4 is omitted, zero is assumed, resulting in a maximum data transfer.

CHAP

Function:

Changes the dispatching priority of the task issuing the instruction. The number (either positive or negative) entered as the operand is added to or subtracted from the current dispatching priority of the task (specified by the switch-priority parameter in the EXEC job control statement). This changes the dispatching priority value resulting in a lesser or greater priority for the task. A positive value will lower the priority, a negative value will raise the priority. This macro instruction does not change the priority to a specific level, instead, it adjusts the priority relative to the level under which it is executed.

The highest priority level to which you can change is to the original priority of the job step.

The lowest priority level to which you can change depends upon the number of priority levels specified by SUPGEN keyword parameter PRIORITY. (See system installation user guide/programmer reference, UP-8074, current version.)

If you try to raise or lower the priority beyond the specified boundaries, the system will automatically stop at the highest (or lowest) priority level with no error.

Format:

LABEL	△ OPERATION △	OPERAND
[symbol]	CHAP	{ n / (1) }

Parameters:

n

Specifies a value that is added to the dispatching priority for the task in order to lower its priority. To raise the priority of the task, use a negative number.

(1)

Specifies that register 1 has been preloaded with either a positive or negative increment.

CHKPT

Function:

The CHKPT macro instruction writes a series of checkpoint records to a specified checkpoint file.

Note that in the job control device assignment set for a checkpoint file, the LFD job control statement cannot use the INIT parameter, otherwise existing records will be overwritten.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	CHKPT	filename [,restart-addr] [,list-name] [,error-addr]

Parameters:

filename
Specifies the symbolic address of the macro instruction that defines the checkpoint file, which is either a DTFMT macro instruction for a magnetic tape SAM file or a DDCPF macro instruction for a disc or tape SAT file.

restart-addr
Specifies the symbolic address of a routine in your program that receives control when restarting from this checkpoint.

If positional parameter 2 is omitted, the instruction following the CHKPT macro instruction receives control.

list-name
Specifies the symbolic address of the DCFLT macro instruction that generates a list of files in your program accessed via PIOCS. This parameter is only required if you are using PIOCS files.

error-addr
Specifies the symbolic address of an error routine that receives control if an error occurs during the execution of the CHKPT macro instruction.

At the completion of a checkpoint, the checkpoint routine checks register 0. If the contents equal zero, the checkpoint completed satisfactorily and processing of the job continues. If not zero, the error routine receives control. The checkpoint error codes are described in the systems messages programmer/operator reference, UP-8076 (current version).

If positional parameter 4 is omitted, the job is abnormally terminated if an error occurs.

CLOSE

(SAT disc files)

Function:

Performs the required termination operations for a partitioned file. Once the CLOSE macro instruction has been issued for a file, only the OPEN macro instruction may reference that file.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	CLOSE	{ filename-1[,...,filename-n] (1) *ALL }

Parameters:

filename-1
Specifies the symbolic address of the DTFPF macro instruction in the program corresponding to the file to be closed.

(1)
Specifies that register 1 has been preloaded with the address of the DTFPF macro instruction.

***ALL**
Specifies that all files currently open in the job step are to be closed.

filename-n
Successive entries specify the symbolic addresses of the DTFPF macro instructions in the program corresponding to the additional files to be closed.

CLOSE

(SAT tape files)

Function:

The CLOSE macro instruction performs the required termination operations for a file; for example, construction of the EOF label group. Once the CLOSE macro instruction has been issued for a file, only the OPEN macro instruction may reference that file.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	CLOSE	{filename-1 [,...,filename-n] / (1)}

Parameters:

filename-1
Specifies the symbolic address of the SAT macro instruction in the program corresponding to the file to be closed.

(1)
Indicates that register 1 has been preloaded with the address of the SAT macro instruction.

filename-n
Successive entries specify the symbolic address of the SAT macro instructions in the program corresponding to the additional files to be closed.

CNTRL

Function:

This macro instruction initiates nondata operations on a tape unit. All tape control functions may be issued whether or not the file is open. Do not issue a WAITF macro instruction following a CNTRL macro instruction.

Format:

LABEL	ΔOPERATIONΔ	OPERAND
[symbol]	CNTRL	{filename / (1)} ,code

Parameters:

filename
Specifies the symbolic address of the corresponding SAT macro instruction in the program.

(1)
Indicates that register 1 has been preloaded with the address of the SAT macro instruction.

code
This is a mnemonic 3-character code specifying the tape unit function to be performed:

BSF	Backspace to tape mark*
BSR	Backspace to interrecord gap*
ERG	Erase gap (writes blank tape)
FSF	Forward space to tape mark*
FSR	Forward space to interrecord gap*
REW	Rewind tape
RUN	Rewind tape with interlock (unloads tape)
WTM	Write tape mark

**Applies only to input files.*

DCFLT

Function:

The DCFLT macro instruction generates a table of PIOCS files. This list is used to locate user program file definitions (PIOCB addresses) for all files accessed via PIOCS, and is required for repositioning and other file-related activities when restarting the checkpointed job.

This macro instruction is only required for PIOCS files; it is not used for data management files.

This is a declarative macro instruction and must not appear in a sequence of executable code.

Format:

LABEL	Δ OPERATION Δ	OPERAND
list-name	DCFLT	{(disc-PIOCB-1) / (tape-PIOCB-1,tmc-1,bc-1)} [{,(...),(disc-PIOCB-n) / ,(...,...,...),(tape-PIOCB-n,tmc-n,bc-n)}]

Note the use of the parentheses to delimit parameters in this macro instruction. For disc, each parameter must be enclosed by parentheses. For tape, each set of parameters must be enclosed by parentheses and separated by commas. Note also that if there are two or more PIOCS files listed in the operand field, these entries (enclosed in parentheses) are separated by commas.

Label:

list-name
Specifies the symbolic address of the PIOCS file list table.

Use this name as positional parameter 3 of the CHKPT macro instruction.

Parameters:

disc-PIOCB-1
Specifies the symbolic address of a PIOCB for a PIOCS disc file in your program.

tape-PIOCB-1
Specifies the symbolic address of a PIOCB for a PIOCS magnetic tape file in your program.

tmc-1
Specifies the symbolic address of a half word in your program containing a binary count of tape marks read between the tape load point and the current position of the magnetic tape volume.

DCFLT

bc-1

Specifies the symbolic address of a full word in your program containing a binary count of blocks (physical records) read from the most recent tape mark to the current tape position. The most recent tape mark must be located between the current tape position and the tape load point.

Successive entries specify the symbolic addresses of additional PIOCBs in your program.

Disc and magnetic tape file entries may be interspersed. Positional parameters 2 and 3 are required only for magnetic tape files. When you omit them, do not insert commas inside the parentheses.

DCPCLS

Function:

The DCPCLS macro instruction closes a disc checkpoint file defined by a DDCPF macro instruction. You must close the file after the last time the CHKPT macro instruction is executed.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	DCPCLS	{filename / (1)}

Parameters:

filename
Specifies the symbolic address of the DDCPF macro instruction that defines the checkpoint file.

(1)
Indicates that register 1 has been preloaded with the address of the DDCPF macro instruction.

DCPOPN

Function:

The DCPOPN macro instruction opens a disc checkpoint file defined by a DDCPF macro instruction. You must open the file before the first time the CHKPT macro instruction is executed.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	DCPOPN	{filename / (1)}

Parameters:

filename
Specifies the symbolic address of the DDCPF macro instruction that defines the checkpoint file.

(1)
Indicates that register 1 has been preloaded with the address of the DDCPF macro instruction.

DDCPF

Function:

The DDCPF macro instruction defines a disc file to which checkpoint records are to be written.

This is a declarative macro instruction and must not appear in a sequence of executable code.

Format:

LABEL	Δ OPERATION Δ	OPERAND
filename	DDCPF	

Label:

filename
Specifies the name of this file. This name must be entered as positional parameter 1 of the CHKPT macro instruction writing checkpoint records to this file.

There are no parameters fcr the DDCPF macro instruction.

DETACH

Function:

Terminates a task by delinking the TCB from the switch list and returning the TCB to the job's free TCB queue. If this macro instruction is executed by the primary task, it is interpreted as an end of job step instruction. All subtasks of the task being detached are also detached.

The macro instruction also clears all I/O locks for the task.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	DETACH	[{ECB-name / (1)}] [,{error-addr / (r)}]

Parameters:

ECB-name
Specifies the symbolic address of the ECB of the task to be detached.

(1)
Specifies that register 1 has been preloaded with the address of the ECB.

If positional parameter 1 is omitted, it indicates that the task issuing the DETACH instruction is terminating. No task other than the primary can terminate the primary task.

error-addr
Specifies the symbolic address of an error routine that receives control if an error occurs.

(r)
Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 2 is omitted, the executing task is abnormally terminated if an error occurs.

DTFPF

Function:

Defines a file consisting of one or more partitions. It generates a table in main storage containing the file name and operating and physical characteristics of your file that can be referenced by the system. The DTF table format is shown in Figure A—7.

This is a declarative macro instruction and must not appear in a sequence of executable code.

Format:

LABEL	△OPERATION△	OPERAND
filename	**DTFPF**	**PCA1=partition-name-1** **[,PCAn=partition-name-n]** **[,ACCESS={EXC | EXCR | SRDO | SRD | SUPD | SADD}]** **[,ALINE=YES]** **[,ERROR=error-addr]** **[,EXTENTS=n]** **[,FCB=YES]** **[,LIBUP=YES]** **[,WAIT=YES]**

Label:

filename
Specifies the name used to identify the file. This is the same as the 8-character name in the LFD job control statement.

Keyword Parameters:

PCA1=partition-name-1
Specifies symbolic address of the PCA for the first (or only) partition for the file. This partition name must be entered in the label field of the corresponding PCA macro instruction describing the partition.

PCAn=partition-name-n
Successive entries specify symbolic addresses of additional PCAs (PCA2, PCA3, etc) up to a total of seven.

ACCESS=EXC
Requests exclusive use of the file. You may read, update, and extend the file. No access is permitted by any other task. This type of filelock is the same as that requested by the LIBUP=YES parameter entry.

DTFPF

ACCESS=EXCR
Requests exclusive-read use of the file. You may read, update, and extend the file. Other tasks may also read the file, but may not write.

ACCESS=SRDO
Requests shared-read-only access to the file. You intend only to read the file. Other tasks may also read the file. No writing is permitted. This type of filelock is the same as the default of LIBUP.

ACCESS=SRD
Requests shared-read access to the file. You intend only to read the file. Other tasks may read, update, or extend the file. ←

ALINE=YES
Specifies that PCAs are to start and end on cylinder boundaries.

If the ALINE keyword parameter is omitted, PCAs start on track boundaries.

ERROR=error-addr
Specifies the symbolic address of your error routine that receives control if an error occurs.

If the ERROR keyword parameter is omitted, the job is abnormally terminated if an error occurs.

EXTENTS=n
Specifies the number of extent table entries you want in an extent table that is generated for you. The value n equals the number of physical extents allocated to the file plus the number of partitions in the file.

When standard system library files are being acessed, the entry should be EXTENTS=19. ←

If the EXTENT keyword parameter is omitted, no extent table is generated.

FCB=YES
Specifies that before issuing the OPEN macro instruction, you have placed the FCB for this file in the I/O area specified by the IOAREA1 keyword parameter of the PCA macro instruction associated with this file, instead of in the transient area where it is normally placed.

If the FCB keyword parameter is omitted, the FCB, which controls file I/O, is placed into the transient area of main storage during file-open operations.

DTFPF

LIBUP=YES

Specifies that the system library file that is referenced is reserved for exclusive use of the job step until the CLOSE macro instruction is issued to close the file. This permits you to write into the following system library files:

YLOD	Load library file
YSRC	Source library file
YOBJ	Object library file
YMAC	Macro library file
YJCS	Job control stream library file

If the LIBUP keyword parameter is omitted, the system library file being accessed cannot be written into. This is essentially a write lock for the system library files listed in the preceding paragraph.

WAIT=YES

Specifies that SAT is to issue the required WAITF macro instruction after each I/O function (GET, PUT, READE, or READH). This initiates a waiting period to assure completion of the input or output operation and sets certain status bytes in the DTF table (Figure A—7).

If the WAIT keyword parameter is omitted, you must issue a WAITF macro instruction after each I/O operation.

DUMP

Function:

Prints out the contents of a job region and terminates the job step.

The printout gives a hexadecimal representation of the contents of main storage at the time the DUMP macro instruction is issued, and includes:

- the calling job's last executed PSW and an identification code indicating the source of the dump;
- the job's 16 general registers;
- the job's prologue area with preamble and TCBs; and
- the job's program region.

The job step termination procedure is identical to that provided by the EOJ macro instruction. The DUMP macro instruction may be used anywhere the EOJ function would be applicable.

The DUMP macro instruction does not require predefined files, but does require a printer to be allocated (or available).

A dump printout is produced if:

- the DUMP, JOBDUMP, or SYSDUMP option was specified via job control; and
- a printer was assigned to the job or is available.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	DUMP	[{ identification-code / (0) / 0 }]

Parameters:

completion-code
Specifies a 1- to 3-byte hexadecimal identification code assigned by you to indicate the source of the dump.

(0)
Specifies that register 0 has been preloaded with the identification code.

If omitted, the identification code is set to binary zeros.

ECB

Function:

Generates and initializes an ECB. The ECB is used by task management to identify a subtask and to indicate status to the other tasks within a job step. The current status of the associated subtask is reflected by bits within the ECB. The ECB format is shown in Figure A—8.

The ECB is used to communicate between task management and the job step. The following programming considerations and conditions are set into the ECB.

1. The ATTACH macro instruction specifies an ECB when the subtask is created. The specified ECB is linked to the TCB and is reserved for this subtask until this subtask is detached.

2. A primary task can synchronize with other tasks by using the TYIELD and AWAKE macro instructions. Also, a primary task can issue a WAIT (or a WAITM) macro instruction to await the completion of a subtask (or subtasks). However, a primary task does not have an ECB associated with it; therefore, a subtask cannot issue a WAIT or a WAITM to await the completion of a primary task.

3. As with I/O, only one task can wait for a given CCB or for a given ECB. However, unlike I/O, which allows only the task that submitted the CCB to wait for it, task management allows only one of the other tasks to wait for the task that is identified by the ECB.

This is a declarative macro instruction and must not appear in a sequence of executable code.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	ECB	

There are no parameters for the ECB macro instruction.

EOJ

Function:

Causes normal job step termination. It terminates a primary task or a subtask. If an EOJ macro instruction is issued from a primary task with active subtasks, all subtasks are terminated. If an EOJ macro instruction is issued from a subtask, only the subtask and any subtasks it created are terminated.

This macro instruction also clears all I/O blocks for the task.

The EOJ macro instruction is usually called in by the job step task after all attached subtasks have been detached and all data files have been closed. Job control is then loaded in the user program area to prepare the next scheduled job step, or to terminate the job if it is the last job step of the job.

The EOJ macro instruction may be used to force subtask termination for the job step. If a subtask encounters an abnormal termination error condition before the EOJ function receives control, the job may be cancelled (depending on the existence and function of an abnormal termination island code routine). An EOJ macro instruction executed by a subtask is treated as a request for the DETACH macro instruction function.

The job is cancelled if errors that prevent normal termination are encountered by the EOJ routine. A hexadecimal error code is provided for display in the diagnostic storage dump produced by the CANCEL function. The error codes and their meaning are shown in the system messages programmer/operator reference, UP-8076 (current version).

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	EOJ	

There are no parameters for the EOJ macro instruction.

EXCP

Function:

Requests that an I/O operation be executed by the PIOCS.

The EXCP macro instruction communicates directly with the I/O scheduler for the purpose of submitting I/O requests to the system. Before the EXCP macro instruction is executed, you must construct an I/O packet as follows:

- Use a CCB macro instruction to define a CCB.
- Use a PIOCB macro instruction to define a PIOCB.
- Use one or more CCW macro instructions or a BCW to construct a channel program.
- Use an RDFCB macro instruction to identify the I/O device and to obtain file information specified by job control.

Linkage between these components is as follows:

- The EXCP macro instruction passes the address of the CCB to the PIOCS routines.
- The address of a 2-byte field in a PIOCB is stored in the CCB. This field contains the address of the PUB for the peripheral device concerned.
- The address of the first CCW or BCW is stored in the CCB.
- Each CCW or BCW contains the address of an input/output data area.

Whenever an EXCP macro instruction is executed, the I/O request counter in the TCB of the requester is incremented and a status indicator in the CCB is set to signify that the order is outstanding. Control is returned to the calling program immediately by the supervisor with the degree of completion of this I/O order uncertain. You must use the WAIT or WAITM macro instruction for synchronization with this I/O.

An EXCP issued to a magnetic tape which is rewinding will result in the posting of the CCB with unique error status. The EXCP should be reexecuted until the status does not occur. At that time the EXCP is considered completed.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	EXCP	{ CCB-name / (1) } [,C]

Parameters:

CCB-name
Specifies the symbolic address of the CCB.

(1)

Specifies that register 1 has been preloaded with the address of the CCB.

C

Specifies that the I/O request is conditional on the peripheral device not being shared with another job running in the system. This enables you to issue conditional seek commands when running in a multijobbing environment.

If positional parameter 2 is omitted, the I/O request is assumed to be unconditional.

EXIT

(program check or operator communications island code)

Function:

This format of the EXIT macro instruction is used for program check and operator communications island code. It terminates a user island code routine, restores the contents of the registers and the PSW, and returns program control to the point immediately following the interrupt. This macro instruction must be the last executable instruction within the island code routine.

Normally island code routines are closed routines and must be exited by either the EXIT macro instruction or the DETACH, CANCEL, DUMP, or EOJ macro instruction in the case of abnormal termination. The normal handling of island code is to either correct the error cause or flag the interrupt task, thereby notifying it of the occurrence; then, the EXIT macro instruction is executed to return control to the executing task. In the case of program check and abnormal termination, the task in error can be detached (DETACH macro instruction), the job step terminated (DUMP or EOJ macro instruction), or the job terminated (CANCEL macro instruction) without problems. Any actions beyond these are not advised within island code.

Do not use the EXIT macro instruction for abnormal termination island code. Instead, use a DUMP, EOJ, or CANCEL macro instruction, or a DETACH macro instruction if only a subtask is to be terminated.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	EXIT	{OC / PC}

Parameters:

OC

Specifies that exit is from the operator communications island code routine.

PC

Specifies that exit is from the program check island code routine.

EXIT

(interval timer island code)

Function:

This format of the EXIT macro instruction is used for interval timer island code. The macro instruction terminates the user island code routine, restores the contents of the registers and the PSW, and returns program control to the point immediately following the interrupt. If positional parameter 2 is specified, the time interval is set to the specified value before program control is returned to the task. When the specified time interval elapses, the timer island code is again executed. This macro instruction must be the last executable instruction within the island code routine.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	EXIT	IT [, { time-interval / (1) }] [, { M / S }]

Parameters:

IT

Specifies that exit is from the interval timer island code routine.

time-interval

Specifies the interval of time that must expire before the timer island code is again activated. This interval is expressed either in seconds or milliseconds depending on the entry in positional parameter 3. The maximum value that may be entered as positional parameter 1 is 4095_{10}. To specify a value greater than 4095_{10}, enter (1) as positional parameter 1 and preload register 1 with the required time interval value.

The effect of this parameter is the same as if you had issued your own SETIME macro instruction immediately before the EXIT. If you had done that, however, an interrupt occurring between the SETIME and EXIT macro instructions could possibly take control away from you long enough for your SETIME to expire and cause an error (referenced island code in busy state). To avoid this possibility, you should only reset the interval timer either when the EXIT macro instruction is issued or by a SETIME macro instruction issued outside the timer island code.

(1)

Indicates that register 1 has been preloaded with the time interval value. If omitted, the interval timer will not be reset by this macro instruction.

M

Specifies that the time interval entered as positional parameter 2 is expressed in milliseconds.

S

Specifies that the time interval entered as positional parameter 2 is expressed in seconds.

If parameter 3 is omitted, S is assumed.

EXTEND

Function:

Assigns additional space to a SAM file or a SAT file after its initial allocation of space has been exhausted. The extend routine is called by data management, or any user, only after the file runs out of space; the additional space, if available, is allocated in increments specified at the time of primary allocation.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	EXTEND	{ FCB-name / filename-addr / (1) } [,{ error-addr / (r) }] [,{ { 01 / 80 } ,{ vol-seq-no / 1 / (0) } [,FCBCORE] }]

Parameters:

FCB-name
Specifies the symbolic address of the file control block.

filename-addr
Specifies the symbolic address of an 8-byte area in main storage in which you have stored the file name (as listed on the LFD job control statement for the file to be extended).

(1)
If FCBCORE was entered as positional parameter 5 or if bit 6 of register 0 is set to 1, indicates that register 1 has been preloaded with the address of the file control block.

If positional parameter 5 is omitted, indicates that register 1 has been preloaded with the address of the file name.

error-addr
Specifies the symbolic address of an error routine that receives control if an error occurs.

(r)
Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 2 is omitted, the calling task is abnormally terminated if an error occurs.

01
Specifies that the file is a SAM file.

80
Specifies that the file is a SAT file.

EXTEND

vol-seq-no
Specifies the volume number of a multivolume file to be extended.

FCBCORE
Specifies that positional parameter 1 refers to the address of an FCB. This assumes that the user has issued an RDFCB macro instruction for this file.

If positional parameter 5 is omitted, it is assumed that positional parameter 1 refers to the address of a file name. In this case, space management will issue an RDFCB macro instruction to read the FCB from the run library into the transient area.

(0)
If filename-addr was specified as positional parameter 1, indicates that register 0 has been preloaded with the following information:

Bit	
16—23	File type
24—31	Volume sequence number

If FCB-name was specified as positional parameter 1, indicates that register 0 has been preloaded with the following information:

Bit	
6	1 = FCBCORE
16—23	File type
24—31	Volume sequence number

This is an alternative to entering FCBCORE as positional parameter 5, and assumes that the user has issued an RDFCB macro instruction for this file.

FETCH

Function:

Locates a program phase in a load library on disc, loads it into main storage, and transfers control to the address specified in the phase transfer record, unless an alternate address has been specified (in positional parameter 2).

After execution of this macro instruction, register 0 contains the job-relative address at which the phase was loaded, and register 1 contains the job-relative entry-point address. This entry-point address is determined at linkage edit time. If an alternate entry-point address is provided (positional parameter 2), the entry-point address specified to the linkage editor is overridden and the phase is given control at the specified address. This new entry-point address is returned in register 1.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	FETCH	$\left\{\begin{matrix}\text{phase-name}\\(1)\end{matrix}\right\}\left[,\left\{\begin{matrix}\text{entry-point-name}\\(0)\end{matrix}\right\}\right]$ [,R] [,DA]

Parameters:

phase-name
Specifies the name of the program phase to be loaded. This may be either the 1- to 6-character user-assigned alias phase name or the 8-character linker-assigned phase name in the format nnnnnnpp, where nnnnnn is the program name and pp is the phase number.

(1)
Specifies that register 1 has been preloaded with the address of the 8-character phase name.

entry-point-name
Specifies the symbolic address of the point in the program at which control is to be passed after a successful load.

(0)
Specifies that register 0 has been preloaded with the entry point address.

If positional parameter 2 is omitted, control is passed to the address specified in the phase transfer record.

R
Specifies that only the load library file (YLOD) is to be searched for the phase.

If positional parameter 3 is omitted, a full search is to be performed. (See *library search order* in the Glossary.)

DA
Specifies that the 8-byte phase name specified in positional parameter 1 will be overwritten with a read pointer during the first execution of this macro instruction. This read pointer is used to find the phase on the second and all subsequent executions of this macro instruction.

If positional parameter 4 is omitted, a search is performed on the phase name specified in positional parameter 1 each time this macro instruction is executed and the 8-byte phase name is not overwritten.

GET

(SAT disc files)

Function:

Reads a logical block of a partitioned file from disc into main storage and makes it accessible for processing. The address into which the data is read is specified in the associated PCA macro instruction by the keyword parameter IOAREA1.

The OPEN macro instruction initializes the current ID field in the PCA table to the start ID of the partition. If the SEQ keyword parameter in the PCA macro instruction is used, the current ID field will be updated after the wait function for each GET macro instruction is completed.

If the SEQ keyword is not used or random access is desired, it is your responsibility to preload the current ID field with the relative ID of the data block to be read. The current ID field is located at the address (label) of the PCA being referenced. This is a 4-byte field and contains a right-justified hexadecimal number representing the number of the block (relative to the first block in the partition) to be read.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	GET	{ filename / (1) } , { PCA-name / (0) }

Parameters:

filename
Specifies the symbolic address of the DTFPF macro instruction in the program corresponding to the file being read.

(1)
Specifies that register 1 has been preloaded with the address of the DTFPF macro instruction.

PCA-name
Specifies the symbolic address of the PCA macro instruction associated with the partition to be accessed.

(0)
Specifies that register 0 has been preloaded with the address of the PCA macro instruction.

GET

(SAT tape files)

Function:

The GET macro instruction reads a logical block from tape into main storage and makes it accessible for processing. The address into which the data is read is specified in the associated TCA macro instruction by the keyword parameter IOAREA1.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	GET	{filename / (1)} , {TCA-name / (0)}

Parameters:

filename
Specifies the symbolic address of the SAT macro instruction in the program corresponding to the file being read.

(1)
Indicates that register 1 has been preloaded with the address of the SAT macro instruction.

TCA-name
Specifies the symbolic address of the TCA macro instruction associated with the partition to be accessed.

(0)
Indicates that register 0 has been preloaded with the address of the TCA macro instruction.

GETCOM

Function:

Retrieves the contents of the 12-byte communication region from within the job preamble and stores it in an area specified in positional parameter 1.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	GETCOM	{to-addr / (1)}

Parameters:

to-addr
Specifies the symbolic address of a 12-byte area in main storage to which the contents of the communication region are to be moved.

(1)
Specifies that register 1 has been preloaded with the address of the area in main storage.

GETCS

Function:

The GETCS macro instruction retrieves embedded data images and control statements that were entered in the system through the job control stream. One or more data images can be retrieved at a time from the enbedded data file. The images may be from 1 to 128 bytes in length and may be obtained from more than one set of embedded data belonging to the same job step. Except for PARAM statements, each retrieved record is an exact image of the source statement. PARAM statements appear according to standard JCL conventions.

Images are read sequentially from the start of the entire data file. The sequence can be altered or the data can be reread by using the SETCS macro instruction.

Following the successful execution of a GETCS macroinstruction, program control is returned to the issuing program at the point immediately following the GETCS macroinstruction. Register 0 and 1 will contain:

R0 — The binary count of records retrieved.

— 0_{16} if a /* that terminates a data set is the first image in the input area.

— $00FFFFFF_{16}$ when all embedded data images have been read.

R1 — The reread pointer (8.8.4). When passed to the SETCS macroinstruction, it allows you to reread embedded data at this pointer.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	GETCS	{input-area / (1)} [,{number-of-records / (0) / 1}] [,{error-addr / (r)}] [,{n / 80}]

Parameters:

input-area
Specifies the symbolic address of an input area in main storage that is to receive the record or records. When more than one record is requested at a time, as each record is retrieved from the control stream, it is stored in contiguous byte locations beginning with this address. This area must be large enough to contain the retrieved records. The record image size is specified in positional parameter 4.

(1)
Specifies that register 1 has been preloaded with the address of the main storage input area.

number-of-records
Specifies the number of records requested.

GETCS

(0)

Specifies that register 0 has been preloaded with the number of records requested.

error-addr

Specifies the symbolic address of an error routine that receives control if an error occurs.

(r)

Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 3 is omitted, the calling task is abnormally terminated if an error occurs.

n

Specifies the size of the data images to be retrieved. To retrieve the entire record, make sure this value equals the data stream record size.

If images smaller than n were originally written, the returned image will be left-justified and the remainder of the input area filled to the right with spaces. If images larger than n were originally written, only the number of bytes specified in this parameter will be returned and the remaining bytes in the data stream record will be ignored.

If positional parameter 4 is omitted, 80-byte images are retrieved. If smaller images were originally written, the returned image will be left-justified and space-filled to the right. If larger images were originally written, only the first 80 bytes will be returned.

GETIME

Function:

Obtains the calendar date and the current time of day from the simulated day clock function of the supervisor. The date is returned in register 0, and the time of day is returned in register 1. If the timer services module is not part of the resident supervisor, the contents of register 1 are unpredictable.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	GETIME	[{M / S}]

Parameters:

M

Specifies that the current time of day is to be expressed in milliseconds in binary representation.

S

Specifies that the current time of day is to be expressed in packed decimal format in the form 0hhmmss(+), where hh is hours, mm is minutes, and ss is seconds, with the low-order half byte as the sign.

The current calendar date is returned in register 0 and expressed in packed decimal format in the form 0yymmdd(+), where yy is year, mm is month, and dd is day, with the low-order half byte as the sign.

GETINF

Function:

Retrieves data from the job preamble, PUB, SIB, or job TCB and stores it in a work area in main storage specified in positional parameter 2.

NOTE:

If you use the GETINF macro instruction in your program, you must reassemble your program upon every major release of the system software.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	GETINF	{PRE / PUB / SIB / TCB} , {work-area / (1)} ,number-of-bytes,displacement

Parameters:

PRE
Specifies that the data requested is from the job preamble.

PUB
Specifies that the data requested is from the PUB . In this case, register 1 must be preloaded with the address of the PUB, or with the identifying number of the PUB. The PUB identifying number is its position within the PUBs specified at SYSGEN; that is, the first PUB is 0, the second PUB is 1, and so on. Positional parameter 2 must specify work area, not (1).

SIB
Specifies that the data requested is from the SIB.

TCB
Specifies that the data requested is from the job TCB.

work-area
Specifies the symbolic address of the work area in main storage to which the data is to be moved. This area must be large enough to contain the data requested.

(1)

If positional parameter 1 is PRE, SIB, or TCB, it specifies that register 1 has been preloaded with the address of the work area.

Not valid if positional parameter 1 is PUB, since register 1 already contains the address of the PUB or the identifying number of the PUB. ←

number-of-bytes

Specifies the number of bytes of data requested.

displacement

Specifies the displacement, that is, the number of bytes from the beginning of the table to the beginning of the data requested.

GETMSG

Function:

Retrieves a message of variable length from the system canned message file, inserts the variables if any are furnished, and stores the completed message in the specified buffer area in main storage. This receiving buffer area is specified either in positional parameter 5 or 1 and must be large enough to contain the complete message text, which can be from 1 to 120 characters in length. The format of this canned message buffer and the insertion of variables are described in Appendix B.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	GETMSG	{buff-addr-1 / (1)} [,{msg-length / (0) / 60}] [,{error-addr / $(r)_3$}] [,] [,{buff-addr-2 / $(r)_4$},{buff-length-2 / $(r)_5$}]

Parameters:

buff-addr-1
Specifies the symbolic address of a buffer area in main storage containing the number of the canned message to be retrieved and any variable characters to be inserted into the message.

The first character in the canned message buffer must be a dollar sign ($). If any other type of message must start with a dollar sign, two dollar signs are required at the beginning of the message buffer.

If positional parameter 5 is blank, the retrieved message overlays this area for the length specified in positional parameter 2.

(1)
Specifies that register 1 has been preloaded with the address of the message buffer area.

msg-length
Specifies the length in bytes of the message to be retrieved from the canned message file. Length may be from 1 to 120 bytes.

(0)
Specifies that register 0 has been preloaded with the length of the buffer area.

error-addr
Specifies the symbolic address of an error routine that receives control if an error occurs.

$(r)_3$
Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

GETMSG

If positional parameter 3 is omitted, the requesting task is abnormally terminated if an error occurs.

[,]

This parameter (positional parameter 4) is not applicable, but a comma must be entered in this position.

buff-addr-2

Specifies the symbolic address of a buffer area in main storage that is to receive the retrieved message from the canned message file.

This parameter gives the caller the option of specifying another buffer that does not destroy the original.

$(r)_4$

Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the buffer area in main storage that is to receive the retrieved message.

If positional parameter 5 is omitted, the retrieved message overlays the buffer area specified in positional parameter 1 for the length specified in positional parameter 2.

buff-length-2

Specifies the length in bytes of the buffer area specified in positional parameter 5. Length may be from 1 to 120 bytes.

This parameter must be present if positional parameter 5 was specified.

$(r)_5$

Specifies that the register designated (other than 0 or 1) has been preloaded with the length of the buffer area specified in positional parameter 5.

If positional parameter 6 is omitted and positional parameter 5 was specified, the macro instruction does not execute.

LOAD

Function:

Locates a program phase in a load library on disc, loads it into main storage, and transfers control to the calling program immediately following the LOAD macro instruction.

After execution of this macro instruction, register 0 contains the job-relative address at which the phase was loaded, and register 1 contains the entry-point address. This entry point address is determined at linkage edit time. If an alternate load address is provided (positional parameter 2), the load point address specified to the linkage editor is overridden and the phase is loaded at the specified address. This new override address is returned in register 0.

This macro instruction does not relocate address constants regardless of whether an alternate load address is specified (positional parameter 2).

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	LOAD	{phase-name / (1)} [,{load-addr / (0)}] [,{error-addr / (r)}] [,R] [,DA]

Parameters:

phase-name
Specifies the name of the program phase to be loaded. This may be either the 1- to 6-character user-assigned alias phase name or the 8-character linker-assigned phase name in the format nnnnnnpp, where nnnnnn is the program name and pp is the phase number.

(1)
Specifies that register 1 has been preloaded with the address of the 8-character phase name.

load-addr
Specifies the symbolic address at which the phase is to be loaded.

(0)
Specifies that register 0 has been preloaded with the load address.

If positional parameter 2 is omitted, the program phase is loaded at the address specified by the linkage editor.

LOAD

error-addr
Specifies the symbolic address of an error routine that receives control if a load error occurs.

(r)
Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 3 is omitted, the calling task is abnormally terminated if a load error occurs.

R
Specifies that only the load library file (YLOD) is to be searched for the phase.

If positional parameter 4 is omitted, a full search is to be performed. (See *library search order* in the Glossary.)

DA
Specifies that the 8-byte phase name specified in positional parameter 1 will be overwritten with a read pointer during the first execution of this macro instruction. This read pointer is used to find the phase on the second and all subsequent executions of this macro instruction.

If positional parameter 5 is omitted, a search is performed on the phase name specified in positional parameter 1 each time this macro instruction is executed, and the 8-byte phase name is not overwritten.

LOADI

Function:

Locates the header record of a program phase and stores it in a work area.

You may then examine the information contained in the program phase header to determine if it is desirable to load the program phase. If the phase is to be loaded, you must use one of the other load instructions to load the program phase.

The format of the phase header is as follows:

Bytes	Contents
0, 1	Systems use
2	Phase number
3, 4	System flags
5—8	Phase load address (linker assigned)
9—12	Phase length
13—20	Phase name (linker assigned)
21—23	Date (packed decimal—yymmdd)
24—26	Time (packed decimal—hhmmss)
27—30	Module length
31—38	Alias phase name
39—68	Comments

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	LOADI	{phase-name / (1)} , {work-area-addr / (0)} [, {work-area-length / 13}] [, {error-addr / (r)}] [,R]

LOADI

Parameters:

phase-name
Specifies the name of the program phase to loaded. This may be either the 1- to 6-character user-assigned alias phase name or the 8-character linker-assigned phase name in the format nnnnnnpp, where nnnnnn is the program name and pp is the phase number.

(1)
Specifies that register 1 has been preloaded with the address of the 8-character phase name.

work-area-addr
Specifies the symbolic address of the area in main storage where the phase header is to be placed.

(0)
Specifies that register 0 has been preloaded with the work area address.

work-area-length
Specifies the number of bytes of the phase header that are to be placed in the work area.

If positional parameter 3 is omitted, the value 13 is assumed. This specifies that the portion of the phase header up to and including the phase load address and the phase length is to be placed in the work area.

error-addr
Specifies the symbolic address of an error routine that receives control if a load error occurs.

(r)
Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 4 is omitted, the calling task is abnormally terminated if a load error occurs.

R
Specifies that only the load library file (YLOD) is to be searched for the phase.

If positional parameter 5 is omitted, a full search is to be performed. (See *library search order* in the Glossary.)

LOADR

Function:

Locates a program phase in a load library on disc, loads it into main storage, and transfers control to the calling program immediately following the LOADR macro instruction.

After execution of this macro instruction, register 0 contains the job-relative address at which the phase was loaded, and register 1 contains the job-relative entry point address. This entry point address is determined at linkage edit time. If an alternate load address is provided (positional parameter 2), the load point address specified to the linkage editor is overridden and the phase is loaded at the specified address. This new override address is returned in register 0.

The format and operation of this macro instruction are identical to those of the LOAD macro instruction except that all address constants in the phase are relocated if an alternate load address is specified (positional parameter 2).

This macro instruction is used to load a phase at an address other than that at which it was linked.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	LOADR	{phase-name / (1)} [,{load-addr / (0)}] [,{error-addr / (r)}] [,R] [,DA]

Parameters:

phase-name
Specifies the name of the program phase to be loaded. This may be either the 1- to 6-character user-assigned alias phase name or the 8-character linker-assigned phase name in the format nnnnnnpp, where nnnnnn is the program name and pp is the phase number.

(1)
Specifies that register 1 has been preloaded with the address of the 8-character phase name.

load-addr
Specifies the symbolic address at which the phase is to be loaded.

(0)
Specifies that register 0 has been preloaded with the load address.

If positional parameter 2 is omitted, the program phase is loaded at the address specified by the linkage editor.

LOADR

error-addr

Specifies the symbolic address of an error routine that receives control if a load error occurs.

(r)

Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 3 is omitted, the calling task is abnormally terminated if a load error occurs.

R

Specifies that only the load library file (YLOD) is to be searched for the phase.

If positional parameter 4 is omitted, a full search is to be performed. (See *library search order* in the Glossary.)

DA

Specifies that the 8-byte phase name specified in positional parameter 1 will be overwritten with a read pointer during the first execution of this macro instruction. This read pointer is used to find the phase on the second and all subsequent executions of this macro instruction.

If positional parameter 5 is omitted, a search is performed on the phase name specified in positional parameter 1 each time this macro instruction is executed, and the 8-byte phase name is not overwritten.

OBTAIN

(disc space management)

Function:

Allows you to access any user block in the VTOC. You must first construct the parameter list which specifies the file, the particular area of the VTOC that is of interest to you, and the address of a buffer in main storage where you want the retrieved data stored.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	OBTAIN	{param-list / (1)}.[,{error-addr / (r)}] [,{vol-seq-no / 1}] [,FCBCORE]

Parameters:

param-list
Specifies the symbolic address of a parameter list containing the following:

Bytes 0—7
An 8-byte file name (as listed on the LFD job control card).

Byte 8
Hexadecimal function code of the requested service for the disc pack containing the volume sequence number specified by positional parameter 3:

Code	Interpretation
00	VOL1 address in form Occchhrr
01	Format 1 address in form Occchhrr
02	Format 2 address in form Occchhrr
03	Format 3 address in form Occchhrr
04	Format 4 address in form Occchhrr
05	Format 5 address in form Occchhrr
06	Format 6 address in form Occchhrr
80	Contents of VOL1 label
81	Contents of format 1 label
82	Contents of format 2 label
83	Contents of format 3 label
84	Contents of format 4 label
85	Contents of format 5 label
86	Contents of format 6 label
87	Contents of format 1—6 label record located at the disk address which is in the first word of the buffer in the form Occchhrr.

NOTE:

Addresses in the form Occchhrr are in discontinuous binary form, where ccc is the cylinder number, hh is the head number, and rr is the record number.

Bytes 9—11
Buffer address of the main storage area into which the addresses or label contents requested through byte 8 are loaded. For codes 00 through 06, the first word of the buffer contains the disc address of the label record. For code 87, you must store the disc address (in the form Occchhrr) of the label desired in bytes 0 through 3 of this buffer area.

Bytes 12—15
Symbolic address of the FCB in main storage. This field is required only if positional parameter 4 is specified.

(1)
Specifies that register 1 has been preloaded with the address of the parameter list desired.

error-addr
Specifies the symbolic address of an error routine that receives control if an error occurs.

(r)
Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 2 is omitted, the calling task is abnormally terminated if an error occurs.

vol-seq-no
Specifies the volume number of a multivolume file from which you retrieve the VTOC information.

FCBCORE
Specifies that the FCB is in main storage. The address of the FCB is contained within bytes 12—15 of the parameter list whose address is specified by positional parameter 1.

If positional parameter 4 is omitted, space management reads the FCB from disc, using the 8-byte file name contained in the parameter list.

OBTAIN

(diskette space management)

Function:

Retrieves the volume label or any file label on the index track. After ensuring that the request is valid, the obtain routine locates the requested label and returns it in a buffer area in main storage. You must construct a parameter list which specifies the type of label requested and gives the address of your buffer.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	OBTAIN	{param-list / (1)} [,{error-addr / (r)}] [,{vol-seq-no / 1}] [,FCBCORE]

Parameters:

param-list
Specifies the symbolic address of a parameter list containing the following:

Bytes 0—7

An 8-byte file name (as listed on the LFD job control card).

Byte 8

Function code specifying the type of label requested.

Code	Interpretation
80	Contents of index track label 7
81	Contents of index track label for the file name specified in bytes 0—7

Bytes 9—11

Buffer address of the storage area into which the label contents are to be loaded. This buffer must be at least 128 bytes.

Bytes 12—15

Symbolic address of the FCB in main storage. This field is required only if positional parameter 4 is specified.

OBTAIN

(diskette space management)

(1)
Indicates that register 1 has been preloaded with the address of the parameter list.

error-addr
Specifies the symbolic address to which control is transferred if an error is encountered.

(r)
Indicates that a register (other than 0 or 1) has been preloaded with the error address.

If positional parameter 2 is omitted, the calling task will be abnormally terminated if an error occurs.

vol-seq-no
Specifies the volume number of a multivolume file.

If positional parameter 3 is omitted, a value of 1 is assumed.

FCBCORE
Specifies that the FCB is in main storage. The address of the FCB is contained within bytes 12—15 of the parameter list whose address is specified by positional parameter 1.

If positional parameter 4 is omitted, space management reads the FCB from disc, usng the 8-byte file name contained in the parameter list.

OPEN

(SAT disc files)

Function:

Opens a partitioned file defined by the DTFPF and PCA macro instructions so that it can be accessed by the logical IOCS.

After the file has been defined by the DTFPF and PCA macro instructions, you must issue an OPEN macro instruction to initialize the file before any other access can be made. Use the GET macro instruction to access the first (or next) data block.

The transient routine called by the OPEN macro instruction allocates disc space to each of the partition control appendages from the VTOC file extents; these areas are then preformatted if necessary. If too little disc space has been allocated to a file to satisfy all PCA requirements, the partitions requiring space may be extended during processing.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	OPEN	{ filename-1 [,...,filename-n] } { (1) }

Parameters:

filename-1
Specifies the symbolic address of the DTFPF macro instruction in the program corresponding to the file to be opened.

(1)
Specifies that register 1 has been preloaded with the address of the DTFPF macro instruction.

filename-n
Successive entries specify the symbolic addresses of the DTFPF macro instructions in the program corresponding to the additional files to be opened.

Use this form (e.g., OPEN FILE1,FILE2) when more than one lockable file is to be accessed by a single task. This opens all the files named and applies the required read or write locks at the same time. In this way, you can avoid the possibility of two jobs locking each other out with each one waiting for the other to give up its file. The operator would then have to cancel one of the jobs to remove the stalemate and continue processing.

Multiple open should be used to open more than one file when filelock is involved to prevent a lockout between two programs contending for the same file. If any one file on an open directive cannot be opened because of lock, then none of the files are opened. In such a case, if an error address is specified on the first file that failed, control returns to that error address. An 88 (lock failure) occurs in the DTF error code (byte 56 of the DTF file table). If no error address is specified, all files specified by the OPEN macro instruction are deactivated pending the closing of those files by the locking program. This results in a DM88 (waiting for lock) console message.

OPEN
(SAT tape files)

Function:

After the file has been defined by the SAT and TCA declarative macro instructions, the OPEN macro instruction must be issued to initialize the file before any other access can be made. This macro instruction validates the DTF and TCA tables and performs any required tape positioning functions.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	OPEN	{ filename-1 [,...,filename-n] } { (1) }

Parameters:

filename-1
Specifies the symbolic address of the SAT macro instruction in the program corresponding to the file to be opened.

(1)
Indicates that register 1 has been preloaded with the address of the SAT macro instruction.

filename-n
Successive entries specify the symbolic addresses of the SAT macro instructions in the program corresponding to the additional files to be opened.

OPR

Function:

Displays a message for operator reply or information. Also, the message, and any reply, will be written to the console log if one was configured at system generation. The message may either be currently in main storage or be retrieved from the canned message file. If a canned message is specified, the macro instruction routine inserts any user-supplied variables into the message before the visual display. The format of the canned message buffer and the insertion of variables are described in Appendix B.

The buffer area to receive the completed message is specified in positional parameter 1 and must be large enough to contain the complete message text, which can be from 1 to 60 characters in length.

Use of this macro instruction is for console communication to the operator. Upon execution, program control is released until either the message is displayed, or the reply is transferred to the appropriate buffer, or an error is encountered.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	OPR	{buff-addr-1 / (1)} [,{msg-length / (0) / 60}] [,{error-addr / (r)3}] [, REPLY] [,{buff-addr-2 / (r)4}] [,{buff-length-2 / (r)5 / 60}]

Parameters:

buff-addr-1

Specifies the symbolic address of the message to be displayed. This may be either the address of a buffer area in main storage containing the complete message, or the address of a buffer area in main storage containing the canned message number and any variable characters to be inserted.

If a canned message is specified, the buffer must be at least four bytes long. (See Appendix B.) The first character in the canned message buffer must be a dollar sign ($). If any other type of message must start with a dollar sign, two dollar signs are required at the beginning of the message buffer.

If the message to be displayed is a canned message with a reply, but positional parameters 5 and 6 are omitted, the reply overlays this buffer area for the number of bytes specified in positional parameter 2.

(1)

Specifies that register 1 has been preloaded with the address of the message buffer area.

msg-length
Specifies the length in bytes of the message to be displayed. For canned messages, this specifies the length of the completed message including any inserted variable characters. If REPLY is specified in positional parameter 4, but positional parameters 5 and 6 are omitted, this is the length of the reply. Maximum length is 60 bytes.

(0)
Specifies that register 0 has been preloaded with the length of the message buffer area or the length of a canned message reply.

error-addr
Specifies the symbolic address of an error routine that receives control if an error occurs.

$(r)_3$
Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 3 is omitted, the requesting task is abnormally terminated if an error occurs.

REPLY
Specifies that a reply is required from the operator. Program control is not returned to the problem program until the operator's reply is received and available in the appropriate buffer area. The first nonblank character of the message text of the reply is stored, beginning at the first byte of the buffer area specified in positional parameter 5 for the length specified in positional parameter 6. If parameter 5 is omitted, then the buffer area specified in positional parameter 1 is overlaid for the length specified in positional parameter 2.

After the reply is received, register 0 contains the number of characters typed by the operator, including the character under the cursor. The maximum length of a reply is limited to 60 bytes or to the length of the message buffer, whichever is smaller. Replies that exceed the length of the message buffer area are truncated. If the reply is shorter than the message buffer area, the remaining positions in the buffer area are space filled. If the reply is all spaces, the buffer will be space filled.

If positional parameter 4 is omitted, the message is displayed and no reply is expected.

OPR

buff-addr-2

Specifies the symbolic address of a buffer area in main storage that is to receive a reply from the operator.

This parameter gives the caller the option of specifying an output buffer that is not destroyed by an incoming reply.

If REPLY was not specified in positional parameter 4, this field is ignored.

$(r)_4$

Specifies that the designated register (other than 0 or 1) has been preloaded with the address of the buffer area in main storage that is to receive a reply from the operator.

If positional parameter 5 is omitted and REPLY was specified in positional parameter 4, any reply overlays the buffer area specified in positional parameter 1 for the length specified in positional parameter 2.

buff-length-2

Specifies the length in bytes of the buffer area specified in positional parameter 5. Length may be from 1 to 60 bytes.

$(r)_5$

Specifies that the register designated (other than 0 or 1) has been preloaded with the length of the buffer area specified in positional parameter 5.

PCA

Function:

Defines the attributes of a partition within a file. It generates a control block in main storage containing the characteristics for the partition. There must be a PCA macro instruction for each partition within a file. The PCA table format is shown in Figure A—9.

This is a declarative macro instruction and must not appear in a sequence of executable code.

Format:

LABEL	△OPERATION△	OPERAND
partition-name	**PCA**	**IOAREA1=area-name** **,BLKSIZE=n** **[,EODADDR=end-of-data-addr]** **[,FORMAT=NO]** **[,KEYLEN=n]** **[,LACE=n]** **[,LBLK=n]** **[,SEQ=YES]** **[,SIZE=n]** **[,UOS=n]** **[,VERIFY=YES]**

Label:

partition-name
Specifies the symbolic address of the PCA table generated by this macro instruction. This must be the same name assigned to the partition by the PCA keyword parameters (PCA1, PCA2, etc) in the DTFPF macro instruction for this file.

Keyword Parameters:

IOAREA1=area-name
Specifies the symbolic address of an input/output area in main storage where the blocks are to be processed. The size of this area is specified in the BLKSIZE keyword parameter.

BLKSIZE=n
Specifies the size in bytes of the area in main storage named by the IOAREA1 keyword parameter.

EODADDR=end-of-data-addr
Specifies the point in the program at which sequential file processing should terminate. When a GET or PUT macro instruction accesses the block with the relative block number equal to the end of data ID for that partition, SAT transfers control to the end of data routine at the address specified by this parameter.

PCA

If the EODADDR parameter is used, SEQ=YES must also be specified.

If the EODADDR keyword parameter is omitted, SAT assumes there is no end of data routine for this partition and indicates that an invalid ID has been requested.

FORMAT=NO

Specifies that the space allocated to the partition is not preformatted. In this case, when your file is loaded on a disc, SAT issues a data-write command for each PUT macro instruction that references relative block numbers less than the end of data address, and then issues a format-write command for each PUT macro instruction that references a relative block number equal to the end-of-data address of the partition being accessed.

This means that data written within the existing file partition is written as update records while data written in the area outside the existing file partition is written as a new file.

Do not use FORMAT=NO with partitions using interlace.

If the FORMAT keyword parameter is omitted, the space allocated to the partition is assumed to be preformatted. This parameter is omitted when writing new files in which each block is written in format (count field followed by either a data field or a key field and data field).

KEYLEN=n

Specifies the length (3 to 255 bytes) of the key field in formatted records. (See Appendix C.) This is required if blocks are to be addressed by key.

If the KEYLEN parameter is omitted, blocks cannot be addressed by key.

LACE=n

Specifies an interlace factor for blocks on a partition so that more than one I/O operation may be performed per disc revolution.

The lace factor is calculated in two steps according to the following formula:

1. $\frac{\text{BLKSIZE}}{256} \times .535 = \text{calculated sector time}$

2. $\frac{\text{required time frame}}{\text{calculated sector time}} + 1 \text{ (rounded high)} = \text{lace factor}$

This lace factor determines how blocks are to be spaced on the track to ensure that the actual time frame (which includes both user and SAT overhead) is equal to or greater than your estimate of required time between block accesses.

Although the formula is based on the use of the SPERRY UNIVAC 8416 Disc Subsystem, SAT will adjust the lace factor to the capacity and speed of the specific device being used.

Do not use LACE if FORMAT=NO has been specified.

If the LACE keyword parameter is omitted, SAT assumes that interlace is not to be applied.

PCA

LBLK=n

Specifies the number of physical blocks (of the length specified in the BLKSIZE keyword parameter) comprising a logical block. Use this parameter when you want to retrieve more than one physical block to construct one logical block.

If the LBLK keyword parameter is omitted, one physical block comprises one logical block (LBLK=1).

SEQ=YES

Specifies sequential file processing. When the OPEN macro instruction is issued, the open transient routine sets the 4-byte current ID field at the address of the PCA being referenced to relative block 1 of the partition. The current ID is updated after the wait function for each GET or PUT macro instruction is completed.

If SEQ=YES is specified, the EODADDR keyword parameter must also be used.

If the SEQ keyword parameter is omitted, SAT does not increment the current ID field.

SIZE=n

Specifies the disc space required for the new file partition being defined. This is expressed as a percentage of the total file allocation to be initially assigned to the partition.

To calculate the SIZE entry, use the following formula:

$$\text{SIZE} = \frac{\text{BLKSIZE x percentage}}{\text{total}}$$

If the SIZE keyword parameter is omitted, the new file partition is assumed to require one percent of the total file allocation (SIZE=1).

UOS=n

Specifies the increment to be allocated to a partition when additional space is required. This is specified as a percentage of secondary allocation called unit of store (UOS). Each time an attempt is made to write a block with a relative block number larger than the current maximum for the partition, an amount of space equal to the unit of store is added to the partition.

If the block number exceeds the new maximum after allocating one unit of store, an invalid ID indication will be posted in the error field in the DTF table in main storage (Figure A—7).

If the UOS keyword parameter is omitted, the unit of store is assumed to be one percent of secondary allocation (UOS=1).

VERIFY=YES

Specifies that records are to be checked for parity when writing to disc. This verification requires an additional disc rotation.

If the VERIFY parameter is omitted, blocks will not be check read for parity when written to disc.

PIOCB

Function:

Generates a PIOCB for a file that provides a buffer in main storage into which the RDFCB macro instruction transfers the FCB. The PIOCB format is shown in Figure A—10.

This is a declarative macro instruction and must not appear in a sequence of executable code.

At assembly time, the PIOCB macro instruction provides main storage space for the following information:

- Eight-byte file name (search key)

  An 8-byte character string is generated within each PIOCB. This character string is required by the RDFCB macro instruction to obtain the FCB. The characters in this 8-byte search key are identical to the characters appearing as the label of the PIOCB macro instruction.

- Half-word length field

  A 2-byte field immediately follows the 8-byte search key. This field contains a binary count of the number of bytes reserved for the FCB. This binary count ranges from a minimum of 16 to a maximum of 256. Altering the contents of this half-word field is not recommended since the field defines the size of the PIOCB and is used as the requested length of the FCB. The RDFCB macro instruction transfers only the number of bytes equal to this size or less.

- Part or all of an FCB

  A 2-byte field is reserved for each device that is allocated to a file. A maximum of eight fields is permitted. The first 2-byte field is referred to in positional parameter 3 of the CCB macro instruction as PUB entry number 0, the second field as entry 2, the third field as entry 4, and the fourth field as entry 6, etc. Following the successful completion of an RDFCB macro instruction, these PUB address fields contain the absolute addresses of the PUBs that identify the assigned devices. Device assignments indicated in the FCB are made by job control from the parameters in the LFD and DVC statements. Thus, the RDFCB macro instruction, in conjunction with the PIOCB macro instruction, dynamically links the user program with the device allocations made by job control.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	PIOCB	[{FCB-length / MAX / 16}]

PIOCB

Label:

symbol
Specifies the symbolic address of the PIOCB. This address is used to refer to the PIOCB. The characters appearing in this field become the 8-byte character string (file name) generated in the first eight bytes of the PIOCB and used as a search key by the RDFCB macro instruction to locate the FCB.

Parameters:

FCB-length
Specifies the number of bytes to be reserved within the PIOCB for the FCB. The size may be from 16 to 256 bytes. This option is used to limit the size of the PIOCB for the purpose of reading partial FCBs.

MAX
Specifies that an area is to be reserved within the PIOCB large enough to contain the complete FCB including the 8-byte file name. The size of the FCB area is stored as a binary constant in the 2-byte FCB length field in the PIOCB following the file name.

If omitted, a minimum size PIOCB of 16 bytes is generated allowing for storage of the file name, the FCB-length, and the first six bytes of the FCB data. These six bytes contain the 4-byte device type code and the absolute address of the PUB for the device assigned to the file.

Error free use of space management functions (e.g., those provided by the ALLOC and SCRTCH macro instructions) requires a fairly complete FCB. When you issue an RDFCB and PIOCB macro instruction combination to read the FCB into main storage, do not use the default value (16 bytes) in the PIOCB macro instruction. Instead, either specify the necessary FCB length, or for maximum safety, use the MAX parameter.

POST

Function:

Activates the waiting task without requiring the awaited task to terminate. When the POST macro instruction is issued by a task, the task waiting on the event completion that was posted is reactivated at the point immediately following the WAIT or WAITM macro instruction.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	POST	

The task being activated by the POST macro instruction is the one waiting for the task executing the POST macro instruction, therefore, there are no parameters for this macro instruction.

PUT

(SAT disc files)

Function:

Writes a logical block of a partitioned file from main storage to disc. The main storage address from which the data is written is specified in the associated PCA macro instruction by the keyword parameter IOAREA1.

The OPEN macro instruction initializes the current ID field in the PCA table to the start ID of the partition. If the SEQ keyword parameter in the PCA declarative macro instruction is used, the current ID field will be updated after the wait function for each PUT macro instruction that is completed.

If the SEQ keyword is not used or random access is desired, it is your responsibility to preload the current ID field with the relative ID of the data block to be written. The current ID field is located at the address (label) of the PCA being referenced. This a 4-byte field and contains a right-justified hexadecimal number representing the number of the block (relative to the first block in the partition) to be written.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	PUT	{filename / (1)},{PCA-name / (0)}

Parameters:

filename
Specifies the symbolic address of the DTFPF macro instruction in the program corresponding to the file being written.

(1)
Specifies that register 1 has been preloaded with the address of the DTFPF macro instruction.

PCA-name
Specifies the symbolic address of the PCA macro instruction associated with the partition to be written.

(0)
Specifies that register 0 has been preloaded with the address of the PCA macro instruction.

PUT

(SAT tape files)

Function:

The PUT macro instruction writes a logical block from main storage to tape. The main storage address from which the data is written is specified in the associated TCA macro instruction by the keyword parameter IOAREA1.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	PUT	{ filename / (1) } , { TCA-name / (0) }

Parameters:

filename
Specifies the symbolic address of the SAT macro instruction in the program corresponding to the file being written.

(1)
Indicates that register 1 has been preloaded with the address of the SAT macro instruction.

TCA-name
Specifies the symbolic address of the TCA macro instruction associated with the partition to be written.

(0)
Indicates that register 0 has been preloaded with the address of the TCA macro instruction.

PUTCOM

Function:

Moves the contents of a 12-byte area in main storage specified in positional parameter 1 to the communication region within the job preamble.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	PUTCOM	{ from-addr / (1) }

Parameters:

from-addr
Specifies the symbolic address of a 12-byte area in main storage containing the data characters to be moved into the communication region within the job preamble.

(1)
Specifies that register 1 has been preloaded with the address of the area in main storage.

RDFCB

Function:

Locates the FCB for a file and stores it in the PIOCB in main storage. The FCB and PIOCB formats are shown in Figure A—10.

FCBs are compiled by job control at the time the control stream is evaluated and are then stored in the job run library file on the system resident direct access device until read into the PIOCB by the RDFCB macro instruction.

Positional Parameter 1 of the RDFCB macro instruction must be the address of a PIOCB that contains an 8-byte character string identifying the desired FCB. This character string is used when locating the FCB. Any references to a physical I/O block, by means of an EXCP macro instruction, before the device assignment fields are filled by the RDFCB macro instruction results in a software validation error. Therefore, each physical I/O block should be initialized by the RDFCB macro instruction before the block is referenced by an EXCP macro instruction.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	RDFCB	{PIOCB-name / (1)} [,{error-addr / (r)}]

Parameters:

PIOCB-name
Specifies the symbolic address of the PIOCB. These characters appear in the first eight bytes of the PIOCB and are used as a search key to identify the desired FCB.

(1)
Specifies that register 1 has been preloaded with the address of the PIOCB.

error-addr
Specifies the symbolic address of an error routine that receives control if an error occurs.

(r)
Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 2 is omitted, the calling task is abnormally terminated if an error occurs.

READE

Function:

Initiates a search by key in a partitioned file for a block having a key equal to the key specified.

After a successful search, the current ID entry in the PCA table is updated to reflect the relative number of the block retrieved. However, if SEQ=YES has been specified in the PCA macro instruction, the current ID field in the PCA table is the relative record number plus 1.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	READE	{ filename / (1) }, { PCA-name / (0) }

Parameters:

filename
Specifies the symbolic address of the DTFPF macro instruction in the program corresponding to the file being processed.

(1)
Specifies that register 1 has been preloaded with the address of the DTFPF macro instruction.

PCA-name
Specifies the symbolic address of the PCA macro instruction associated with the partition to be accessed.

(0)
Specifies that register 0 has been preloaded with the address of the partition to be accessed.

READH

Function:

Initiates a search by key in a partitioned file for a block having a key equal to or higher than the key specified.

After a successful search, the current ID entry in the PCA table is updated to reflect the relative number of the block retrieved. However, if SEQ=YES has been specified by the PCA macro instruction, the current ID field in the PCA table is the relative record number plus 1.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	READH	{filename / (1)}, {PCA-name / (0)}

Parameters:

filename
Specifies the symbolic address of the DTFPF macro instruction in the program corresponding to the file being processed.

(1)
Specifies that register 1 has been preloaded with the address of the DTFPF macro instruction.

PCA-name
Specifies the symbolic address of the PCA macro instruction associated with the partition to be accessed.

(0)
Specifies that register 0 has been preloaded with the address of the partition to be accessed.

RENAME

Function:

Assigns a new physical file name to any file except a system scratch file. This is accomplished by specifying the new name to be used in place of the file ID as contained in the format 1 label. Do not issue the RENAME macro instruction to a file that is currently open.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	RENAME	{param-list / (1)} [,{error-addr / (r)}] [,{vol-seq-no / 1}] ,[FCBCORE]

Parameters:

param-list
Specifies the symbolic address of a parameter list containing the 8-byte file name (as listed on the LFD job control card) and a new 44-byte file identifier. If FCBCORE is specified, the parameter list also contains a 3-byte symbolic address for the FCB.

(1)
Specifies that register 1 has been preloaded with the address of the parameter list.

error-addr
Specifies the symbolic address of an error routine that receives control if an error is encountered.

(r)
Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 2 is omitted, the calling task is abnormally terminated if an error occurs.

vol-seq-no
Specifies the volume number of a multivolume file to be renamed.

FCBCORE
Specifies that the FCB is in main storage. The symbolic address of the FCB is contained in the parameter list.

If omitted, the FCB is read from disk using the 8-byte file name contained in the parameter list.

RETURN

Function:

The RETURN macro instruction is written at the exit point of the called program. It restores the contents of the calling program registers, branches back to the calling program, and reserves storage for the current save area. All code is generated inline with no inner subroutine calls or SVCs.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	RETURN	[(r1,r2)] [,T] [,SA={savearea-name / *}]

Parameters:

(r1,r2)
Specifies that the registers designated in r1 through r2 are to be restored from the calling program save area. The address of the save area is assumed to be in register 13. All combinations or valid r1 and r2 register addresses are acceptable. If r1 > r2 the register addresses wrap around from 15 to 0. If register 13 is included within this range, it is ignored. However, if the SA parameter is coded, register 13 is reloaded from word 2 of the save area before the registers are restored.

If positional parameter 1 is omitted, no registers are restored by this parameter.

T
Specifies that if the return and entry point registers (14 and 15) are not restored by positional parameter 1, these registers are to be restored from the calling program save area (words 4 and 5).

If positional parameter 2 is omitted, registers 14 and 15 are not saved by this parameter.

Keyword Parameters:

The SA keyword parameter creates a 72-byte save area, or else it indicates that you have created the save area elsewhere in the routine. It reloads register 13 (from word 2 of this program's save area) with the pointer to the calling program's save area. It generates a branch via register 14 as the last executable instruction.

SA-savearea-name
Specifies the symbolic address of a 72-byte register save area to be created by this macro instruction. The format of the register save area is shown in Figure A—11.

SA=*
Specifies that you have defined a save area elsewhere in the routine.

If the SA keyword parameter is omitted, a save area is not created by this macro instruction, and register 13 is unaltered.

RENAME

Function:

Assigns a new physical file name to any file except a system scratch file. This is accomplished by specifying the new name to be used in place of the file ID as contained in the format 1 label. Do not issue the RENAME macro instruction to a file that is currently open.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	RENAME	{param-list / (1)} [,{error-addr / (r)}] [,{vol-seq-no / 1}]

Parameters:

param-list
Specifies the symbolic address of a parameter list containing the 8-byte file name (as listed on the LFD job control card) and a new 44-byte file identifier.

(1)
Specifies that register 1 has been preloaded with the address of the parameter list.

error-addr
Specifies the symbolic address of an error routine that receives control if an error is encountered.

(r)
Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 2 is omitted, the calling task is abnormally terminated if an error occurs.

vol-seq-no
Specifies the volume number of a multivolume file to be renamed.

RETURN

Function:

The RETURN macro instruction is written at the exit point of the called program. It restores the contents of the calling program registers, branches back to the calling program, and reserves storage for the current save area. All code is generated inline with no inner subroutine calls or SVCs.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	RETURN	[(r1,r2)] [,T] [,SA={savearea-name / *}]

Parameters:

(r1,r2)
Specifies that the registers designated in r1 through r2 are to be restored from the calling program save area. The address of the save area is assumed to be in register 13. All combinations or valid r1 and r2 register addresses are acceptable. If r1 > r2 the register addresses wrap around from 15 to 0. If register 13 is included within this range, it is ignored. However, if the SA parameter is coded, register 13 is reloaded from word 2 of the save area before the registers are restored.

If positional parameter 1 is omitted, no registers are restored by this parameter.

T
Specifies that if the return and entry point registers (14 and 15) are not restored by positional parameter 1, these registers are to be restored from the calling program save area (words 4 and 5).

If positional parameter 2 is omitted, registers 14 and 15 are not saved by this parameter.

Keyword Parameters:

The SA keyword parameter creates a 72-byte save area, or else it indicates that you have created the save area elsewhere in the routine. It reloads register 13 (from word 2 of this program's save area) with the pointer to the calling program's save area. It generates a branch via register 14 as the last executable instruction.

SA-savearea-name
Specifies the symbolic address of a 72-byte register save area to be created by this macro instruction. The format of the register save area is shown in Figure A—11.

SA=*
Specifies that you have defined a save area elsewhere in the routine.

If the SA keyword parameter is omitted, a save area is not created by this macro instruction, and register 13 is unaltered.

SAT

Function:

The SAT macro instruction defines a magnetic tape file to be processed by the System Access Technique. It generates a DTF table in main storage that identifies the name of the file and its operating and physical characteristics that can be referenced by the system.

This is the DTF macro instruction for TSAT files. The assembler also accepts the name DTFPF, but the name SAT is used here to avoid confusion between the DTF macro instruction for disc SAT files and tape SAT files.

This is a declarative macro instruction and must not appear in a sequence of executable code.

Format:

LABEL	△OPERATION△	OPERAND
filename	**SAT**	**TCA=TCA-name** **[,CKPTREC=YES]** **[,ERROR=error-addr]** **[,FCB=YES]** **[,WAIT=YES]**

Label:

filename
Specifies the name used to identify the file. This is the same as the 8-character name in the LFD job control statement.

Keyword Parameters:

TCA=TCA-name
Specifies the symbolic address of the TCA for the file. This name must be entered in the label field of the corresponding TCA macro instruction describing the tape control appendage.

CKPTREC=YES
Specifies that any checkpoint records occurring in an input tape file are to be bypassed by TSAT. In this case, your BLKSIZE specification in the TCA macro instruction must equal or exceed the length of a header or trailer label of the checkpoint set.

In OS/3 tape files, the first and last blocks of a checkpoint dump begin with the following:

//△CHKPT△//nnttCsss

SAT

where:

nn
Is the number, in binary, of image records plus control blocks, less 1, not including the header or trailer labels.

tt
Is the total number, in EBCDIC, of checkpoint records following the header label, including the trailer label; *tt* is 00 in a trailer label.

C
Is a constant, coded in EBCDIC as shown.

sss
Is the serial number of the checkpoint, in EBCDIC.

If the CKPTREC keyword parameter is omitted, any checkpoint records occurring are accepted as data by TSAT, and your program must include the coding to recognize them.

ERROR=error-addr
Specifies the symbolic address of your error routine that receives control if an error occurs.

If the ERROR keyword parameter is omitted, the job is abnormally terminated if an error occurs.

FCB=YES
Specifies that before issuing the OPEN macro instruction, you have placed the FCB for this file in the I/O area specified by the IOAREA1 keyword parameter of the TCA macro instruction associated with this file, instead of in the transient area where it is normally placed.

If the FCB keyword parameter is omitted, the FCB, which controls file I/O, is placed into the transient area of main storage during file-open operations.

WAIT=YES
Specifies that TSAT is to issue the required WAITF macro instruction after each I/O function (GET, PUT). This initiates a waiting period to assure completion of the input or output operation and sets certain status bytes in the DTF table.

If the WAIT keyword parameter is omitted, you must issue a WAITF macro instruction after each I/O operation.

SAVE

Function:

The SAVE macro instruction is written at the entry point of the called program. It saves the contents of the calling program registers, loads one or more base registers, establishes addressability, and sets the linking pointers of the save areas. All code is generated inline with no inner subroutine calls or SVCs.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	SAVE	[(r1,r2)] [,T] [,COVER={r / (r1,r2,...,rn) / 15}] [,COVADR={base-addr / *}] [,SA=savearea-name]

Parameters:

(r1,r2)
Specifies that the registers designated in r1 through r2 are to be saved in the calling program save area. The registers are always stored in their respective fields of the save area. For example, if register 2 is specified, it is stored in word 8. All combinations of valid r1 and r2 register addresses are acceptable. If r1 > r2 the register address wrap around from 15 to 0. If register 13 is included within this range, it is ignored. However, if the SA keyword parameter is coded, the contents of register 13 is stored in the save area specified.

If positional parameter 1 is omitted, no registers are saved by this parameter.

T
Specifies that if the return and entry point registers (14 and 15) are not saved by positional parameter 1, these registers are to be stored in the calling program save area in words 4 and 5.

If positional parameter 2 is omitted, registers 14 and 15 are not saved by this parameter.

Keyword Parameters:

The COVER and COVADR keyword parameters are used to establish addressability. The values specified by COVADR are loaded in the registers specified by COVER.

COVER=r
Specifies the register designated as base register for the called program.

COVER=(r1,r2,...,rn)
Specifies the registers to be designated as base registers. A total of nine registers can be designated.

If the COVER keyword parameter is omitted, register 15 is assumed to be the base register.

SAVE

COVADR=base-addr

Specifies the base address for the called program. If only one register is specified by the COVER keyword parameter, this base address is loaded in that register. If several registers are specified by the COVER keyword parameter, they are successively loaded with 4096 increments of COVADR. A USING statement is generated indicating the base address and all cover registers, regardless of whether this parameter is specified or omitted.

If the COVADR keyword parameter is omitted, the base address is assumed to be the address of this SAVE macro instruction, that is, the contents of the location counter at the time this macro instruction is assembled.

SA=savearea-name

Specifies the symbolic address of a 72-byte register save area. This address is loaded into register 13 after register 13 (which is assumed to contain the address of a previous save area if there is one) is stored in word 2 of the save area. This process provides linkage to a higher level save area if there is one. The format of the register save area is shown in Figure A—11.

If the SA keyword parameter is omitted, register 13 is unaltered.

SCRTCH

(disc space management)

Function:

Deallocates one or more files, identified by the 44-byte file ID and makes that space available for future use. Do not issue the SCRTCH macro instruction to a file that is currently open.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	SCRTCH	{FCB-name / (1)} [,{PREFIX / ALL / (0)}] [,{error-addr / (r)}]

Parameters:

FCB-name
Specifies the symbolic address of the FCB in main storage.

(1)
Indicates that register 1 has been preloaded with the address of the FCB in main storage.

PREFIX
Specifies that all files that have the specified 4-byte prefix are to be deallocated. The 4-byte prefix must be placed in bytes 76—79 of the FCB.

ALL
Specifies that all files whose expiration date has been exceeded are to be deallocated. The expiration date must be included in the 3-byte *expiration date* field of the FCB.

SCRTCH

(disc space management)

(0)

Specifies that bits 24 through 31 of register 0 have been preloaded with a hexadecimal code:

Code	Interpretation
00	Scratch file.
82	Scratch all by date.
83	Scratch by prefix.

If positional parameter 2 is omitted, the file specified by the 44-byte file ID in the FCB is scratched.

error-addr

Specifies the symbolic address of an error routine that receives control if an error occurs.

(r)

Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 3 is omitted, the calling task is abnormally terminated if an error occurs.

SCRTCH

(diskette space management)

Function:

Deallocates diskette space for a file and makes it available for future use. After ensuring that the request is valid, the scratch routine searches the file labels for a file identifier (17 bytes) that matches the first 17 bytes of the 44-byte file ID retrieved from the FCB. If a match occurs, the file's extent is scratched by marking the file label 'deleted'. Do not issue the SCRTCH macro instruction to a file that is currently open.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	SCRTCH	{ FCB-name / (1) } [,] [,{ error-addr / (r) }]

Parameters:

FCB-name
Specifies the symbolic address of the file control block (FCB) in main storage.

(1)
Indicates that register 1 has been preloaded with the address of the FCB in main storage.

[,]
This parameter is not applicable, but a comma must be entered in this position if positional parameter 3 is used.

error-addr
Specifies the symbolic address that receives control if an error is encountered.

(r)
Indicates that a register (other than 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 3 is omitted, the calling task is abnormally terminated if an error occurs.

SEEK

Function:

Initiates movement of the disc read/write head to the position specified in the current ID field of the PCA. This is a 4-byte field that contains a right-justified hexadecimal number representing any block number on the track (relative to the first block in the partition) to which head movement is to be initiated. It is your responsibility to store the desired relative block number in this field.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	SEEK	$\left\{\begin{matrix}\text{filename}\\(1)\end{matrix}\right\},\left\{\begin{matrix}\text{PCA-name}\\(0)\end{matrix}\right\}$

Parameters:

filename
Specifies the symbolic address of the DTFPF macro instruction in the program corresponding to the file being accessed.

(1)
Specifies that register 1 has been preloaded with the address of the DTFPF macro instruction.

PCA-name
Specifies the symbolic address of the PCA macro instruction associated with the partition to be accessed.

(0)
Specifies that register 0 has been preloaded with the address of the PCA macro instruction.

SETCS

Function:

Alters the sequence in which a subsequent GETCS macro instruction retrieves embedded data images from the job control stream. To do this, you may back up the GETCS pointer, skip backward or forward to the start of any embedded data set, or resume sequential reading of the data file at the beginning of the next data set.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	SETCS	{ data-set-no / NEXT / pointer / (1) } [,{R / S}] [,{error-addr / (r)}]

Parameters:

data-set-no
The number of the embedded data set from which subsequent GETCS macro instructions are to retrieve data images. Data sets are numbered sequentially starting with 1.

NEXT
Specifies that subsequent GETCS macro instructions are to retrieve data images, starting at the beginning of the next data set.

pointer
Specifies the symbolic address of a full-word embedded data file pointer provided by a previous GETCS macro instruction.

Upon successful completion of a GETCS macro instruction, control is returned to the program at the point immediately following the GETCS macro instruction, and register 1 contains a pointer to the last set of data images read from the embedded data file in the job run library file. When passed to the SETCS macro instruction, it allows embedded data to be reread starting at the pointer. Note that the pointer points to the *first* data image.

(1)
Specifies that register 1 has been preloaded with either the 4-byte GETCS pointer itself or with a data set number.

R
Specifies that the entry in positional parameter 1 is the address of the reread pointer provided by a previous GETCS macro instruction. ←

S
Specifies that the entry in positional parameter 1 is a data set number.

SETCS

error-addr
Specifies the symbolic address of an error routine that receives control if an error occurs.

(r)
Indicates that the register designated (other 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 3 is omitted, the calling task is abnormally terminated if an error occurs.

SETIME

Function:

Requests a scheduled timer interrupt in the requesting task and continues executing the requesting task. When the specified time interval elapses, the task's timer island code (as specified by a STXIT macro instruction) is executed.

Note that in this case the STXIT macro instruction must have been previously issued to set up timer island code for this task. There may be only one set of timer island code per task.

If written with the WAIT parameter, this macro instruction requests a timer interrupt and suspends execution of the requesting task until the timer interval elapses. At this time, the task resumes execution with the next instruction following the SETIME macro instruction.

This macro instruction cancels any previous SETIME request if entered with no parameters.

Format:

LABEL	ΔOPERATIONΔ	OPERAND
[symbol]	SETIME	$\left[\left\{\begin{matrix}\text{time-interval}\\(1)\end{matrix}\right\}\right]$ [,WAIT] $\left[,\left\{\begin{matrix}M\\S\end{matrix}\right\}\right]$

Parameters:

time-interval
Specifies the interval of time that must expire before the interrupt is generated. This interval is expressed either in seconds or milliseconds, depending on the entry in positional parameter 3. The maximum value that may be entered as positional parameter 1 is 4095_{10}. To specify a greater value, enter (1) as positional parameter 1 and preload register 1 with the required time interval value.

(1)
Indicates that register 1 has been preloaded with the time interval value.

If positional parameters 1, 2, and 3 are omitted, any previous SETIME request for this task is cancelled, preventing the scheduled interrupt.

WAIT
Specifies that the problem program is to relinquish control until the specified time interval expires, at which time control is returned to the point immediately following the SETIME macro instruction.

If positional parameter 2 is omitted, the requesting program retains program control. When the time interval expires, the timer island code is activated.

M
Specifies that the time interval entered as positional parameter 1 is expressed in milliseconds.

S
Specifies that the time interval entered as positional parameter 1 is expressed in seconds.

SNAP/SNAPF

Function:

The SNAP and SNAPF macro instructions print out the contents of a job's 16 general registers, as well as selected areas of main storage. Up to 50 separate areas may be specified in a single SNAP or SNAPF macro instruction statement.

The printout gives a hexadecimal representation of the contents of the registers and storage at the time the macro instruction is issued, with program relative addresses on the left and absolute addresses on the right of each line.

The SNAP and SNAPF macro instructions perform the same function, but SNAPF is designed for use in a spooling environment. It directs the snapshot dump to a specified allocated printer or to a spool file via a virtual printer. If the snapshot dump is placed on a spool file other than the job log file, the printed output can be obtained prior to job termination by using the spool breakpoint feature or closing the file. When the SNAPF macro instruction is used, register 0 must be preloaded with the address of either an allocated printer or a virtual printer physical unit block (PUB), as obtained from execution of either a data management OPEN or a read file control block (RDFCB) macro instruction.

The contents of register 1 are destroyed by either macro instruction. If you want to record the true contents of register 1, you can store it in a field within the area of main storage to be printed. If you do not specify full word addresses, the macro instruction uses the nearest half-word location to the left of the address specified.

A snapshot printout is produced if:

- the DUMP, JOBDUMP, or SYSDUMP option was specified via job control; and
- a printer was assigned to the job or is available.

You can use this macro instruction to print out only the 16 general registers by omitting the parameters.

This macro instruction does not terminate the job step; control is returned to the calling job.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	{SNAP / SNAPF}	[{start-addr-1,end-addr-1 [,...,start-addr-n,end-addr-n] / (1)}]

Parameters:

start-addr-1
Specifies the symbolic starting address of the first area of main storage to be printed.

end-addr-1
Specifies the symbolic ending address of the first area of main storage to be printed.

SNAP/SNAPF

start-addr-n,end-addr-n

Successive pairs of parameters specify the starting and ending addresses of additional areas in main storage to be printed. A total of 50 areas may be specified in one SNAP macro instruction statement.

(1)

Indicates that register 1 has been preloaded with the address of a predefined list of one or more full-word address pairs specifying the main storage area or areas to be dumped. The leftmost bit of the last end-addr specified must be set to 1 to indicate the end of the list to the SNAP routine.

If this macro instruction is written without parameters, only the contents of the 16 general registers are printed.

STXIT

(abnormal termination, interval timer, or program check island code linkage)

Function:

Establishes or terminates linkage between your task and the user island code routine specified by the parameters. If only parameter 1 is supplied, the previous linkage with the type of island code specified is terminated.

If a program check or an abnormal termination condition occurs for which no linkage is provided, the task is terminated. If the task is a primary task, the entire job is terminated; if it is a subtask, only the subtask is terminated.

If a timer interrupt occurs for which no linkage is provided, the interrupt is ignored.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	STXIT	{AB / IT / PC} [,{entry-point / (1)} ,{save-area / (0)}]

Parameters:

AB

Establishes linkage with the abnormal termination island code routine, or else terminates linkage if no other parameters are specified.

Abnormal termination island code is established to allow a job step to monitor the error termination of the job. It receives control when a task enters cancel processing. The cancel can be either intentional (execution of CANCEL macro instruction) or unintentional, as with a system-imposed cancellation due to a software-detected error. This island code allows the abnormal termination island code to intervene when an error could force the system or subtask to terminate the job. Abnormal termination island code must exit via the DETACH, CANCEL, DUMP, or EOJ macro instructions, at least terminating the task in error.

When control is received by the abnormal termination island code, the least significant 12 bits of register 0 contain an error status code and register 1 contains the ECB address of the subtask in error, with a content of zero indicating primary task.

Abnormal termination island code is entered, executing under the cancelled task's TCB, and is not reentered until it has exited via DETACH, CANCEL, DUMP, or EOF macro instructions. Multiple task cancellations are queued for entry into the one abnormal termination island code routine for the job step.

STXIT

(abnormal termination, interval timer, or program check island code linkage)

IT

Establishes linkage with the interval timer island code routine, or else terminates linkage if no other parameters are specified.

Timer island code receives control as the result of the expiration of a timer value for a task which is not waiting for the timer. The task has executed the SETIME macro instruction without the WAIT parameter.

Upon receipt of control at timer island code entry point, register 1 contains the ECB address of the task for which the timer expired. If a timer expires for a task not having timer island code, the interrupt is ignored.

In a multitasking environment, each task using the SETIME macro instruction without the WAIT parameter must establish a timer island code save area and entry point. However, only one island code routine is required for this function and the entry point can be repeated for each task. In this case, timer island code routine should be coded to be reentrant or the task can establish discrete routines. Exit from the timer island code routine must be via the EXIT macro instruction.

PC

Specifies linkage with the program check island code routine.

Program check island code receives control as the result of a hardware program check. This condition does not necessarily indicate an error condition since occurrences such as arithmetic overflow can cause the interrupt. You can take whatever action is necessary to correct the situation and return to the interrupted routine by executing the EXIT macro instruction.

Upon receipt of control at the program check island code entry point, the least significant eight bits of register 0 contain an error status code and register 1 contains the ECB address of the task causing the error (zero indicates primary task). All other registers are as they were when the task was interrupted. The error codes are listed and described in the system messages programmer/operator reference, UP-8076 (current version).

In a multitasking environment, a program check island code save area and entry point should be specified for each task. The entry point can be to a common program check island code routine, in which case the routine must be reentrant, or it is possible to have discrete entry points for each task. If a subtask causes a program check and no island code is established for the task, abnormal termination island code is called via a cancellation of the subtask. If no abnormal termination island code is specified for this job step, the job is cancelled.

If only positional parameter 1 is specified, the previous linkage with the particular user island code routine is terminated; otherwise, a linkage is established.

entry-point

Specifies the symbolic address of the entry point of the user island code routine that processes the interrupt.

(1)

Specifies that register 1 has been preloaded with the address of the entry point.

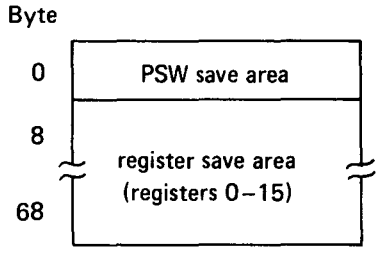
If positional parameters 2 and 3 are omitted, the previous linkage with the island code specified in positional parameter 1 is terminated.

STXIT

(abnormal termination, interval timer, or program check island code linkage)

save-area

Specifies the symbolic address of an 18-word save area for PSW and general register storage. This save area must be aligned on a full-word boundary. The format for the save area is:



(0)

Specifies that register 0 has been preloaded with the address of the save area.

If positional parameters 2 and 3 are omitted, the previous linkage with the island code specified in positional parameter 1 is terminated.

STXIT

(operator communication island code linkage)

Function:

Establishes or terminates linkage between your task and the user island code routine specified by the parameters. If only parameter 1 is supplied, the previous linkage with the operator communication island code is terminated.

If an unsolicited system console message interrupt occurs for which no linkage is provided, the interrupt is ignored.

Operator communication island code is activated when the operator enters an unsolicited message at the system console. This routine receives control under the primary task's TCB, with the job step environment stored in the operator communication island code save area and the unsolicited message within the message buffer. Multiple activations are not possible since this subroutine verifies the existence and availability of operator communication island code.

When the control is received by the operator communication island code, register 0 contains the length of the message entered by the operator. Register 1 contains either a zero, indicating that the operator communication was initiated at the console, or a negative sign, indicating that the operator communication was initiated at the workstation.

Exit from this island code routine should be via an EXIT macro instruction.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	STXIT	OC [,{entry-point,save-area,msg-area,length / (1)}]

Parameters:

OC

Establishes linkage with the operator communication island code routine, or else terminates linkage if no other parameters are specified.

If only positional parameter 1 is specified, the previous linkage with the operator communication island code routine is terminated; otherwise, a linkage is established.

entry-point

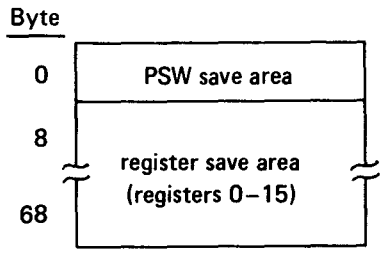
Specifies the symbolic address of the entry point of the operator communication user island code routine that processes the interrupt.

STXIT

(operator communication island code linkage)

save-area

Specifies the symbolic address of an 18-word save area for PSW and general register storage. This save area must be aligned on a full-word boundary. The format for the save area is:



msg-area

Specifies the symbolic address of an input area reserved for unsolicited messages from the operator.

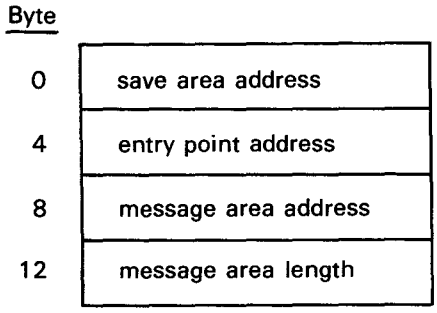
length

Specifies the length (in bytes) of the message area. The size of the area can be from 1 to 60 bytes; any message exceeding the specified length is truncated, while any message smaller is left-justified and space filled.

If positional parameters 2, 3, 4, and 5 are omitted, the previous linkage with the operator communication island code subroutine is terminated.

(1)

Specifies that register 1 has been preloaded with the address of a 4-word table containing parameters 2, 3, 4, and 5 in the following format:



DELETION

Page 2—97 has been deleted.

TCA

Function:

The TCA macro instruction defines the logical attributes of a magnetic tape file to be processed by TSAT. It generates a tape control appendage to the DTF table for the file.

This is a declarative macro instruction and must not appear in a sequence of executable code.

Format:

LABEL	△OPERATION△	OPERAND
TCA-name	TCA	IOAREA1=area-name
		,BLKSIZE=n
		[,BKNO=YES]
		[,CLRW={NORWD / RWD}]
		[,EOFADDR=end-of-data-addr]
		[,FILABL={STD / NSTD / NO}]
		[,LBLK=n]
		[,OPRW=NORWD]
		[,READ={FORWARD / BACK}]
		[,REWIND={UNLOAD / NORWD}]
		[,TPMARK=NO]
		[,TYPEFLE=OUTPUT]

Label:

TCA-name
Specifies the symbolic address of the TCA table generated by this macro instruction. This must be the same name as that which is specifjed in the TCA parameter of the SAT macro instruction for this file.

TCA

Keyword Parameters:

IOAREA1=area-name
Specifies the symbolic address of an input/output area in main storage where the blocks are to be processed. The size of this area is specified in the BLKSIZE keyword parameter.

When processing block numbered tapes (BKNO=YES), a 4-byte storage area must be reserved immediately preceding the input/output area for supervisor processing of the block number. The 4-byte block number area and the input/output area must be aligned on a full-word boundary. Do not include these four bytes as part of the IOAREA1 specification.

BLKSIZE=n
Specifies the size in bytes of the area in main storage named by the IOAREA1 keyword parameter.

When processing block numbered tapes (BKNO=YES), a 4-byte storage area must be reserved immediately preceding the input/output area. Do not include these four bytes as part of the BLKSIZE specification.

If you are reading input tapes backward (READ=BACK), your BLKSIZE specification must accommodate the largest block on tape.

BKNO=YES
Specifies that you have reserved a 4-byte storage area, aligned on a full-word boundary, immediately preceding the input/output area. Do not include these four bytes as part of either the IOAREA1 specification or the BLKSIZE specification.

This keyword parameter is required if the supervisor was generated to support block numbering and you wish to use the block number option. This enables PIOCS to write block numbers on tape output files and check them on tape input files.

CKPTREC=YES
Specifies that any checkpoint records occurring in an input tape file are to be bypassed by TSAT. In this case, your BLKSIZE specification in the TCA macro instruction must equal or exceed the length of a header or trailer label of the checkpoint set.

In OS/3 tape files, the first and last blocks of a checkpoint dump begin with the following:

//△**CHKPT**△//**nnttCsss**

where:

nn
Is the number, in binary, of image records plus control blocks, less 1, not including the header or trailer labels.

tt
Is the total number, in EBCDIC, of checkpoint records following the header label, including the trailer label; *tt* is 00 in a trailer label.

TCA

C
Is a constant, coded in EBCDIC as shown.

sss
Is the serial number of the checkpoint, in EBCDIC.

If the CKPTREC keyword parameter is omitted, any checkpoint records occurring are accepted as data by TSAT, and your program must include the coding to recognize them.

CLRW=NORWD
Specifies that a tape is not to be rewound when a file is closed.

CLRW=RWD
Specifies that a tape is to be rewound without interlock when a file is closed.

If the CLRW keyword parameter is omitted, the tape is rewound with interlock when a file is closed, which causes the tape to be unloaded from the take-up reel.

EOFADDR=end-of-data-addr
Specifies the symbolic address of your end of data routine to which TSAT transfers control when the tape mark following the last block of input data is sensed. This keyword parameter is required for all input files. The optional spelling, EODADDR, of this parameter is also acceptable.

FILABL=STD
Specifies that a tape contains standard labels.

FILABL=NSTD
Specifies that a tape contains nonstandard labels. These labels are not checked by TSAT. No provision is made in TSAT to create this type of label.

FILABL=NO
Specifies that labels are undefined or absent.

LBLK=n
Specifies the number of physical blocks (of the length specified in the BLKSIZE keyword parameter) comprising a logical block. The entry for n specifies the number of contiguous buffers supplied at the address specified by the IOAREA1 keyword parameter. Use this parameter when you want to act upon more than one physical block to construct one logical block.

If the LBLK keyword parameter is omitted, one physical block comprises one logical block (**LBLK=1**).

OPRW=NORWD
Specifies that a tape is not to be rewound before labels are checked during the processing of the OPEN macro instruction. When read backward processing is specified, NORWD is assumed. This keyword parameter must not be used if the REWIND keyword parameter is specified. If both are used, they are mutually exclusive.

If the OPRW keyword parameter is omitted, the tapes are rewound at open time.

TGO

Function:

Reactivates a specified task or tasks deactivated by a previous TPAUSE macro instruction. It is also possible to use the TGO macro instruction to reactivate all tasks of a job step (other than the issuing task) previously deactivated by TPAUSE. If the TYIELD parameter is specified, the issuing task also relinquishes control of the processor.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	TGO	{ECB-name / ALL / (1)} [,{error-addr / (r)}] [,TYIELD]

Parameters:

ECB-name
Specifies the symbolic address of the ECB of the task to be reactivated.

ALL
Specifies that all tasks of the job step are to be reactivated. Note that the calling task will not be acted upon in this case.

(1)
Specifies the register 1 has been preloaded with the address of one or more addresses pointing to the ECBs controlling the task to be activated. The last address in the list must have the X'80' bit set in the high order byte to indicate the termination of the list.

error-addr
Specifies the symbolic address of an error routine to be executed if an error occurs.

(r)
Specifies that the designated register (other than 0 or 1) has been preloaded with the address of the error routine.

An error is returned if:

1. the address or addresses passed do not point to a valid ECB,
2. the ECB does not point to an attached TCB, or
3. the ECB points to the calling task.

TYIELD
Specifies that the TYIELD function is to be performed on the issuing task at the end of TGO processing. If you use this parameter, the effect is exactly as if you had coded a TGO followed immediately by a TYIELD. If you had done this, however, your task could have been interrupted between TGO and TYIELD macro instructions and possibly could lose control to another task. This parameter eliminates that possibility and the TYIELD takes effect immediately.

TPAUSE

Function:

Causes a specified task to be deactivated until a subsequent TGO macro instruction is issued to reactivate that task. The TPAUSE macro instruction permits better control of task synchronization within a job step. It is also possible to use the TPAUSE macro instruction to deactivate all tasks of the job step other than the issuing task. This is useful to make certain that no other task, even if it had a higher priority or pending island code activation, can possibly interrupt the task and take control.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	TPAUSE	{ECB-name / ALL / (1)} [,{error-addr / (r)}]

Parameters:

ECB-name
Specifies the symbolic address of the task to be deactivated.

ALL
Specifies that all tasks of this job are to be deactivated. Note that the calling task will not be acted upon in this case.

(1)
Specifies that register 1 has been preloaded with the address of one or more addresses pointing to the ECBs controlling the task to be deactivated. The last address in the list must have the X'80' bit set in the high order byte to indicate the termination of the list.

error-addr
Specifies the symbolic address of an error routine to be executed if an error occurs.

(r)
Specifies that the designated register (other than 0 or 1) has been preloaded with the address of the error routine.

An error is returned if:

1. the address or addresses passed do not point to a valid ECB,
2. the ECB does not point to an attached TCB, or
3. the ECB points to the calling task.

TYIELD

Function:

Deactivates a task by relinquishing control of the processor and setting the TCB in a waiting state. The ECB is tested and, if there is a task awaiting the yielding task, the waiting task is activated.

The TYIELD macro instruction is used in combination with the AWAKE macro instruction which reactivates a task made dormant by the TYIELD macro instruction.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	TYIELD	

There are no parameters for the TYIELD macro instruction.

WAIT
(I/O completion)

Function:

Temporarily suspends program execution until a specified I/O operation is completed (or until all I/O operations in the task are completed). If the related operation is completed, control is returned to the point immediately following the WAIT macro instruction. If the operation is not complete, the task is placed in a wait state and control is passed to another task.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	WAIT	$\left\{\begin{matrix}\text{ALL}\\\text{CCB-name}\\(1)\end{matrix}\right\}\left[,\left\{\begin{matrix}\text{branch-addr}\\(15)\end{matrix}\right\}\right]$

Parameters:

ALL
Specifies that the I/O counter in the TCB is tested instead of the status byte in the CCB. If no orders are outstanding, the user program continues execution with the instruction following the WAIT macro instruction. If I/O orders are outstanding, the program is suspended until the I/O counter is zero (indicating all orders are completed).

CCB-name
Specifies the symbolic address of the CCB to be tested for completion.

(1)
Specifies that register 1 has been preloaded with the address of the CCB.

branch-addr
Specifies the symbolic address to which program control is transferred if the requested I/O operation is completed and an exception has occurred. The cause of the exception is posted in the appropriate CCB.

NOTE:

When a label is used as positional parameter 2, the contents of register 15 are not altered by the WAIT macro instruction, even though transfer of control may occur. Also, it is assumed that the base register coverage is provided in the user program to permit branching to this alternate address.

(15)
Specifies that register 15 has been preloaded with the branch address.

If positional parameter 2 is omitted, the WAIT macro instruction tests for complete or incomplete status without testing for exceptions. If ALL is specified as positional parameter 1, this parameter must be blank.

WAIT

(task completion)

Function:

Temporarily suspends program execution until a specified task is completed. If the related task is completed, control is returned to the point immediately following the WAIT macro instruction. If the awaited task is not complete, the issuing task is placed in a wait state and control is passed to another task.

The ECB indicates the status of the task. When a WAIT macro instruction is issued, the issuing task relinquishes control until the ECB is marked complete or until a POST macro instruction is executed by the awaited task in behalf of the waiting task.

Format:

LABEL	△ OPERATION △	OPERAND
[symbol]	WAIT	{ ECB-name / (1) }

Parameters:

ECB-name
Specifies the symbolic address of the ECB to be tested for completion.

(1)
Specifies that register 1 has been preloaded with the address of the ECB.

WAITF

(SAT disc files)

Function:

Ensures that a command initiated by a preceding GET, PUT, READE, or READH macro instruction accessing a partitioned file has been completed. When completed, the error status field contains the error status information pertaining to the I/O request. It is your responsibility to check these bits, which are in bytes 50 and 51 of the DTF table.

If the keyword parameter WAIT=YES was not specified in the DTFPF macro instruction, the WAITF macro instruction must be issued after a GET, PUT, READE, or READH macro instruction and before another imperative macro instruction is issued for that file.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	WAITF	{filename / (1)}

Parameters:

filename
Specifies the symbolic address of the DTFPF macro instruction in the program corresponding to the file being accessed.

(1)
Specifies that register 1 has been preloaded with the address of the DTFPF macro instruction.

WAITF
(SAT tape files)

Function:

The WAITF macro instruction ensures that a command initiated by a preceding GET or PUT macro instruction has been completed. When completed, the error status field contains the error status information pertaining to the I/O request. It is your responsibility to check these bits, which are in bytes 50 and 51 of the DTF table.

If the keyword parameter WAIT=YES was not specified in the SAT macro instruction, the WAITF macro instruction must be issued after a GET or PUT macro instruction and before another imperative macro instruction is issued for that file.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	WAITF	{filename / (1)}

Parameters:

filename
Specifies the symbolic address of the SAT macro instruction in the program corresponding to the file being accessed.

(1)
Indicates that register 1 has been preloaded with the address of the SAT macro instruction.

WAITM

(I/O completion)

Function:

Temporarily suspends program execution until any one of several I/O operations specified by the instruction is completed. Upon completion of one of the I/O operations, control is returned to the program at the point immediately following the WAITM macro instruction, with register 1 containing the address of the CCB associated with the I/O operation. The appropriate wait indicators are cleared with regard to the unfinished I/O operation.

When this macro instruction is executed, each referenced CCB is marked as being awaited. Upon completion of a marked CCB, the waiting task is activated and the remaining CCBs that are marked as being awaited are cleared.

The WAITM macro instruction always requires more than one event to be tested. If only one event is to be tested, use the WAIT macro instruction.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	WAITM	{ CCB-name-1,CCB-name-2[,...,CCB-name-n] / list-name / (1) }

Parameters:

CCB-name-1,CCB-name-2,...,CCB-name-n
Specifies the symbolic addresses of the CCBs to be tested that are associated with the I/O operations to be awaited. At least two CCBs must be specified.

list-name
This is a single entry which specifies the symbolic address of a list containing full-word addresses of CCBs associated with the I/O operations to be awaited. The byte following the last full word must be nonzero to indicate end of table.

(1)
Specifies that register 1 has been preloaded with the address of the list of CCB addresses.

NOTE:

The WAITM macro instruction may also specify a combination of CCB and ECB addresses as parameters. See also the multiple task wait macro instruction (WAITM for task completion).

WAITM

(task completion)

Function:

Temporarily suspends program execution until any one of several tasks specified by the instruction is completed or executes a POST macro instruction in behalf of the waiting task. Upon completion of one of the tasks, control is returned to the program at the point immediately following the WAITM macro instruction, with register 1 containing the address of the ECB associated with the task.

When this macro instruction is executed, each referenced ECB is marked as being awaited. Upon completion of a marked ECB, the waiting task is activated and the remaining ECBs that are marked as being awaited are cleared.

The WAITM macro instruction always requires more than one event to be tested. If only one event is to be tested, use the WAIT macro instruction.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	WAITM	{ ECB-name-1,ECB-name-2[,...,ECB-name-n] / list-name / (1) }

Parameters:

ECB-name-1,ECB-name-2,...,ECB-name-n
Specifies the symbolic addresses of the ECBs to be tested that are associated with the tasks to be awaited. At least two ECBs must be specified.

list-name
This is a single entry which specifies the symbolic address of a list containing full-word addresses of ECBs associated with the tasks to be awaited. The byte following the last full word must be nonzero to indicate end of list.

(1)
Specifies that register 1 has been preloaded with the address of the list of ECB addresses.

NOTE:

The WAITM macro instruction may also specify a combination of ECB and CCB addresses as parameters. See also the multiple I/O wait macro instruction (WAITM for I/O completion).

WTL

Function:

Writes a message to the system log file for subsequent printing on a high-speed printer. The message may either be currently in main storage or be retrieved from the canned message file. If you specify a canned message, the macro instruction routine inserts any user-supplied variables into the message before writing it to the log. The format of the canned message buffer and the insertion of variables are described in Appendix B.

Messages written to the log are destined for the printer and are limited to a maximum of 120 characters. Each message occupies one print line or less than a line.

Normally, job logs are printed as soon after job termination as possible. However, printing of a job log can be initiated before job termination (see BRKPT).

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	WTL	$\left\{\begin{matrix}\text{buff-addr}\\(1)\end{matrix}\right\}\left[,\left\{\begin{matrix}\text{msg-length}\\(0)\\60\end{matrix}\right\}\right]\left[,\left\{\begin{matrix}\text{error-addr}\\(r)_3\end{matrix}\right\}\right]$

Parameters:

buff-addr
Specifies the symbolic address of the message to be logged. This may be either the address of a buffer area in main storage containing the complete message or the address of a buffer area in main storage containing the canned message number and any variable characters to be inserted.

If a canned message is specified, the buffer must be at least four bytes long (see Appendix B). The first character in the canned message buffer must be a dollar sign ($). If any other type of message must start with a dollar sign, two dollar signs are required at the beginning of the message buffer.

(1)
Specifies that register 1 has been preloaded with the address of the message area.

msg-length
Specifies the length in bytes of the message to be logged. For canned messages, this specifies the length of the completed message including any inserted variable characters. Maximum length of the completed message is 120 bytes.

(0)
Specifies that register 0 has been preloaded with the length of the message.

error-addr
Specifies the symbolic address of an error routine that receives control if an error occurs.

$(r)_3$
Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 3 is omitted, the requesting task is abnormally terminated if an error occurs.

WTLD

Function:

Writes a message to the system log file for subsequent printing on a high-speed printer and simultaneously displays the message for operator reply or information. Also, the message, and any reply, will be written to the console log if one was configured at system generation. The message may either be currently in main storage or be retrieved from the canned message file. If you specify a canned message, the macro instruction routine inserts any user-supplied variables into the message before the visual display and writing to the log. The format of the canned message buffer and the insertion of variables are described in Appendix B.

Messages written to the log are destined for the printer and are limited to a maximum of 120 characters. Each message occupies one print line or less than a line.

Messages are displayed 60 characters per line on the system console, with messages longer than 60 characters occupying two lines.

When an operator reply is requested, do not use a message longer than 60 characters, because the reply is written with the message to the system log file, making a total of 120 characters.

Normally, job logs are printed as soon after job termination as possible. However, printing of a job log can be initiated before job termination (see BRKPT).

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	WTLD	{buff-addr-1 / (1)} [,{msg-length / (0) / 60}] [,{error-addr / (r)$_3$}] [,REPLY] [,{buff-addr-2 / (r)$_4$},{buff-length-2 / (r)$_5$}]

Parameters:

buff-addr-1

Specifies the symbolic address of the message to be logged and displayed. This may be either the address of a buffer area in main storage containing the complete message or the address of a buffer area in main storage containing the canned message number and any variable characters to be inserted.

If a canned message is specified, the buffer must be at least four bytes long (see Appendix B). The first character in the canned message buffer must be a dollar sign ($). If any other type of message must start with a dollar sign, two dollar signs are required at the beginning of the message buffer.

If the message to be displayed is a canned message with a reply but positional parameters 5 and 6 are omitted, the reply overlays this buffer area for the number of bytes specified in positional parameter 2.

(1)

Specifies that register 1 has been preloaded with the address of the message buffer area.

WTLD

msg-length

Specifies the length in bytes of the message to be logged and displayed. If REPLY is specified in positional parameter 4 but positional parameters 5 and 6 are omitted, this is the length of the reply.

Maximum length for the completed message is 120 bytes. If an operator reply is requested, maximum length is 60 bytes. Similar to the OPR macro instruction, a minimum of 60 characters is displayed when a canned message is specified.

(0)

Specifies that register 0 has been preloaded with the length of the message buffer area or the length of a canned message reply.

error-addr

Specifies the symbolic address of an error routine that receives control if an error occurs.

$(r)_3$

Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the error routine.

If positional parameter 3 is omitted, the requesting task is abnormally terminated if an error occurs.

REPLY

Specifies that a reply is required from the operator. Program control is not returned to the user program until the operator's reply is received, written to the log, and available in the appropriate buffer area. The message text of the reply is stored beginning at the first byte of the buffer area specified in positional parameter 5 for the length specified in positional parameter 6. If parameter 5 is omitted, then the buffer area specified in positional parameter 1 is overlaid for the length specified in positional parameter 2.

The maximum length of a reply is limited to 60 bytes or to the length of the message buffer, whichever is smaller. Replies that exceed the length of the message buffer area are truncated. If the reply is shorter than the message buffer area, the remaining positions in the buffer area are space filled.

After the reply is received, the message and the reply are written to the system log file.

If positional parameter 4 is omitted, the message is logged and displayed, with no reply expected.

buff-addr-2

Specifies the symbolic address of a buffer area in main storage that is to receive a reply from the operator.

This parameter gives the caller the option of specifying an output buffer that is destroyed by an incoming reply.

If REPLY was not specified in positional parameter 4, this field is ignored.

$(r)_4$

Specifies that the register designated (other than 0 or 1) has been preloaded with the address of the buffer area in main storage that is to receive a reply from the operator.

If positional parameter 5 is omitted and REPLY was specified in positional parameter 4, any reply overlays the buffer area specified in positional parameter 1 for the length specified in positional parameter 2.

buff-length-2

Specifies the length in bytes of the buffer area specified in positional parameter 5. Length may be from 1 to 60 bytes.

This parameter must be present if positional parameter 5 was specified.

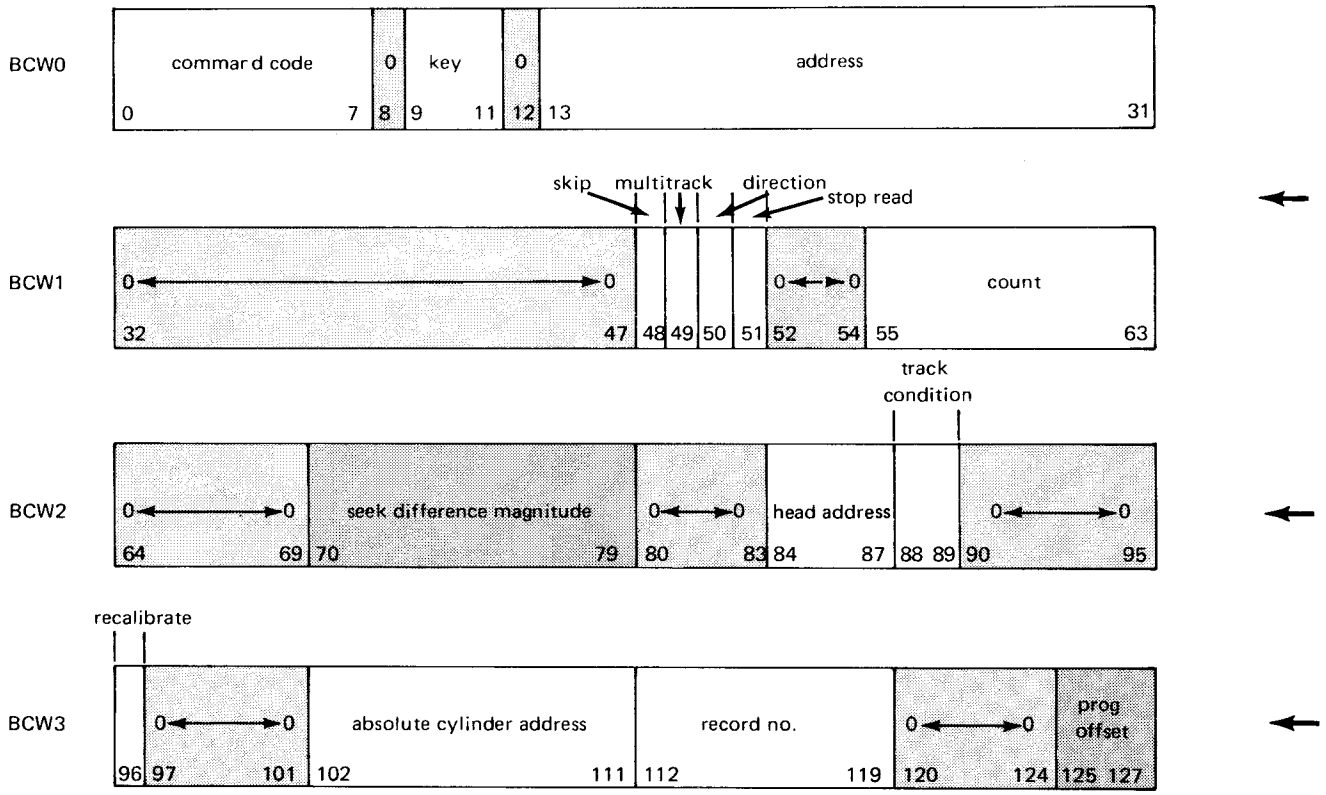
$(r)_5$

Specifies that the register designated (other than 0 or 1) has been preloaded with the length of the buffer area specified in positional parameter 5.

If positional parameter 6 is omitted and positional parameter 5 was specified, the macro instruction does not execute.

Appendix A. Supervisor Control Block Formats

In the figures in this appendix (Figure A—1 through A—10), shaded fields indicate system-supplied data; fields that are not shaded indicate data supplied by the user via the macro instruction that directs the supervisor to generate the control block.



Bits	Allocation	Function
0–7	Command code	Command code to be executed by IDA; bits 0–3 must be zero
8		Unassigned; must be set to zero
9–11	Key	3-bit field containing storage protection key
12		Unassigned; must be set to zero
13–31	Address	Storage address on which command operates

Figure A—1. Buffer Control Word (BCW) Format for Integrated Disc Adapter (Part 1 of 3)

Bits	Allocation	Function
32–47		Unassigned; must be set to zero
48	Skip sentinel	Set with read data command to indicate data transfers inhibited to main storage; set with search/read commands to indicate search begins at index
49	Multitrack sentinel	Set to 1 with search/read command to indicate search limited to cylinder boundaries rather than single track
50	Direction sentinel	If 1, specifies accessor moves in direction of decreasing cylinder numbers
51	Stop read	Stop read command on record which causes record
52–54		Unassigned; must be set to zero
55–63	Count	On search/read commands – number of bytes to be searched On data read or write commands – number of records to be processed
64–69		Unassigned; must be zero
70–79	Seek difference magnitude	During seek operation, specifies magnitude of difference between accessor present position and desired position
80–83		Unassigned; must be set to zero
84–87	Head address	4-bit field specifying current operation head address
88,89	Track condition	Condition of track where operation acts
90–95		Unassigned; must be set to zero

Figure A—1. Buffer Control Word (BCW) Format for Integrated Disc Adapter (Part 2 of 3)

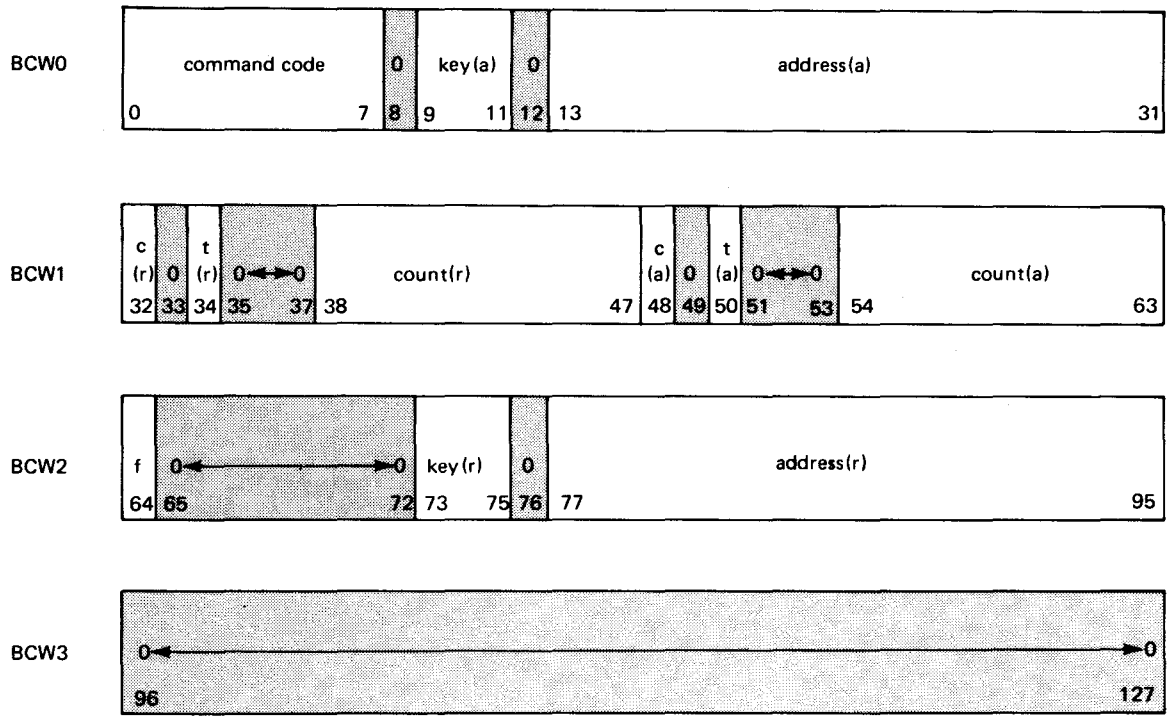
Bits	Allocation	Function	
96	Recalibrate	Set to 1 — accessor reoriented and moved to cylinder 0; overrides bits 71—79 and 50	
97—101		Unassigned; must be set to zero	
102—111	Absolute cylinder address	Final position of accessor after completed seek or recalibrate	
112—119	Record number	Number of record where operation is performed or initiated	
120—124		Unassigned; must be set to zero	
125—127	Programmer offset	Bit 125 = 1	Programmed offset used for command
		Bit 125 = 0	Programmed offset not used; bits 126 and 127 ignored
		Bit 126 = 1	Major offset
		Bit 126 = 0	Minor offset
		Bit 127 = 1	Offset away from hub
		Bit 127 = 0	Offset toward hub

LEGEND:

System-supplied data

Data supplied by the user via the macro instruction that directs the supervisor to generate the control block

Figure A—1. Buffer Control Word (BCW) Format for Integrated Disc Adapter (Part 3 of 3)



Bits	Allocation	Function
0–7	Command code	Field accessed by IPC during SIO instruction
8		Unassigned; must be set to zero
9–11	Key (a)	3-bit field containing I/O storage protection key
12		Unassigned; must be set to zero
13–31	Address (a)	Allows IPC to reference any byte in main storage during data transfer sequences Bit 31 = 0 — Most significant byte of addressed half word Bit 31 = 1 — Least significant byte of addressed half word

Figure A—2. Buffer Control Word (BCW) Format for Integrated Peripheral Channel (Part 1 of 3)

Bits	Allocation	Function
32	c (a)	Specifies data chaining operations when set to 1
33		Unassigned; must be set to zero
34	t (a)	Single control bit used with c(a) bit: c(a) = 0 and t = 0 — Use a fields for current data transfer sequence (no data chaining) c(a) = 0 and t = 1 — Terminates control c(a) = 1 and t = 0 — Use a fields for current data transfer sequence (data chaining initial a and r setting) c(a) = 1 and t = 1 — a fields depleted; replacement operation required t(a) and c(a) = 1: f = 0 — Terminates with buffer wraparound error f = 1, c(r) = 1 or 0, t(r) = 1 — Terminates normally f = 1, c(r) = 0, t(r) = 0 — Normal data transfer; no chaining f = 1, c(r) =1, t(r) = 0 — Normal data transfer with chaining
35–37		Unassigned; must be set to zero
38–47	Count (r)	Byte count required for all data transfer operations
48	c (a)	Specifies data chaining operations when set to 1
49		Unassigned; must be set to zero
50	t(a)	Same as for bit 34
51–53		Unassigned; must be set to zero
54–63	Count (a)	Byte count required for all data transfer operations

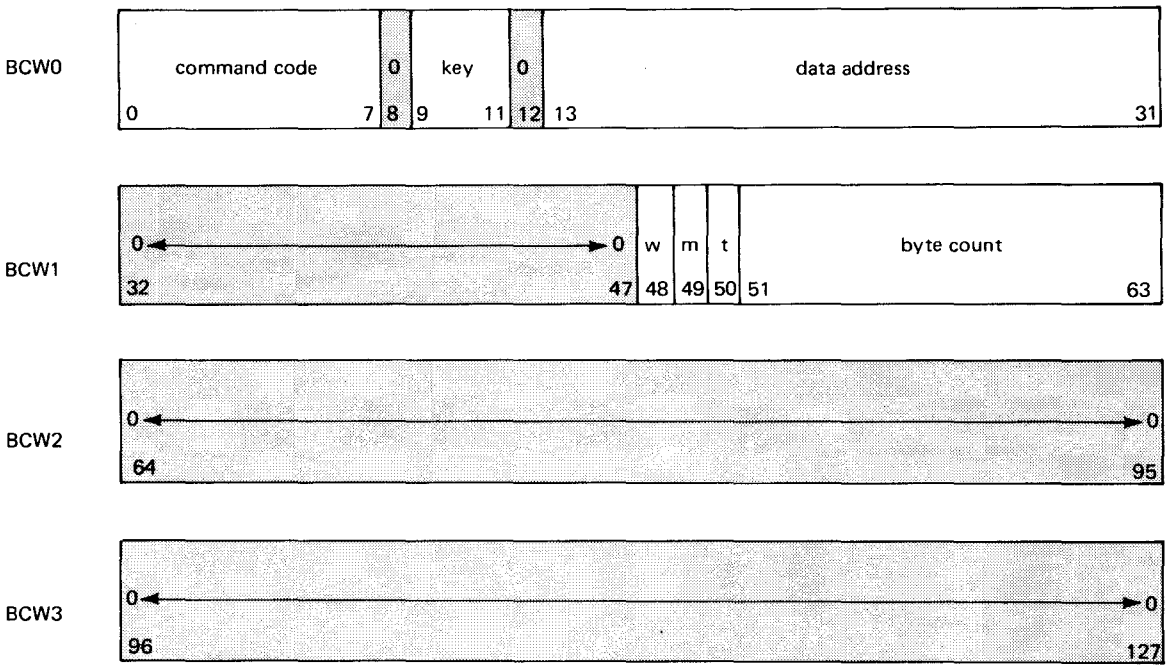
Figure A—2. Buffer Control Word (BCW) Format for Integrated Peripheral Channel (Part 2 of 3)

Bits	Allocation	Function
64	f (flag bit)	Indicates to IPC that current contents of r fields are valid for replacement operation
65–72		Unassigned; must be set to zero
73–75	Key (r)	3-bit field containing I/O storage protection key
76		Unassigned; must be set to zero
77–95	Address (r)	Allows IPC to reference any byte in main storage during data transfer sequences Bits 31 and 95 = 0 — Most significant byte of addressed half word Bits 31 and 95 = 1 — Least significant byte of addressed half word
96–127		Unassigned; must be set to zero

LEGEND:

▒ = System-supplied data
□ = Data supplied by the user via the macro instruction that directs the supervisor to generate the control block
a = active
c = chaining
f = flag
r = replacement
t = transfer

Figure A—2. Buffer Control Word (BCW) Format for Integrated Peripheral Channel (Part 3 of 3)

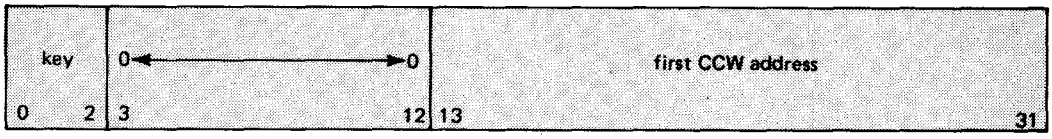


Bits	Allocation	Function
0–7	Command code	Specifies operation to be performed by device and channel
8		Unassigned; must be set to zero
9–11	Key	Contains I/O storage protection key
12		Unassigned; must be set to zero
13–31	Data address	Allows multiplexer channel to reference any byte in main storage during data transfer sequences
32–47		Unassigned; must be set to zero
48	w	w = 0 Input operation (read) w = 1 Output operation (write)
49	m	m = 0 Ascending address (forward sequence) m = 1 Descending address (reverse sequence)
50	t	t = 0 Transfer data t = 1 Termination of data transfer
51–63	Byte count	Contains byte count required for all data transfers
64–127		Unassigned; must be set to zero

LEGEND:

System-supplied data
Data supplied by the user via the macro instruction that directs the supervisor to generate the control block

Figure A—3. Buffer Control Word (BCW) Format for Multiplexer Channel



Bits	Allocation	Function
0–2	Key	I/O storage protection key used by channel for all storage accesses of data and CCWs
3–12		Bits set to zero
13–31	First CCW address	Controls I/O operation initiated by SIO instruction

LEGEND:

System-supplied data

Figure A—4. Channel Address Word (CAW) Format

Byte	0	1	2	3
0	control byte 1	I/O error count	transmission byte	control byte 2
4	TCB address① or next CCW address			
8	CCB link	address② or residual CCW byte count		
12	CCW address			
16	PIOCB pointer (PUB address)			
20	sense byte 0	sense byte 1	sense byte 2	sense byte 3
24	sense byte 4	sense byte 5	device status	channel status

NOTES:

① During the I/O command execution, contains the address of the TCB associated with this CCB. At I/O command termination, PIOCS inserts the address of the next CCW in the chain.

② During I/O command execution, bytes 8 through 11 contain the address of the next CCB in the chain at this job level. At I/O command termination, PIOCS inserts the number of bytes remaining in the CCW byte count (when the I/O command terminated) into bytes 10 and 11.

Figure A—5. Command Control Block (CCB) Format (Part 1 of 2)

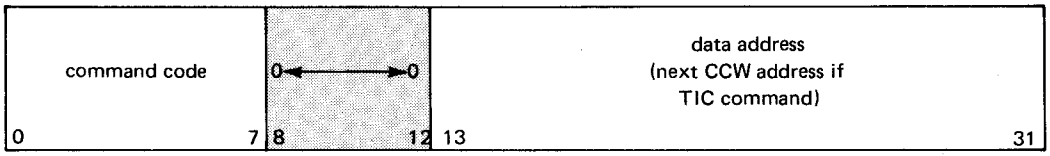
Byte	Length	Content
0	1	Control byte 1 Bits 0–2 Reserved 3 1 = Ignore block numbers 4 Reserved 5 1 = CCB in wait condition 6–7 Reserved
1	1	Binary count of errors encountered processing the CCB
2	1	Transmission byte Bit 0 0 = CCB in process 1 = CCB processed 1 1 = Unrecoverable error 2 1 = Unique error 3 1 = No record found 4 1 = Unit exception 5 1 = Block numbers not equal 6 1 = Track end 7 1 = Cylinder end
3	1	Control byte 2 Bit 0 1 = User error recovery 1 1 = Accept unrecoverable errors 2 1 = Accept unique errors 3 1 = Diagnostic CCB 4 1 = System access CCB 5 Reserved 6 Reserved 7 1 = Block number area reserved
4–7	4	During I/O command execution, full-word address of TCB associated with this CCB or At I/O command termination, full-word address of next CCW (if not at end of command chain)
8–11	4	During I/O command execution, full-word address of next CCB or
10–11	2	At I/O command termination, bytes remaining in CCW byte count when I/O command was terminated
12–15	4	Full-word address of first CCW or BCW
16–19	4	Address of PIOCB entry which contains the half-word address of PUB associated with this CCB
20–23	4	Sense bytes 0 through 3
24, 25	2	Sense bytes 4 and 5
26	1	Device status Bit 0 1 = Attention 1 1 = Status modifer 2 1 = Control unit end 3 1 = Busy 4 1 = Channel end 5 1 = Device end 6 1 = Unit check 7 1 = Unit exception
27	1	Channel status Bit 0 0 1 1 = Incorrect length 2 1 = Program check 3 1 = Invalid address 4 1 = Channel data check 5 1 = Interface control check 6 1 = Channel control check 7 1 = Buffer terminate

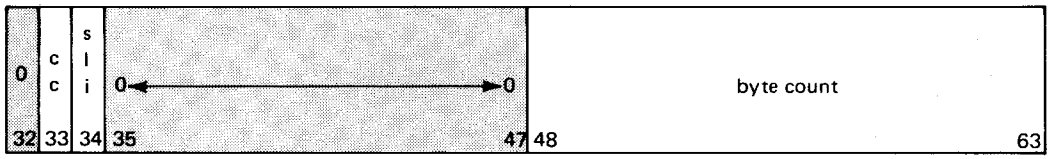
LEGEND:

System-supplied data

Data supplied by the user via the macro instruction that directs the supervisor to generate the control block

Figure A—5. Command Control Block (CCB) Format (Part 2 of 2)





Bits	Allocation	Function
0–7	Command code	Specifies operation to be performed by device and channel
8–12		Unassigned; must be set to zero
13–31	Data address	Address of location in main storage into or from which first byte of data is transferred
32		Unassigned; must be set to zero
33	cc (chain command flag)	When valid ending device status received, new CCW fetched and operation specified by new command code initiated
34	sli (suppress length indication flag)	If set to 1, incorrect length condition not indicated to program; if cc = 1 also, command chaining not suppressed
35–47		Unassigned; must be set to zero
48–63	Byte count	Byte count required for all data transfer operations

LEGEND:

System-supplied data
Data supplied by the user via the macro instruction that directs the supervisor to generate the control block

Figure A—6. Channel Command Word (CCW) Format for Selector Channel

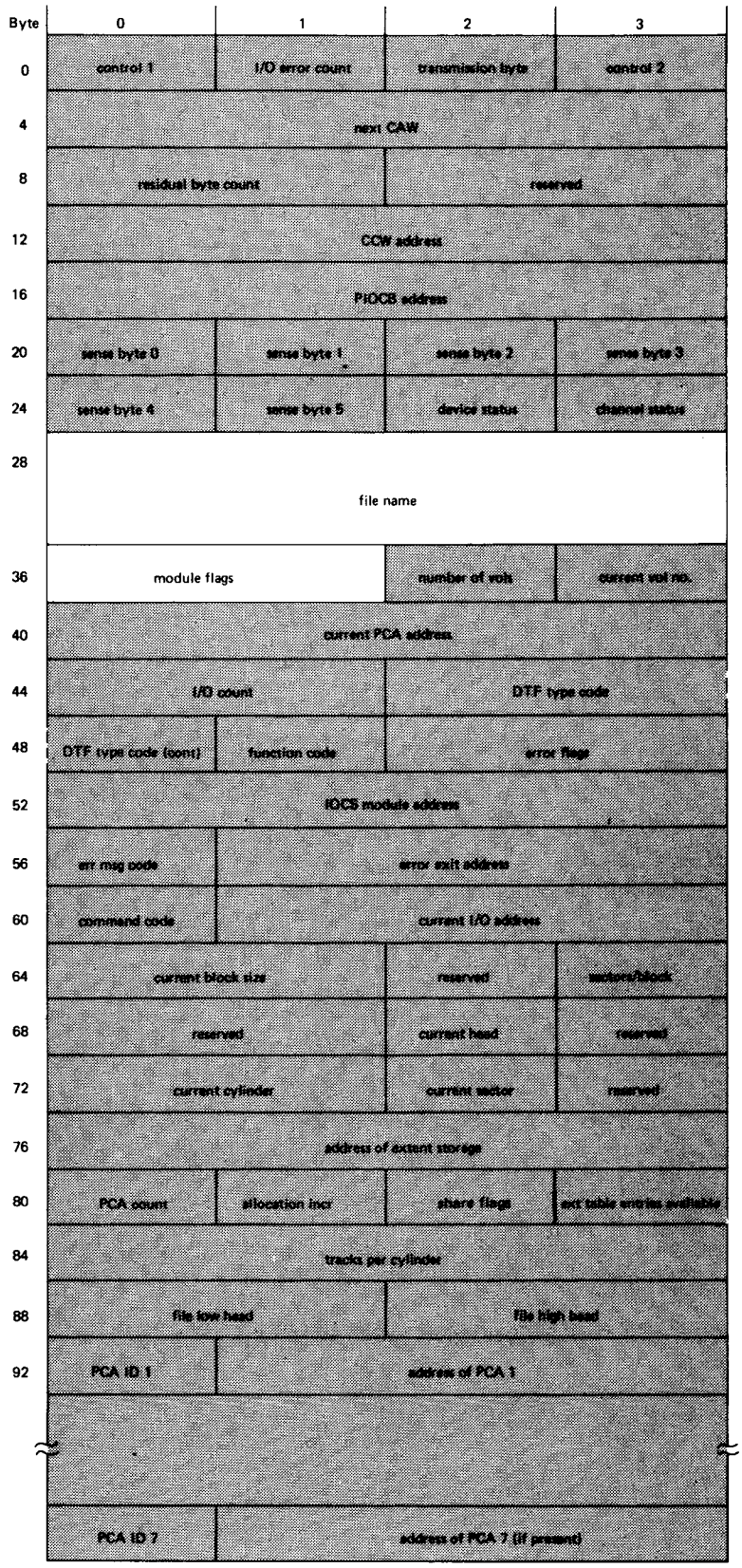


Figure A—7. Define the File (DTF) Table Format (Part 1 of 2)

MODULE FLAGS

Byte	Content	
1	Bit 0	Open
	1	Wait required
	2	WAIT = YES
	3	Sector type disc
	4	F2 active
	5	No extension made
	6	FCB not found
	7	Multiple I/O permitted
2	Bit 0	Search wait required
	1	Cylinder alignment
	2	Format entered by extend
	3	Reserved
	4	Library lock required
	5	FCB in core
	6	Single mount
	7	Unassigned space available

ERROR FLAGS

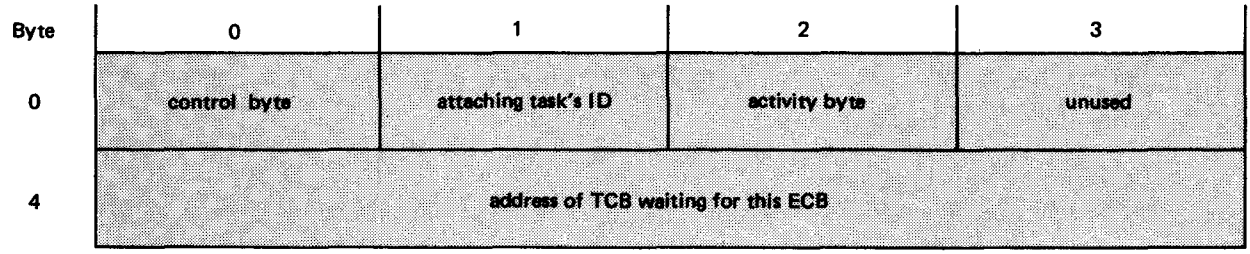
Byte	Content	
1	Bit 0	Access to last record on track
	1	Invalid ID
	2	Invalid PCA
	3	Hardware error
	4	Reserved
	5	Reserved
	6	Reserved
	7	Reserved
2	Bit 0	I/O complete
	1	Unrecoverable error
	2	Unique unit error
	3	No record found
	4	Unit exception
	5	Reserved
	6	End of track
	7	End of cylinder

LEGEND:

System-supplied data

Data supplied by the user via the macro instruction that directs the supervisor to generate the control block

Figure A—7. Define the File (DTF) Table Format (Part 2 of 2)



Byte	Content
0	Control byte Bit 0 1 = This is an ECB. 1–4 Not used 5 1 = This completion is being awaited. 6–7 Not used
1	Attaching task's ID Task identification number of task with which this ECB is associated. This ID number is not related to subtask name. It is the number of the TCB, counting from the job step TCB, which is number 0.
2	Activity byte Bit 0 0 = Task is active in that it has not executed either a TYIELD or DETACH macro. 1 = Task is idle in that it has executed a TYIELD or DETACH macro. 1–6 Not used 7 1 = Task has abnormally terminated and should be detached.
3	Unused
4–7	Address of TCB that is awaiting completion of the task with which this ECB is associated.

LEGEND:

System-supplied data

Figure A—8. Event Control Block (ECB) Format

Byte	0	1	2	3
0	current ID			
4	max relative block			
8	logical blocks/track			
12	PCA ID	EOD ID		
16	I/O count	IOAREA/address		
20	block size		reserved	sectors/block
24	lace factor/key length		unit of store	
28	DTF address			
32	PCA flags	EOD address		

PCA FLAGS

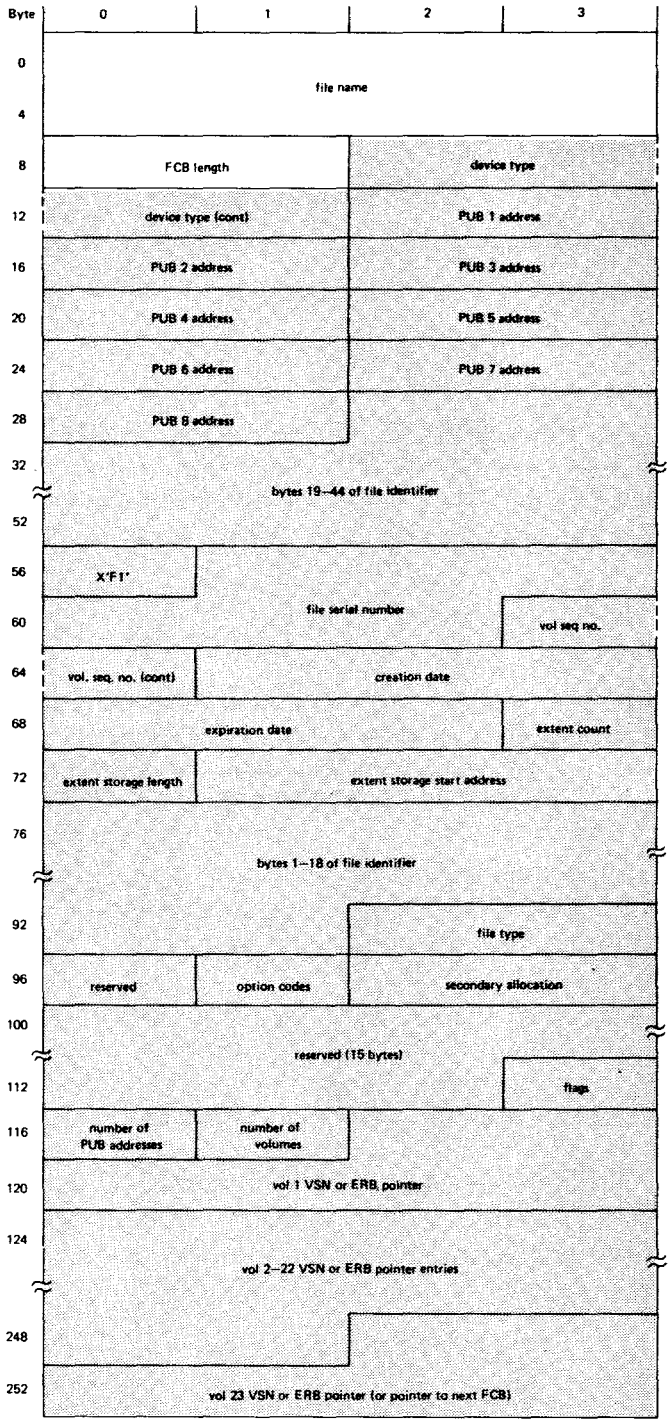
Bit	
0	Format write
1	Interlace
2	SEQ = YES
3	WRITE VERIFY
4	Verify required/initial allocation
5	No extension permitted
6	Interlace adjust/keyed data
7	LBLK specified

LEGEND:

System-supplied data

Data supplied by the user via the macro instruction that directs the supervisor to generate the control block

Figure A—9. Partition Control Appendage (PCA) Format



LEGEND:

System-supplied data

Data supplied by the user via the macro instruction that directs the supervisor to generate the control block

NOTES:

1. Physical I/O control block (PIOCB) refers to the buffer area generated by the PIOCB macro instruction. File control block (FCB) refers to the file control data read into the buffer area by the RDFCB macro instruction. Minimum PIOCB includes bytes 0 to 15; maximum PIOCB includes bytes 0 to 255.

2. Bytes 118 to 255 consist of a 6-byte field for each volume which contains either a 6-byte volume serial number (VSN) or a 4-byte extent request block (ERB) pointer if the volume was not allocated at the time the FCB was built. If there are more than 23 volumes, the last field contains a pointer to the next FCB.

Figure A—10. Physical I/O Control Block (PIOCB) and File Control Block (FCB) Format

Word	Byte	Content
1	0	(reserved for system use)
2	4	SAVE AREA BACKWARD LINK ADDRESS
3	8	SAVE AREA FORWARD LINK ADDRESS
4	12	CALLING PROGRAM RETURN ADDRESS
5	16	CALLED PROGRAM ENTRY POINT ADDRESS
6	20	REGISTER 0
7	24	REGISTER 1
8	28	REGISTER 2
9	32	REGISTER 3
10	36	REGISTER 4
11	40	REGISTER 5
12	44	REGISTER 6
13	48	REGISTER 7
14	52	REGISTER 8
15	56	REGISTER 9
16	60	REGISTER 10
17	64	REGISTER 11
18	68	REGISTER 12

NOTE:

Each word in the save area is aligned on a full-word boundary.

Figure A–11. Register Save Area Format (Part 1 of 2)

Word	Content
1	Reserved for system use
2	If zero, indicates the first save area of a chain. Otherwise, this is the address of the save area used by the calling program which is located in the higher level program that called the calling program. For example, bytes 4–7 of SAVE B (a save area in program B for the use of program C) contains the address of SAVE A (a save area in program A for the use of program B). It is the responsibility of the calling program to store the backward link address in this field from register 13 before loading the current save area address in register 13.
3	If zero, indicates the last save area in a chain. Otherwise, this is the address of the save area in the most recently called program. It is the responsibility of this called program to store the save area address in this field before calling a lower level program.
4	The address in the calling program (as loaded in register 14) to which control is to be returned. This address must be stored in this field by the called program if that program intends to alter the contents of register 14.
5	The entry point address of the called program (as stored in register 15) to which control is to be transferred. This address must be moved to this field by the calling program.
6 to 8	A storage area provided to contain the contents of registers 0 through 12. The called program determines the number of registers, if any, to be saved.

Figure A–11. Register Save Area Format (Part 2 of 2)

Appendix B. Canned Message Buffer Format

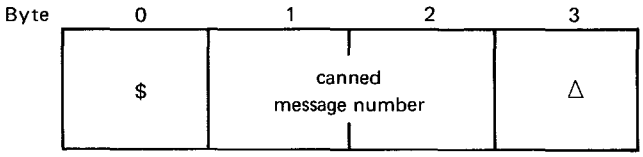
A canned message file is maintained on disc. These canned messages, with or without variables, may be retrieved, logged, or displayed. Canned messages to be displayed are 60 bytes long.

If a canned message is to be retrieved, logged, or displayed, positional parameter 1 of the WTL, WTLD, GETMSG, or OPR macro instruction must be the address of a buffer containing canned message information.

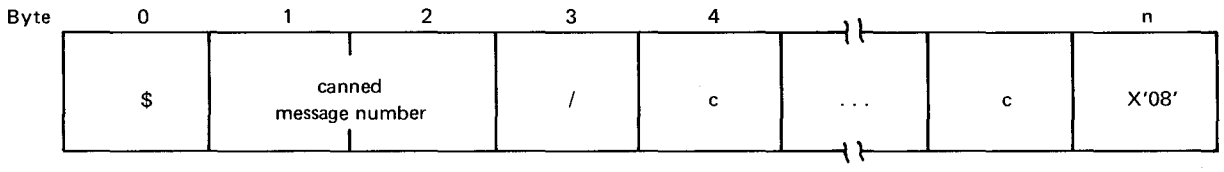
The canned message that will have characters inserted in it is created so that an "underline" (EBCDIC X'6D') represents a byte into which a character is to be inserted. When an insert character list is specified in positional parameter 1 of the WTL, WTLD, GETMSG, or OPR macro instructions, these macro instructions scan the canned message from left to right and, when an underline is found, a character is moved from the character list to the message. The character in the list is moved to the next underline found in the message, etc. This process is continued until either the terminating EBCDIC code X'08' is found in the character list or the length of the canned message has been scanned.

The format for the canned message is shown in Figure B—1.

Buffer format for canned messages without insert characters:



Buffer format for canned messages with insert characters:



$
First character of the buffer; indicates that this buffer pertains to a canned message. If any other type of message must start with a dollar sign, two dollar signs are required at the beginning of the message buffer.

canned message number
The (16 bit) canned message number as a binary value.

Δ (space)
A space in byte 3 indicates there are no characters to be inserted into the canned message.

/
A slash in byte 3 indicates there are characters to be inserted into the canned message.

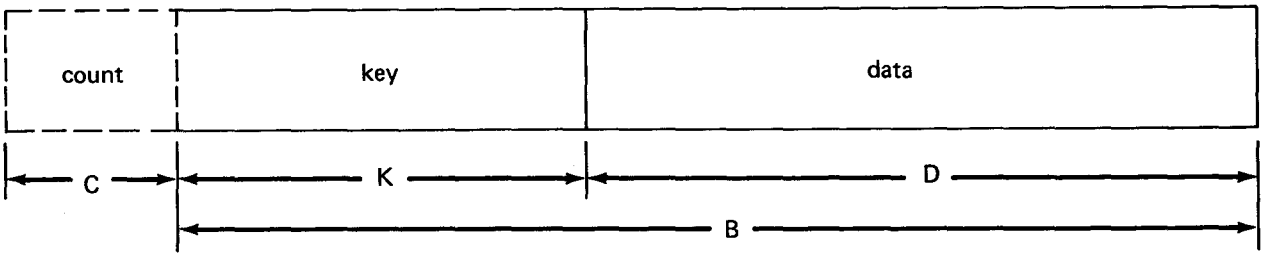
c
If present, the characters to be inserted into the canned message are contained in the buffer starting at byte 4.

X'08'
Hex code '08' used to terminate the character string.

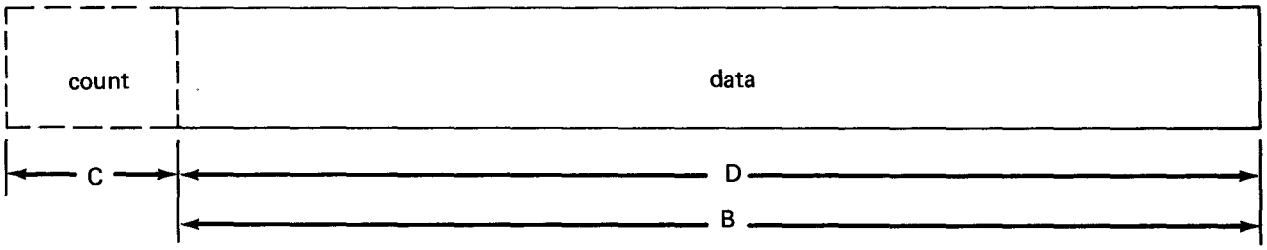
Figure B—1. Canned Messages Buffer Formats

Appendix C. Record Formats for Disc Devices

WITH KEYS



WITHOUT KEYS

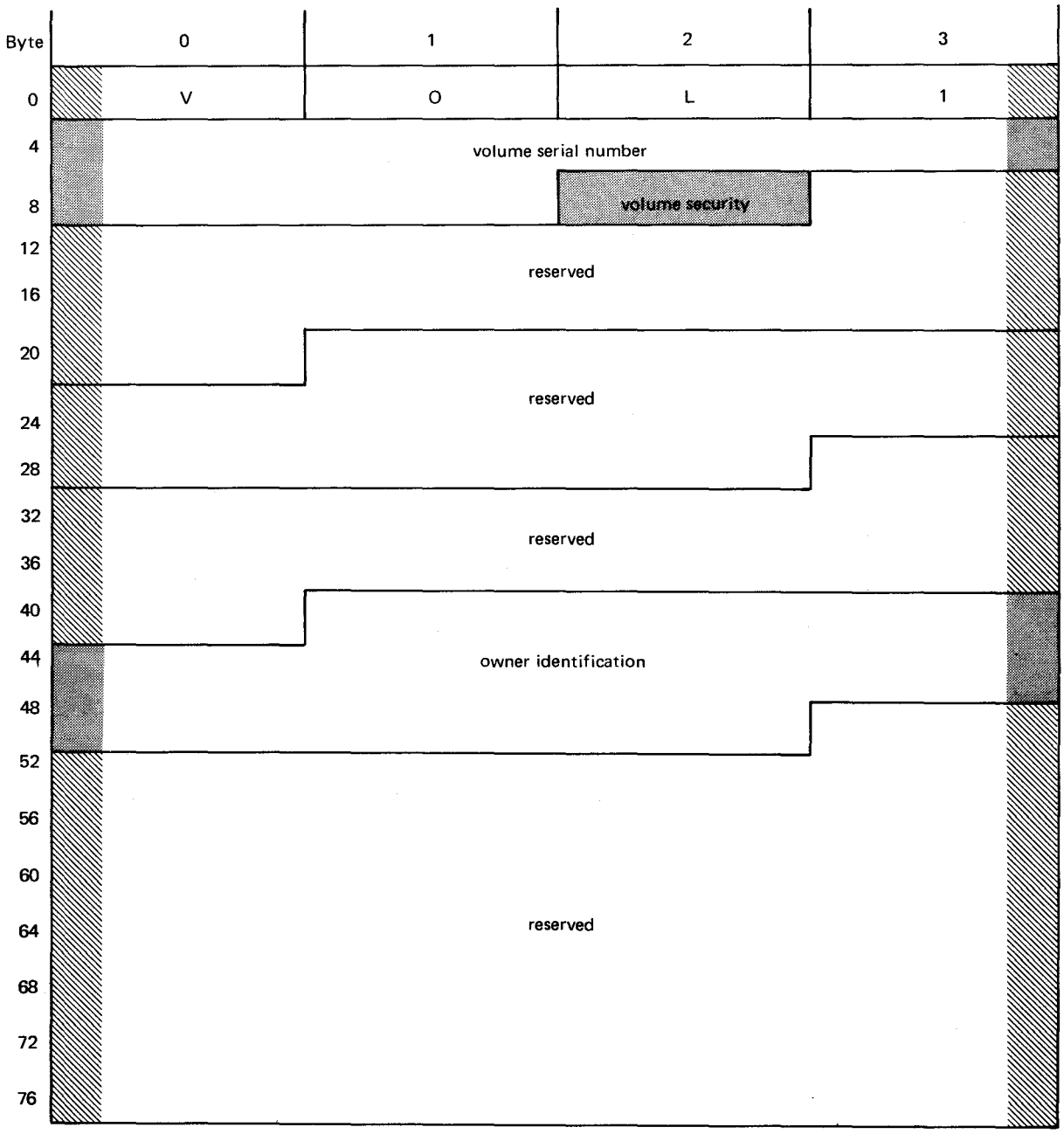


LEGEND:

C = Count field length (8 bytes). Count field is used only by data management.
K = Key field length (3—255 bytes)
D = Data field length
B = Block length (< track length and cannot span track boundaries)

Figure C—1. Record Formats for Disc Devices

Appendix D. System Standard Tape Label Formats



LEGEND:

Generated by TSAT or reserved for system expansion.

Written by TSAT from user-supplied data.

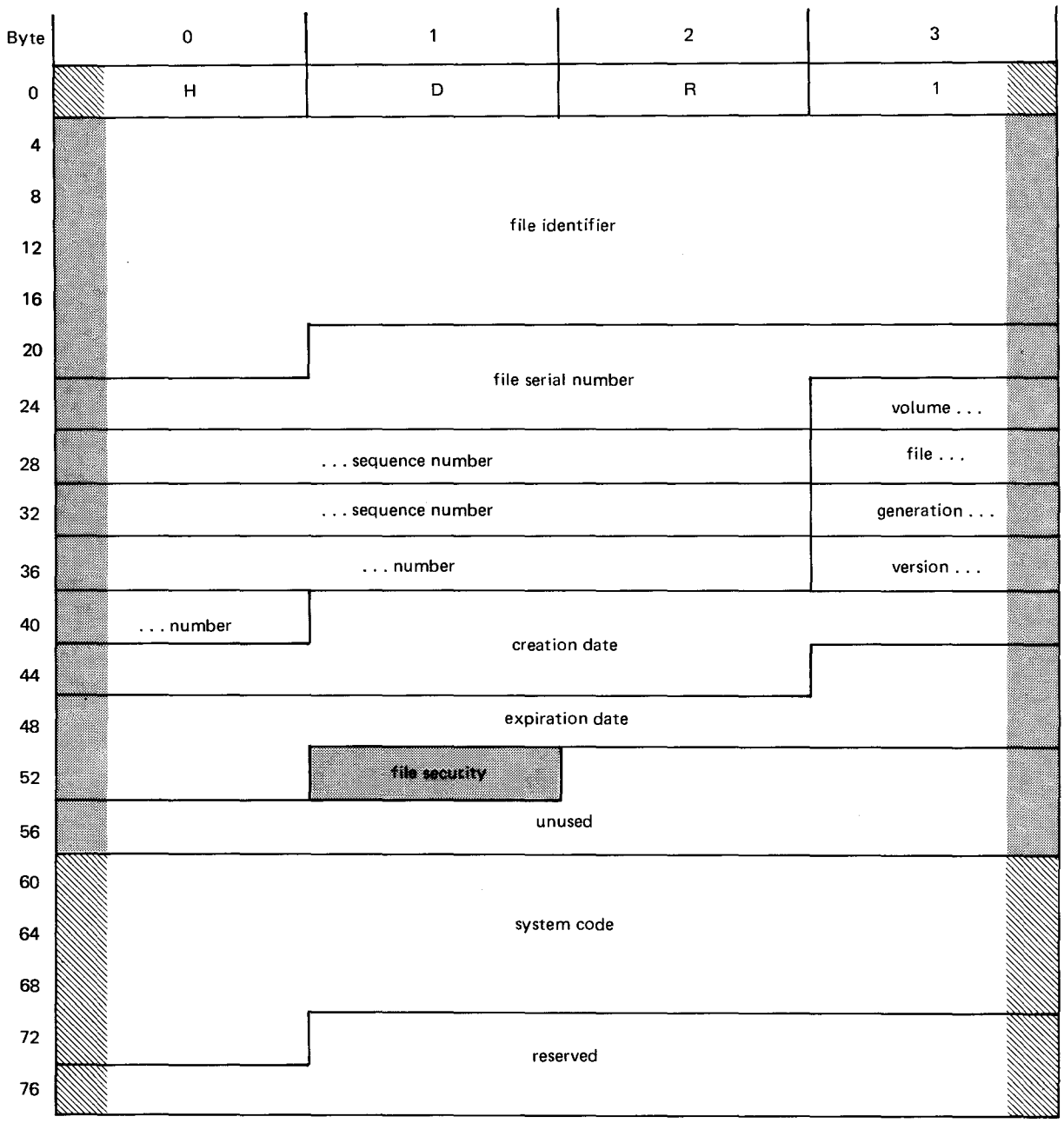
Figure D—1. Tape Volume 1 (VOL1) Label Format for an EBCDIC Volume (Part 1 of 2)

Field	Initialized By	Bytes	Code	Description
Label identifier	Tape prep	0–2	EBCDIC	Contains VOL to indicate that this is a volume label
Label number	Tape prep	3	EBCDIC	Always 1 for the initial volume label
Volume serial number	Tape prep	4–9	EBCDIC	Unique identification number assigned to this volume by your installation. TSAT expects 1 to 6 alphanumeric characters, the first of which is alphabetic.
Volume security	TSAT	10	EBCDIC	Reserved for future use by installations requiring security at the reel level. Currently blank
Reserved	-----	11–20	EBCDIC	Contains blanks (40_{16})
Reserved	-----	21–30	EBCDIC	Contains blanks (40_{16})
Reserved	-----	31–40	EBCDIC	Contains blanks (40_{16})
Owner identification		41–50	EBCDIC	Unique identification of the owner of the reel: any combination of alphanumerics
Reserved	-----	51–79	EBCDIC	Contains blanks (40_{16})

NOTE:

For ASCII files, Bytes 0—36 of a VOL1 label have the same significance as shown in the preceding example. Bytes 37—50 indicate the owner identification field. Bytes 51—78 are blank and are reserved for future standardization. Byte 79 indicates the label standard level, and when set to 1, indicates formats on this volume meet the American National Standard. X3.27—1969.

Figure D—1. Tape Volume 1 (VOL1) Label Format for an EBCDIC Volume (Part 2 of 2)



LEGEND:

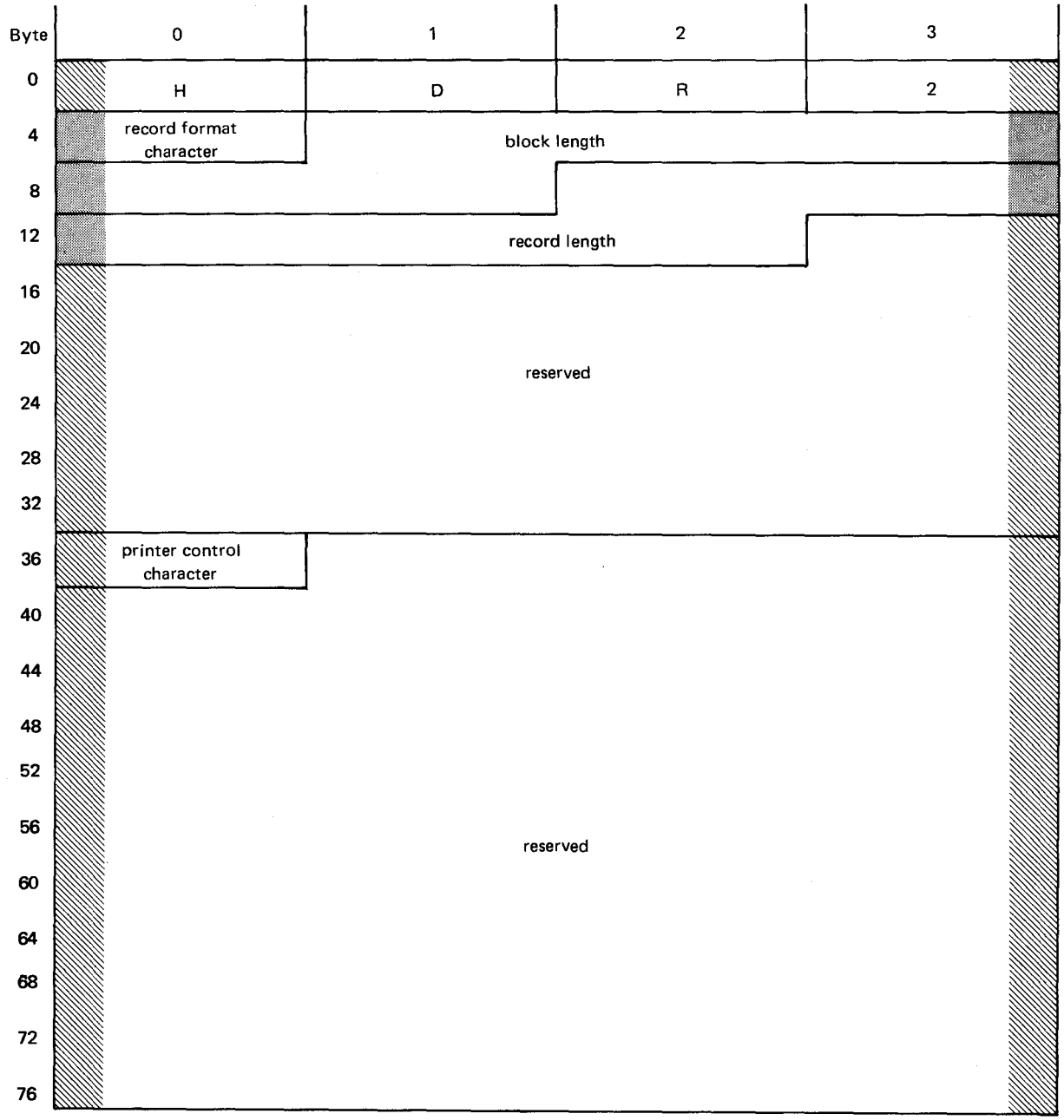
Generated by TSAT or reserved for system expansion.

Written by TSAT from user-supplied data.

Figure D—2. First File Header Label (HDR1) Format for an EBCDIC Tape Volume (Part 1 of 2)

Field	Bytes	Description
Label identifier	0–2	Contains HDR to indicate a file header label
Label number	3	Always 1
File identifier	4–20	A 17-byte configuration that uniquely identifies the file. It may contain embedded blanks and is left-justified in the field if fewer than 17 bytes are specified.
File serial number	21–26	The same as the VSN of the VOL1 label for the first reel of a file or a group of multifile reels
Volume sequence number	27–30	The position of the current reel with respect to the first reel on which the file begins. This number is used with multivolume files only.
File sequence number	31–34	The position of this file with respect to the first file in the group
Generation number	35–38	The generation number of the file (0000–9999)
Version number of generation	39–40	The version number of a particular generation of a file
Creation date	41–46	The date on which the file was created, expressed in the form YYDDD and right-justified. The leftmost position is blank.
Expiration date	47–52	The date the file may be written over or used as scratch, in the same form as the creation date
File security indicator	53	Reserved for file security indicator. Indicates whether additional qualifications must be met before a user program may have access to the file. 0 = No additional qualifications are required. 1 = Additional qualifications are required.
(unused)	54–59	Unused field, containing EBCDIC 0's
System code	60–72	Reserved for system code, the unique identification of the operating system that produced the file
(reserved)	73–79	Reserved field, containing blanks (40_{16}).

Figure D–2. First File Header Label (HDR1) Format for an EBCDIC Tape Volume (Part 2 of 2)



LEGEND:

Generated by TSAT or reserved for system expansion.

Written by TSAT from user-supplied data.

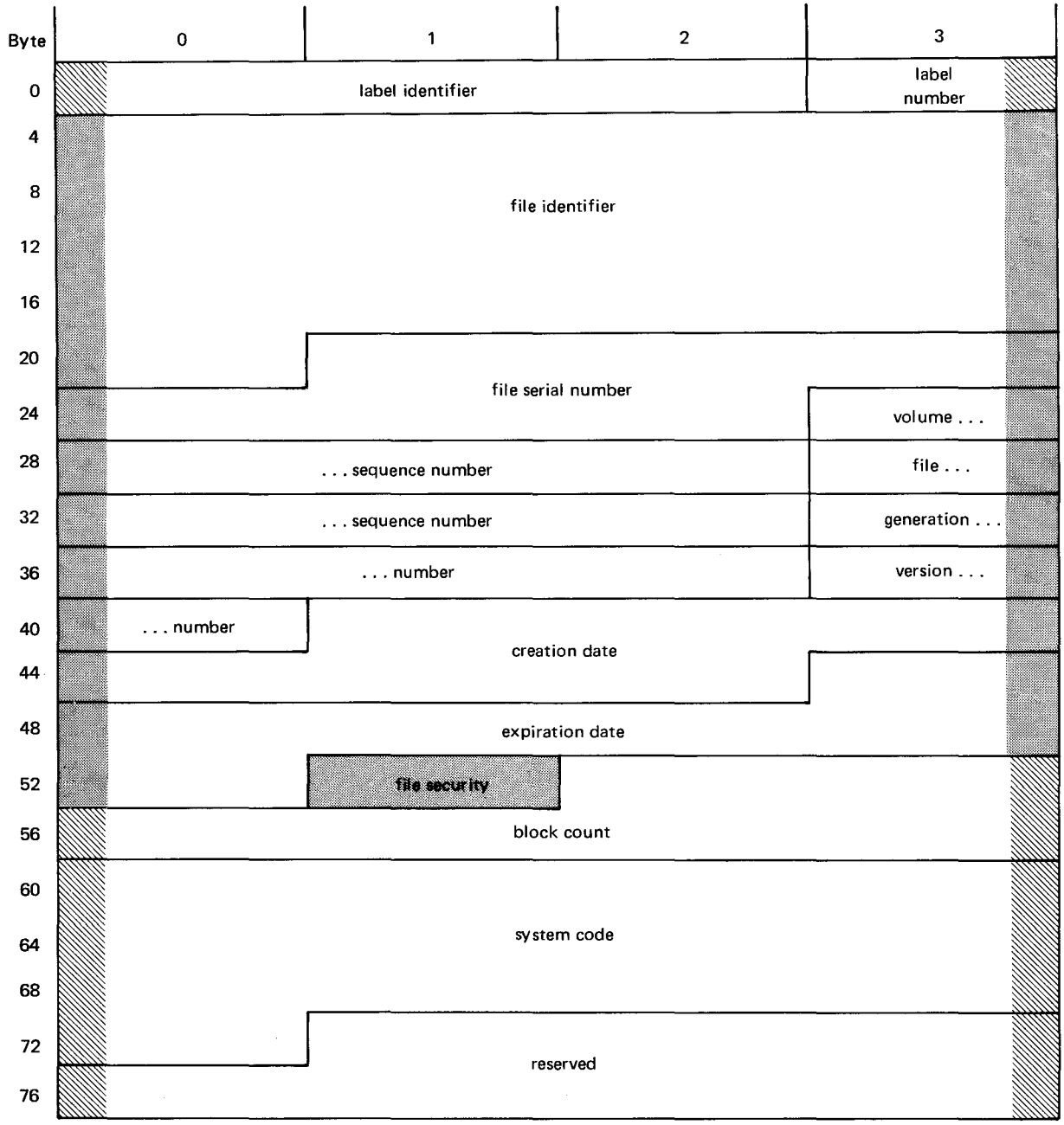
Figure D—3. Second File Header Label (HDR2) Format for an EBCDIC Tape Volume (Part 1 of 2)

Field	Bytes	Description
Label identifier	0–2	Contains HDR to indicate a file header label
Label number	3	Always 2
Record format character	4	Character / Meaning D Variable-length (ASCII), with length fields specified in decimal F Fixed-length S Spanned U Undefined V Variable-length (EBCDIC), with length fields specified in binary
Block length	5–9	Five EBCDIC characters specifying the maximum number of characters per block
Record length	10–14	Five EBCDIC characters specifying the record length for fixed-length records. For any other record format, this field contains 0's.
(reserved)	15–35	Reserved for future system use
Printer control character	36	One EBCDIC character indicating which control character set was used to create the data set. A Special (ASA) control character present D Device independent control character present M IBM control character present U SPERRY UNIVAC control character present
(reserved)	37–79	Reserved for future system use

NOTE:

For ASCII files, bytes 0—14 of a HDR2 label have the same significance as shown in the preceding example. Bytes 50 and 51 indicate the buffer offset field which must be included in the block length. All other fields are recorded as ASCII spaces.

Figure D—3. Second File Header Label (HDR2) Format for an EBCDIC Tape Volume (Part 2 of 2)



LEGEND:

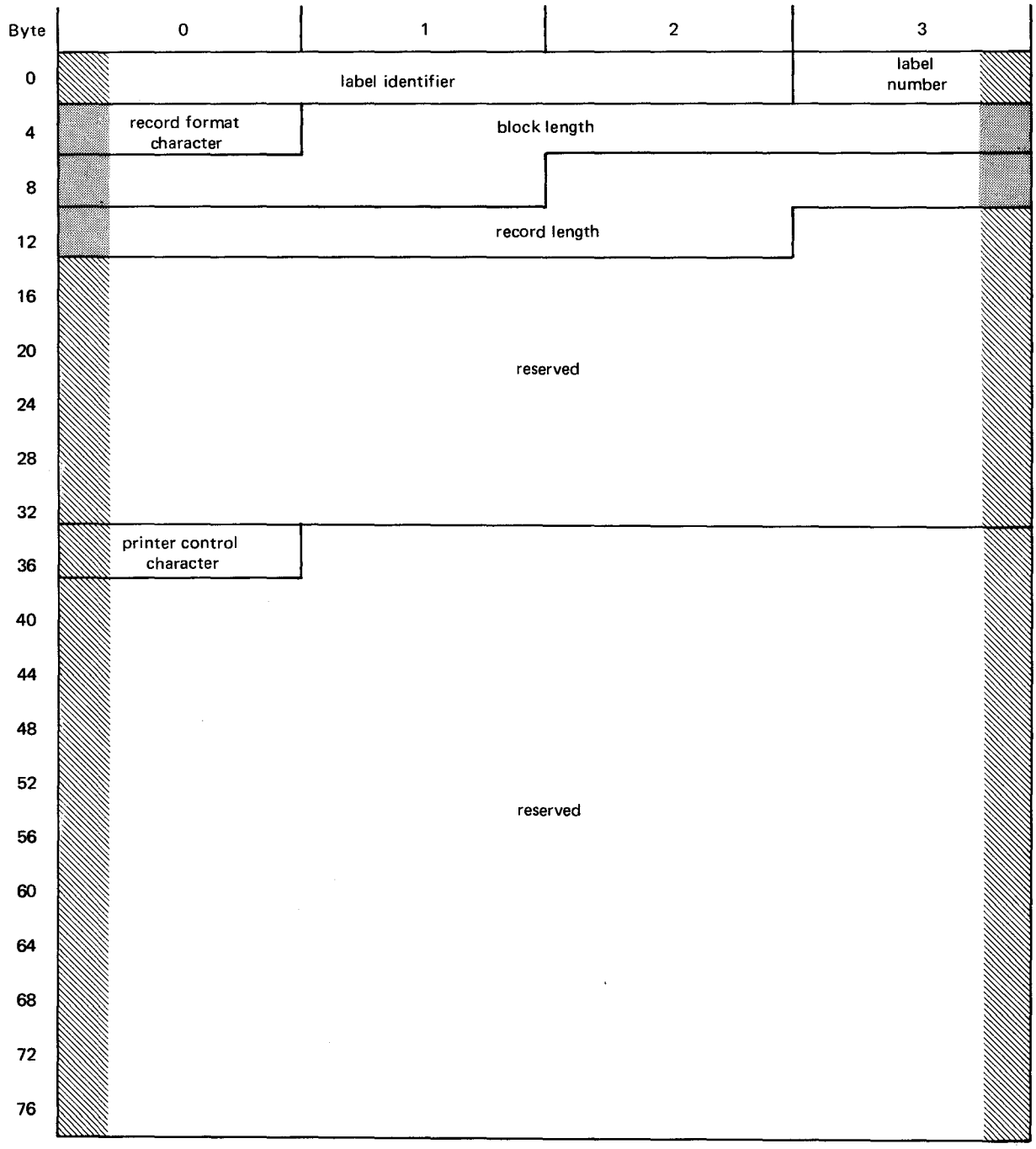
Generated by TSAT or reserved for system expansion.

Written by TSAT from user-supplied data.

Figure D—4. Tape File EOF1 and EOV1 Label Formats (Part 1 of 2)

Field	Bytes	Description
Label identifier	0–2	Indicates that this is a file trailer label; contains EOF for an end-of-file label, or EOV for an end-of-volume label
Label number	3	Always 1
File identifier	4–20	A 17-byte configuration that uniquely identifies the file. It may contain embedded blanks and is left-justified in the field if fewer than 17 bytes are specified.
File serial number	21–26	The same as the VSN of the VOL1 label for the first reel of a file or a group of multifile reels
Volume sequence number	27–30	The position of the current reel with respect to the first reel on which the file begins. This number is used with multivolume files only.
File sequence number	31–34	The position of this file with respect to the first file in the group
Generation number	35–38	The generation number of the file (0000–9999)
Version number of generation	39–40	The version number of a particular generation of a file
Creation date	41–46	The date on which the file was created, expressed in the form YYDDD and right-justified. The left-most position is blank.
Expiration date	47–52	The date the file may be written over or used as scratch, in the same form as the creation date
File security indicator	53	Reserved for file security indicator. Indicates whether additional qualifications must be met before a user program may have access to the file. 0 = No additional qualifications are required. 1 = Additional qualifications are required.
Block count	54–59	In the first file trailer label, indicates the number of data blocks: either in this file of a multifile reel, or on the current reel of a multivolume file. TSAT checks the block count for input files or writes the count for output files.
System code	60–72	Reserved for system code, the unique identification of the operating system that produced the file
(reserved)	73–79	Reserved field, containing blanks (40_{16})

Figure D—4. Tape File EOF1 and EOV1 Label Formats (Part 2 of 2)



LEGEND:

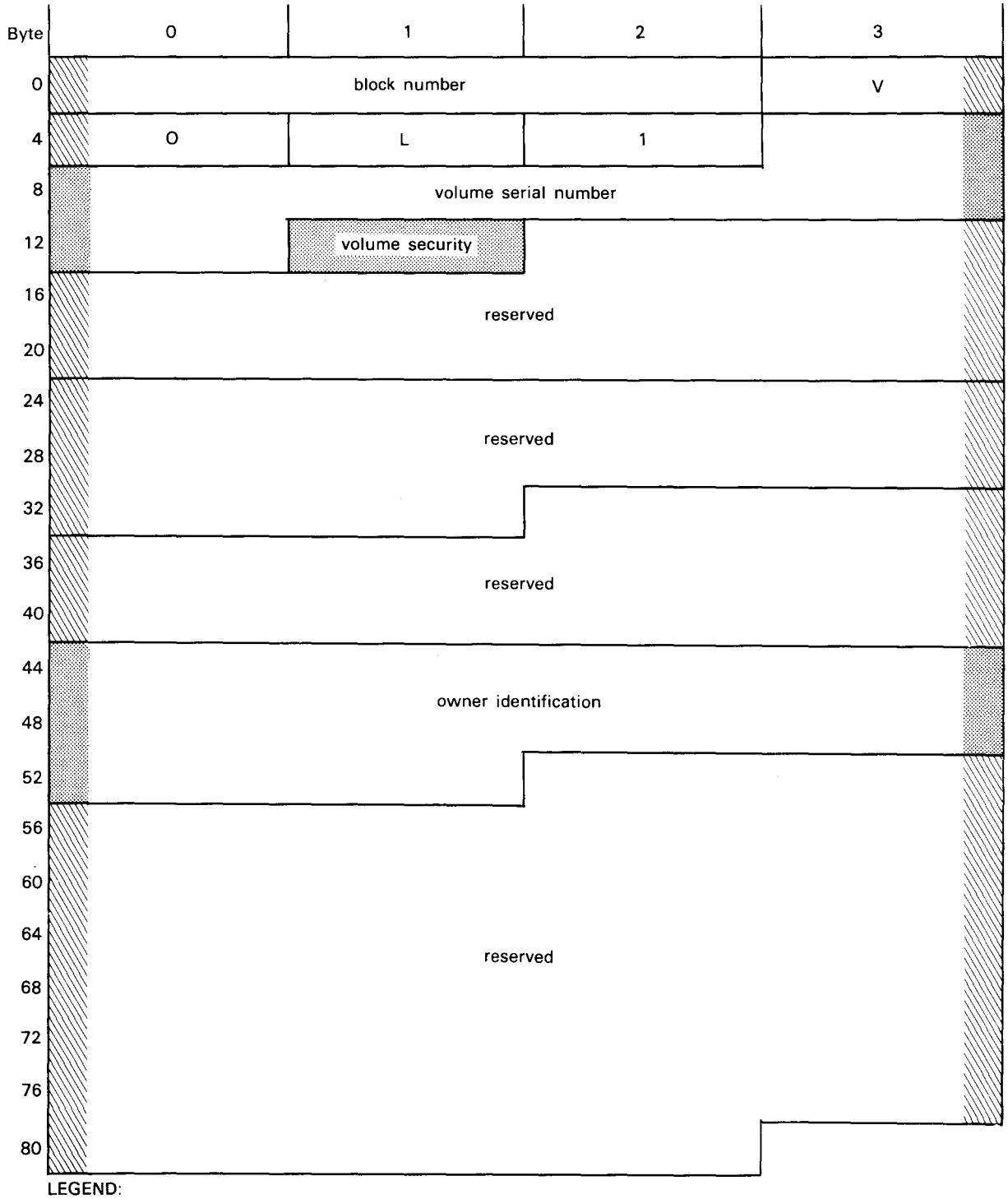
Generated by TSAT or reserved for system expansion.

Written by TSAT from user-supplied data.

Figure D—5. Tape File EOF2 and EOV2 Label Formats (Part 1 of 2)

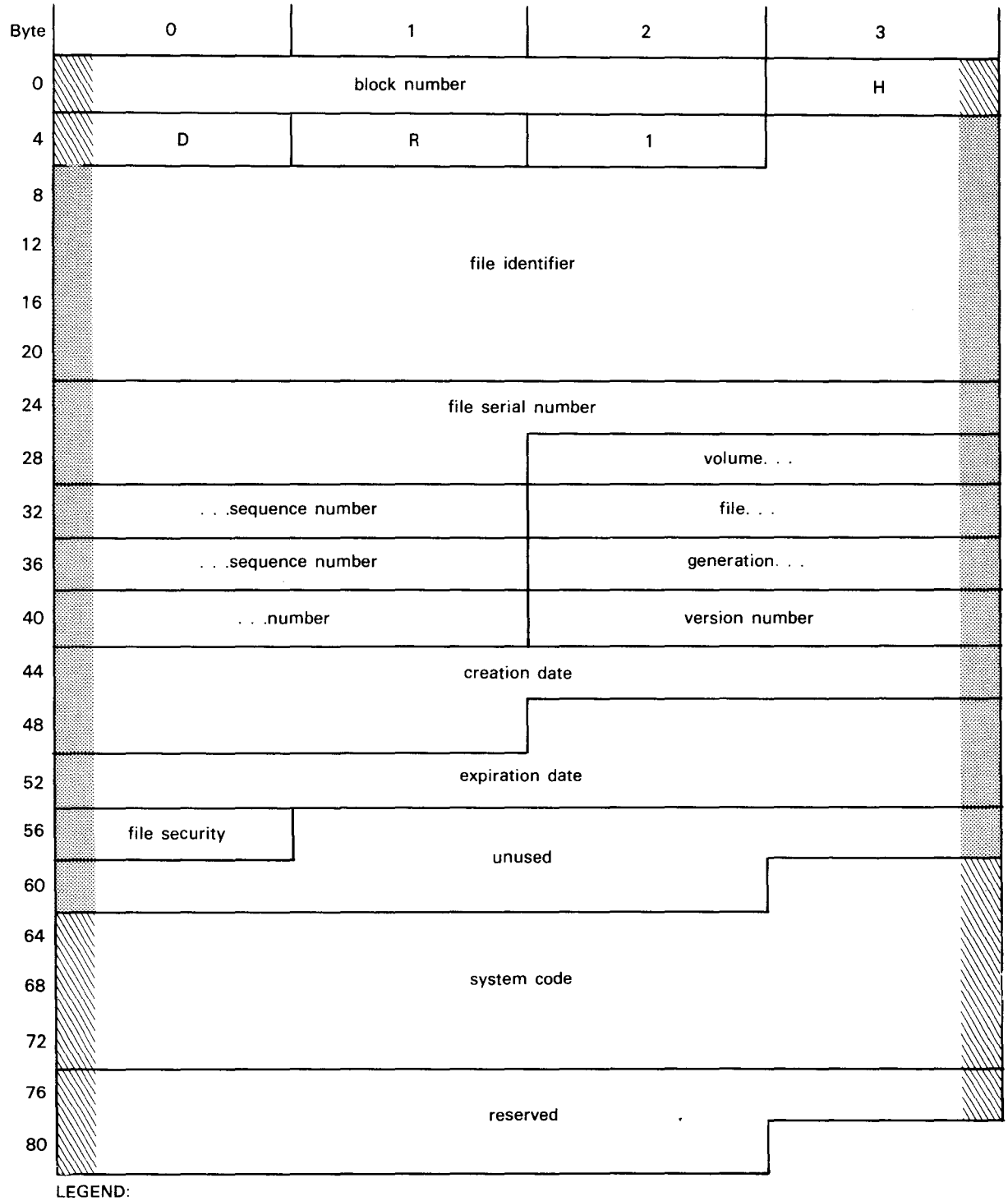
Field	Bytes	Description
Label identifier	0–2	Indicates that this is a file trailer label; contains EOF for an end-of-file label, or EOV for an end-of-volume label
Label number	3	Always 2
Record format character	4	Character — Meaning D — Variable-length (ASCII), with length fields specified in decimal F — Fixed-length S — Spanned U — Undefined V — Variable-length (EBCDIC), with length fields specified in binary
Block length	5–9	Five EBCDIC characters specifying the maximum number of characters per block
Record length	10–14	Five EBCDIC characters specifying the record length for fixed-length records. For any other record format, this field contains 0's.
(reserved)	15–35	Reserved for future system use
Printer control character	36	One EBCDIC character indicating which control character set was used to create the data set. A Special (ASA) control character present D Device independent control character present M IBM control character present U SPERRY UNIVAC control character present
(reserved)	37–79	Reserved for future system use

Figure D—5. Tape File EOF2 and EOV2 Label Formats (Part 2 of 2)



The first three bytes (bytes 0—2) of the tape file label contain a 24-bit block number field. The contents of the remainder of the VOL1 label are the same as described in Figure D—1, except that each field is offset three bytes.

Figure D—6. Tape Volume 1 (VOL1) Label Format for an EBCDIC Volume With Block Numbers



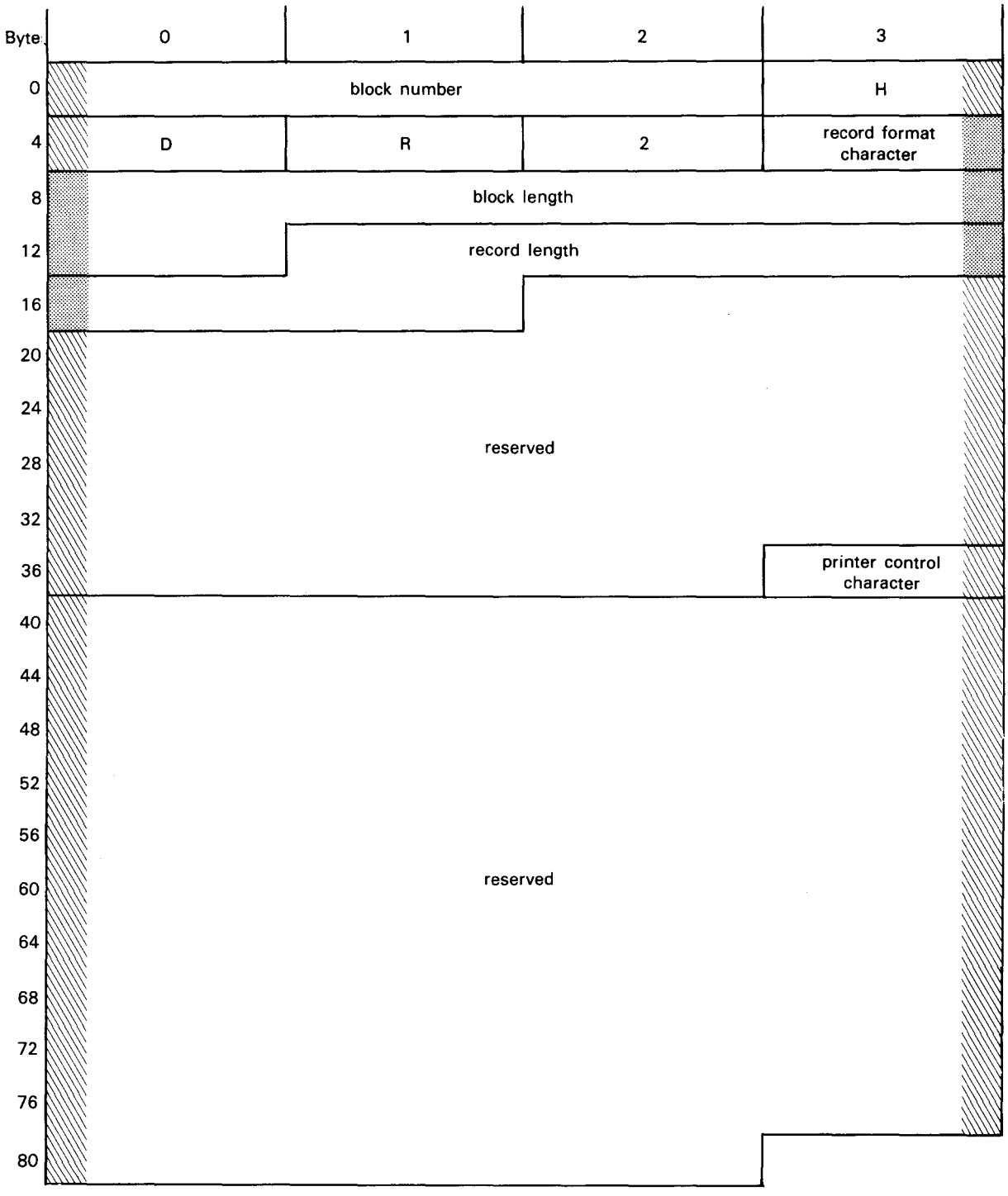
LEGEND:

Generated by TSAT or reserved for system expansion.

Written by TSAT from user-supplied data.

→ The first three bytes (bytes 0—2) of the tape file label contain a 24-bit block number field. The contents of the remainder of the HDR1 label are the same as described in Figure D—2, except that each field is offset three bytes.

Figure D—7. First File Header Label (HDR1) Format for an EBCDIC Tape Volume With Block Numbers



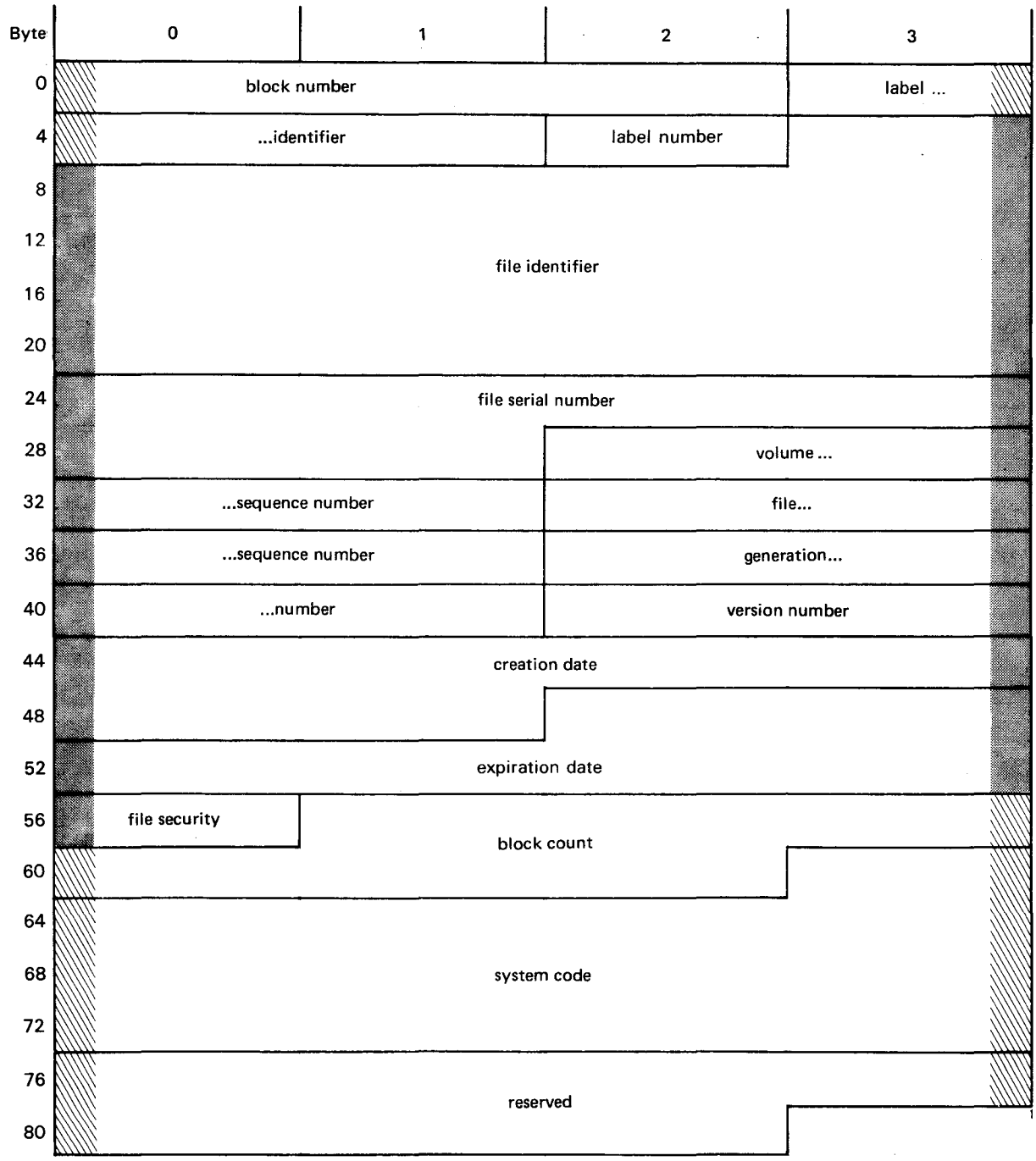
LEGEND:

Generated by TSAT or reserved for system expansion.

Written by TSAT from user-supplied data.

The first three bytes (bytes 0—2) of the tape file label contain a 24-bit block number field. The contents of the remainder of the HDR2 label are the same as described in Figure D—3, except that each field is offset three bytes.

Figure D—8. Second File Header Label (HDR2) Format for an EBCDIC Tape Volume With Block Numbers



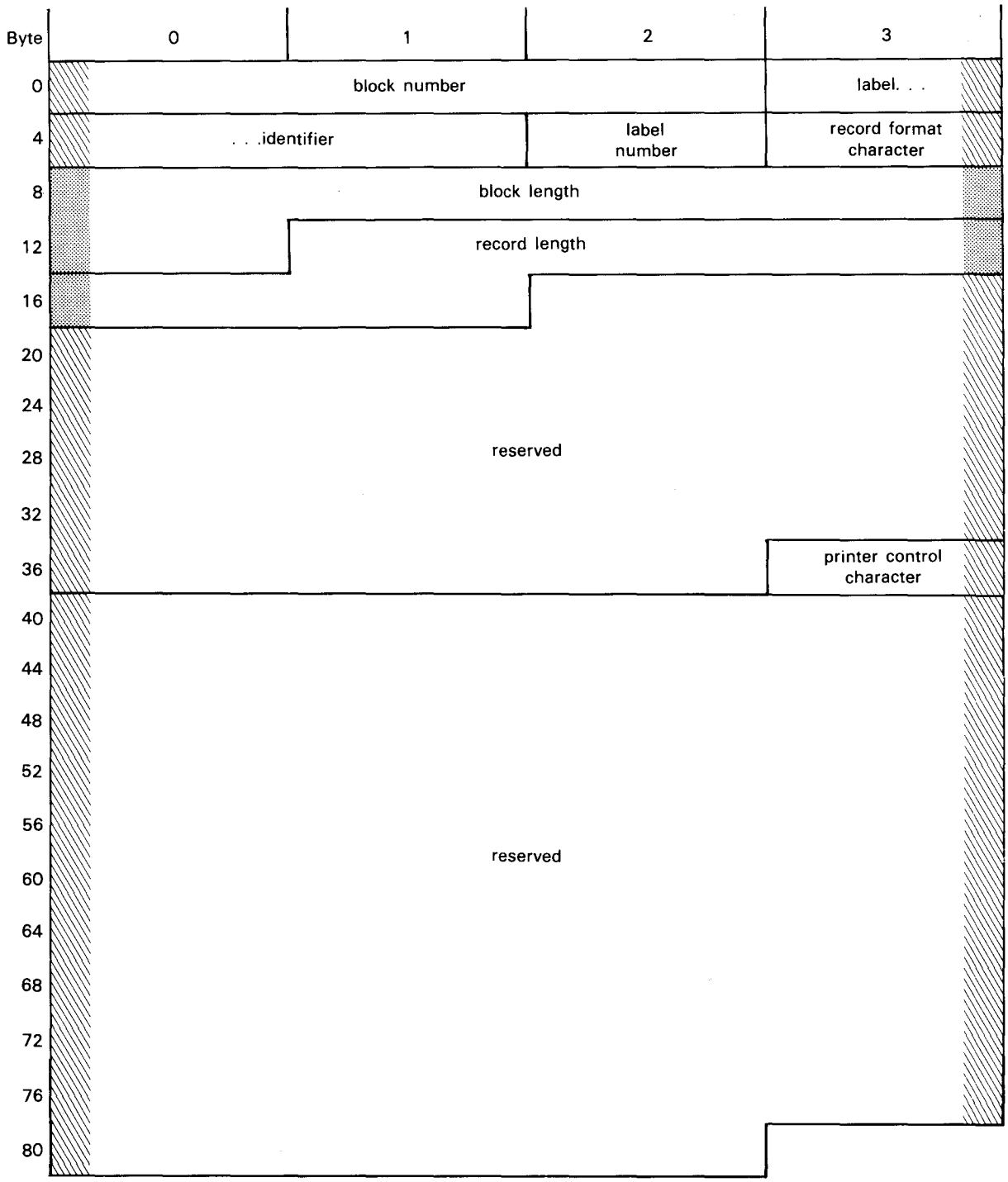
LEGEND:

Generated by TSAT or reserved for system expansion.

Written by TSAT from user-supplied data.

The first three bytes (bytes 0—2) of the tape file label contain a 24-bit block number field. The contents of the remainder of the EOF1 and EOV1 labels are the same as described in Figure D—4, except that each field is offset three bytes.

Figure D—9. Tape File EOF1 and EOV1 Label Formats for Block Numbered Files



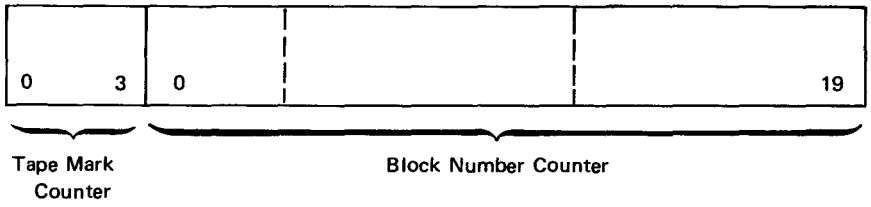
LEGEND:

Generated by TSAT or reserved for system expansion.

Written by TSAT from user-supplied data.

The first three bytes (bytes 0—2) of the tape file label contain a 24-bit block number field. The contents of the remainder of the EOF2 and EOV2 labels are the same as described in Figure D—5, except that each file is offset three bytes.

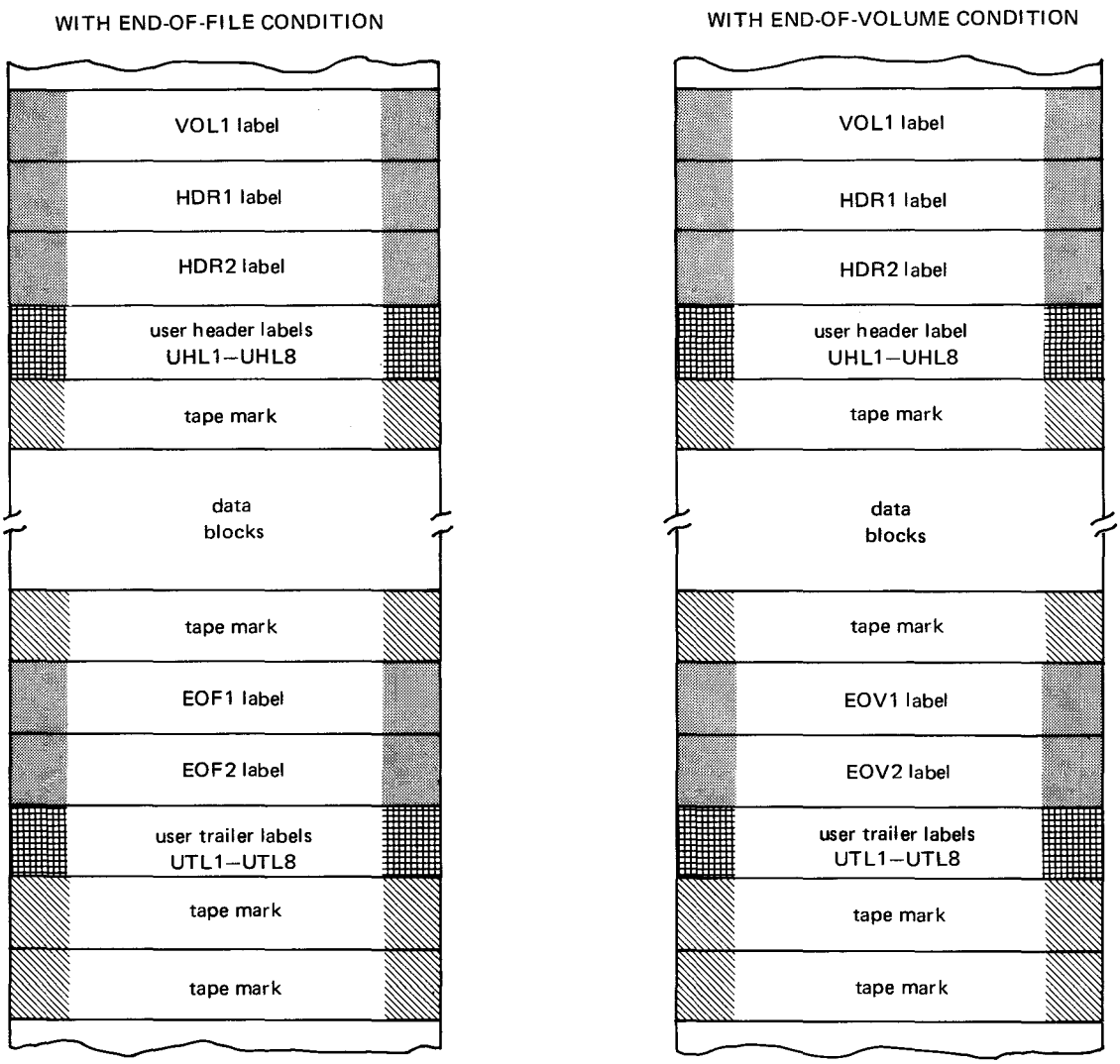
Figure D—10. Tape File EOF2 and EOV2 Label Formats for Block Numbered Files



Field	Bit	Description
Tape mark counter	0	0 = Preceding block is not a tape mark 1 = Preceding block is a tape mark
	1–3	All zeros if bit 0 = 0 0 to 7_{16} if bit 0 = 1. Number of contiguous preceding tape marks (modulo 8)
Block number counter	0–19	Tape block number in binary

Figure D–11. Tape Block Number Field Format

Appendix E. Tape Volume and File Organization



LEGEND:

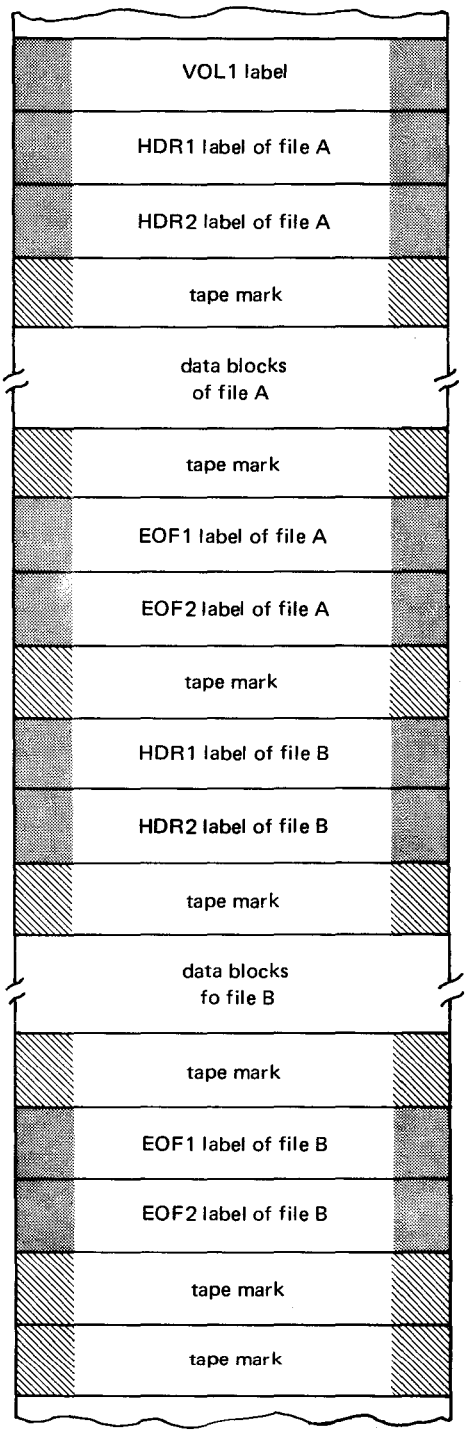
Contents supplied by user.

Required and generated by TSAT.

Generated by TSAT ; user supplies content for certain fields.

Generated by user at his option. Content is at user's option except for content of 4-byte label ID fields. User is limited to eight UHL and eight UTL. Bypassed by TSAT.

Figure E—1. Reel Organization for EBCDIC Standard Labeled Tape Volumes Containing a Single File



LEGEND:

Content supplied by user.

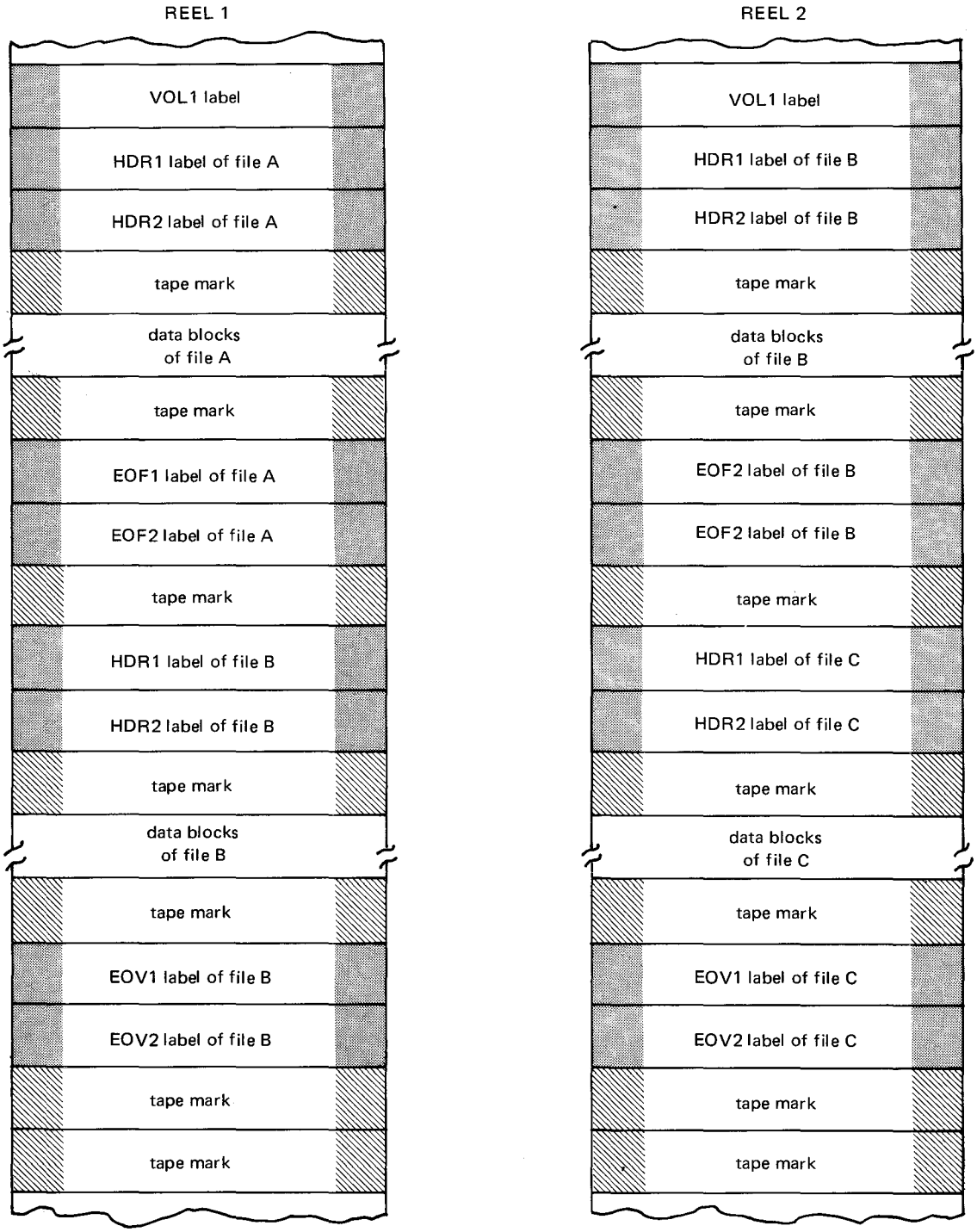
Required and generated by TSAT.

Generated by TSAT; user supplies content for certain fields.

NOTE:

Assume that file B completes on this volume.

Figure E—2. Reel Organization for EBCDIC Standard Labeled Tape Volume: Multifile Volume With End-of-File Condition



LEGEND:

Content supplied by user.

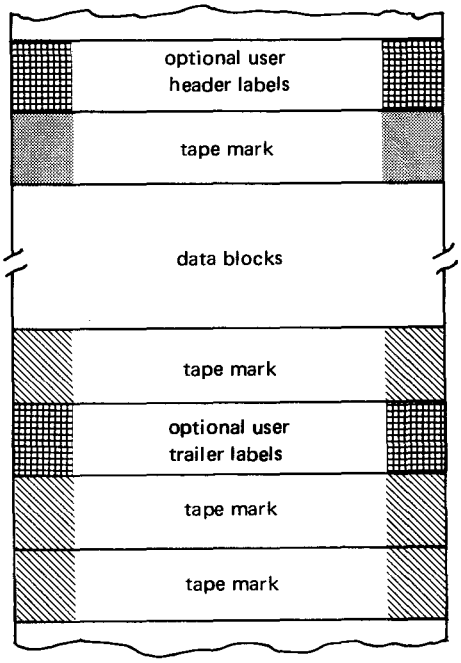
Required and generated by TSAT.

Generated by TSAT; user supplies content for certain fields.

NOTE:

Assume that file C is not completed on reel 2, but carries over (like file B) onto another volume. If file C were completed on reel 2, its EOV1 and EOV2 labels shown here would be replaced with EOF1 and EOF2 labels.

Figure E–3. Reel Organization for EBCDIC Standard Labeled Tape Volumes: Multifile Volumes With End-of-Volume Condition



LEGEND:

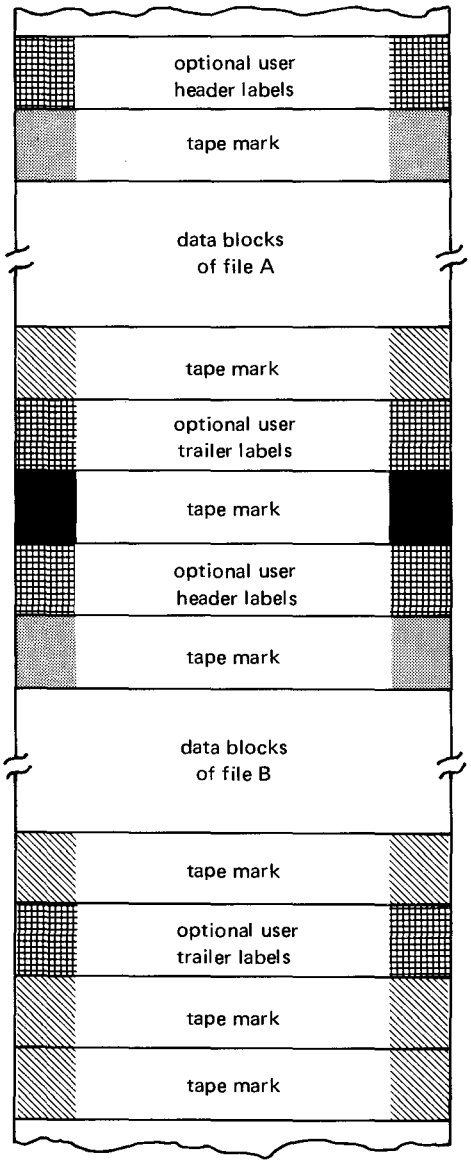
Contents supplied by user.

Required and generated by TSAT; only two tape marks follow data blocks if UTL are not present.

Generated by TSAT unless user specifies TPMARK=NO; required only if label checking is omitted or user specifies READ=BACK.

Presence, content, format, and number entirely at user's option. Bypassed by TSAT.

Figure E—4. Reel Organization for EBCDIC Nonstandard Volume Containing a Single File



LEGEND:

Content supplied by user.

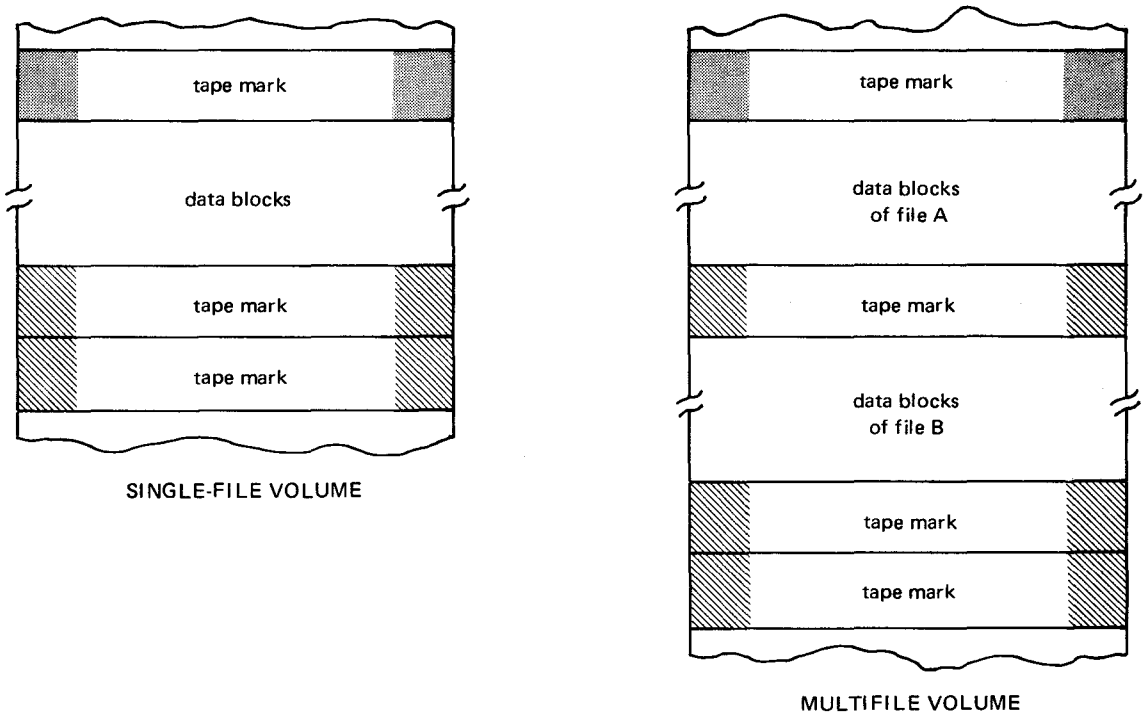
Required and generated by TSAT; only two tape marks follow data blocks of last file on volume if UTL are not present.

Generated by TSAT unless user specifies TPMARK=NO; required only if label checking is omitted or user specifies READ=BACK.

Presence, content, format, and number entirely at user's option. Bypassed by TSAT.

Always present; written by TSAT

Figure E—5. Reel Organization for EBCDIC Nonstandard Multifile Volume



LEGEND:

Content supplied by user.

Required and generated by TSAT; two tape marks follow data blocks of last file on volume.

Generated by TSAT unless user specifies TPMARK=NO; required only when user specifies READ=BACK.

Figure E—6. Reel Organization for Unlabeled EBCDIC Volumes

Appendix F. Monitor and Trace

F.1. TO TRACE PROGRAM EXECUTION

The monitor routine enables you to trace the execution of a program by means of a hardware monitor interrupt so that errors can be located and corrected. In your input to the monitor routine, you can specify actions to be performed at specific points in the program. The monitor routine interrupts each instruction before it is executed and tests for the following conditions specified in the monitor input:

1. A specified storage location is referenced or the data at that location is changed.

2. A specified instruction location is reached.

3. A specified instruction sequence occurs.

4. A specified register is changed.

At each of these possible locations, you can request a monitor printout of current program information (PSW contents, next instruction to execute, etc.), and either:

1. continue program execution under monitor control;

2. suspend program execution; or

3. continue program execution without monitor intervention.

Depending upon how the monitor routine is called into main storage and your choice of actions to be performed, you can monitor an entire task or part of a task.

F.2. REQUIREMENTS

The monitor and trace functions are activated under the following conditions:

1. the monitor routine is in main storage;

2. the monitor bit in the PSW is set;

3. the task to be monitored, location options, and actions have been specified to the monitor routine; and

4. a printer is available.

F.3. TO CALL THE MONITOR ROUTINE INTO MAIN STORAGE

The procedure used to call the monitor routine into main storage depends upon whether you want to begin monitoring from the start of your job, or whether you want to monitor after job execution has begun. In either case, remember that because the monitor routine uses 3K bytes of main storage, you should not overestimate your minimum main storage requirements on the JOB job control otherwise, there may not be enough main storage available for the monitor routine.

F.3.1. To Monitor From the Start of the Job

If you want to begin monitoring with the first instruction executed, the monitor routine must be called into main storage before the job to be monitored is run. In this case, the monitor input is entered as embedded data in the control stream.

The system operator types MO at the system console which brings the monitor routine into main storage. The monitor initializes itself and awaits activation.

The OPTION job control statement for the job step to be monitored must include the TRACE parameter. When input from the control stream is processed, the TRACE parameter activates the monitor by setting the monitor bit in the PSW, and allows it to establish the necessary linkages to the job. Monitoring begins as soon as the program being executed is loaded, and continues until the end of the job step is reached or until the monitor is deactivated by the Q action specified in your monitor input.

F.3.2. To Monitor After Job Execution Has Begun

Once the monitor is activated, it executes several instructions for every instruction in the program being monitored. For a large program, this could require prohibitive amounts of processor time. When you can determine the section of a program in which a problem exists, you can limit the monitoring to this portion. In this case, the monitor routine is called into main storage after job execution has begun and the monitor input is entered through the card reader (or at the system console if there are no card facilities available).

The system operator types 00 MO R at the system console which interrupts the executing job and brings the monitor routine into main storage. It reads in the monitor input from the card reader, initializes itself, and sets the monitor bit in the PSW. The monitor routine immediately scans main storage for the program named on the first card of the monitor input, which means the job to be monitored must be in main storage at this time.

There may be a situation where there is no card reader available to read in monitor input data, or no key punch immediately available to prepare monitor input cards. In this case, the operator types 00 MO C at the system console (C indicates monitor input via the console). This interrupts the executing job and allows the operator to enter the task name, location options, and actions at the console. He enters one card at a time, a line on the screen corresponding to a card in the monitor input data, and indicates end of card by pressing the TRANSMIT button.

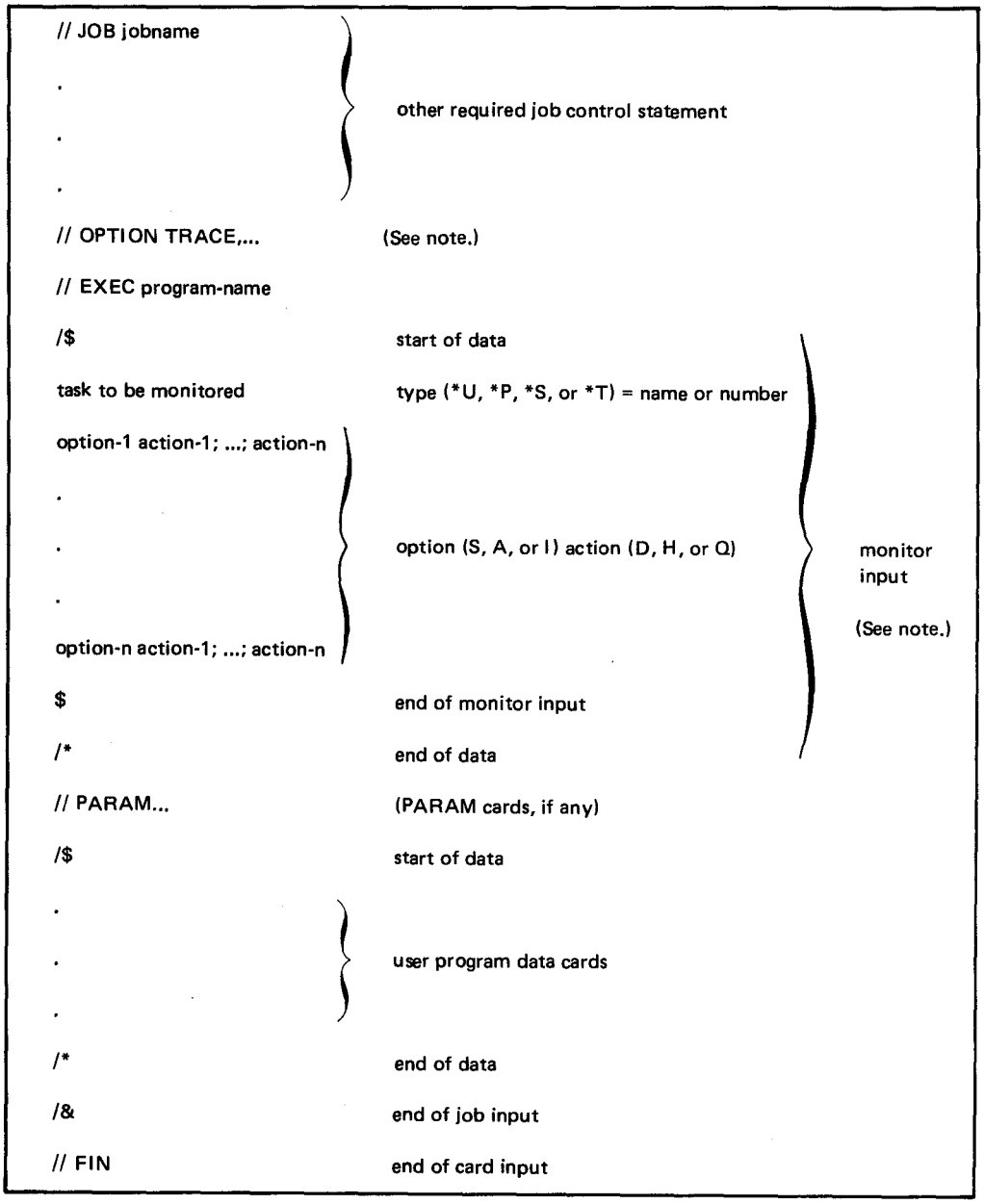
When the monitor routine is called into main storage after job execution has begun, the job step must be suspended to insure that the monitor is activated before the program passes the area of code to be monitored. You can do this by using an ALTER job control statement to change an instruction in your program to a Yield SVC (X'0A04') at some point prior to the code to be monitored. You furnish instructions to the operator to place the monitor input deck in the card reader, and when the job halts, type 00 MO R at the console. Then he types GO, followed by the job name, to resume program execution under monitor control.

If you have an OPR macro instruction with the REPLY operand in your program at a convenient location, you can take advantage of the halt when the program is suspended at this macro instruction. Instruct the operator to position the monitor input deck, type 00 MO R at the console, then enter the reply requested by the OPR macro instruction to resume program execution and commence monitoring.

One other technique that can be used with a long running program is to instruct the operator to type the PAUSE console command at some specified interval after program execution begins. He positions the monitor input deck, types 00 MO R, and finally types GO and the job name to resume program execution and begin monitoring. When using this technique, the *P=phase-name format cannot be used to specify the type of task to be monitored. Use either the *U=jobname or *S=symbiont-name format in the monitor input deck. These formats are described in F.6.

F.4. CONTROL STREAM FORMAT

Figure F—1 shows the control stream format for a job to be monitored from the start of the program.



NOTE 1:

The TRACE entry is required if monitor input is entered via the job control stream.

Figure F—1. Control Stream Format for Monitor Input

In Figure F—1, note that the OPTION job control statement immediately precedes the EXEC job control statement and that the monitor input, delimited by /$ and /*, immediately follows the EXEC job control statement. The addition of this data set does not affect processing of any other control stream data in the job. Note also that if PARAM job control statements are present, they follow the monitor input.

The control stream format for a job to be monitored after program execution has begun is the same as illustrated except that the TRACE option is not required in the OPTION job control statement, and the monitor input and /$ and /* delimiters are not included. Instead, the monitor input (without the delimiters) is entered via the card reader. This procedure was described in F.3.2.

F.5. MONITOR INPUT FORMAT

The input to the monitor routine specifies which task to monitor, when to act, and what actions to perform. Figure F—2 shows the monitor input format. This applies both to input via the control stream before the job is run, and to input by the operator after program execution has begun, except that the /$ and /* delimiters are required only for the control stream.

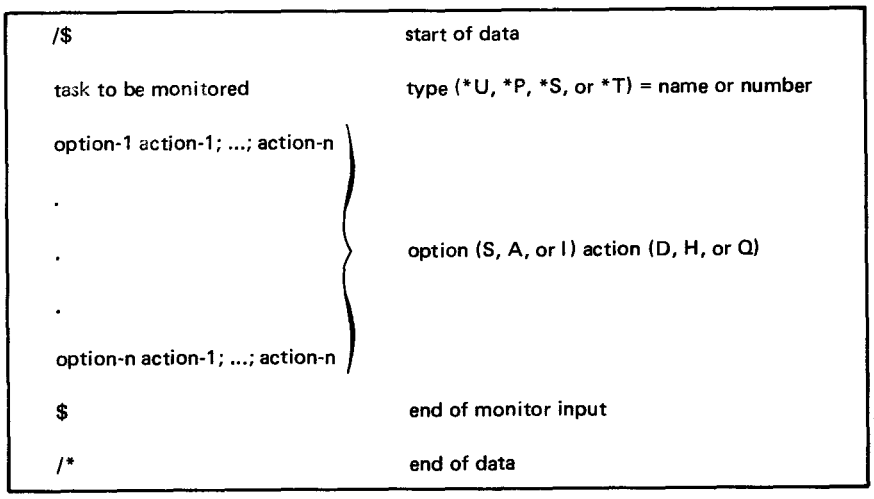
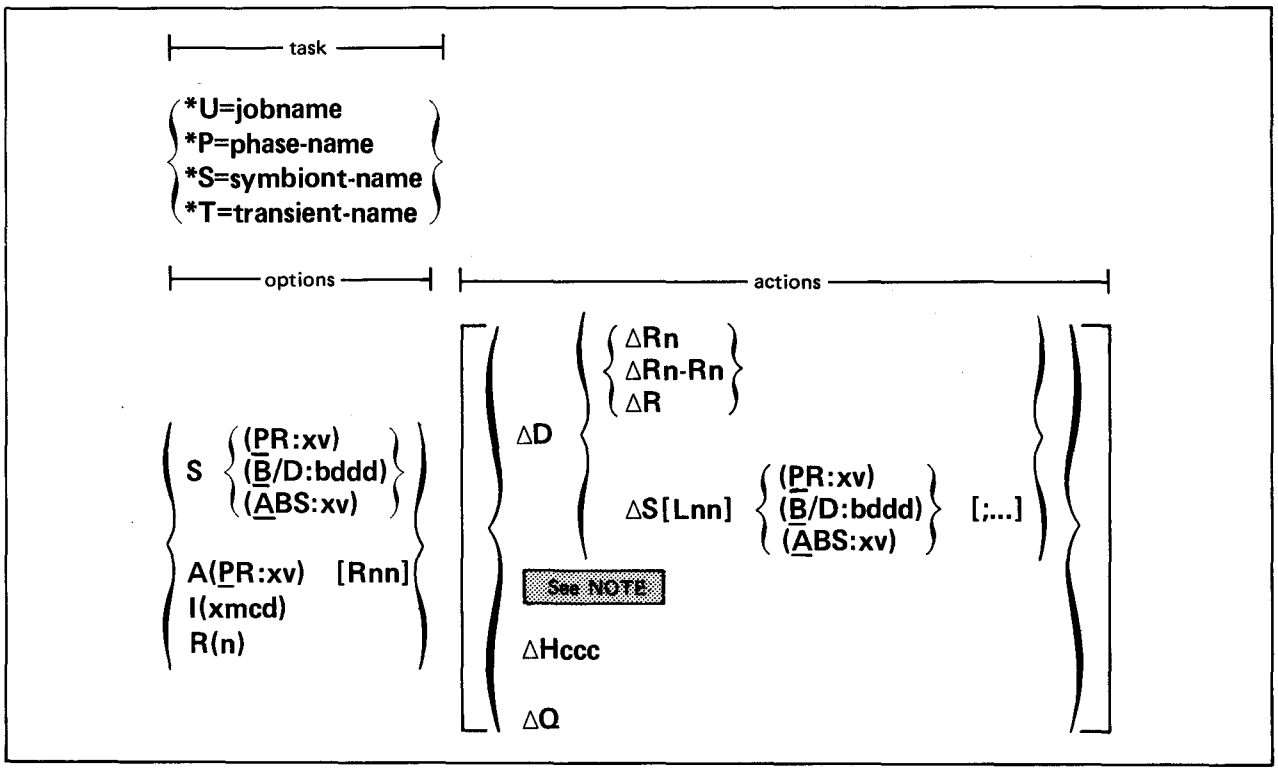


Figure F—2. Monitor Input Format

Only one task can be specified. (See F.6.) However, up to 15 options are allowed and each option may specify a number of actions. The limit of actions per option is the number that can be included on a single card. A space precedes each action, and each action is separated from the next by a semicolon. Options are described in F.7 and actions are described in F.8.

Figure F—3 shows the format of monitor statements that specify the task to be monitored, options to be selected, and actions to be performed.



NOTE:

If no action is specified, the monitor routine produces the default display. (See F.8.1.3.)

Figure F—3. Format of Monitor Statements Specifying Task, Options, and Actions

F.6. TO SPECIFY THE PROGRAM TO BE MONITORED

The first card of your input specifies the type of task to be monitored (user program, user phase, symbiont, or transient) and its name or number. The formats are:

***U=jobname**
***P=phase-name**
***S=symbiont-name**
***T=transient-number**

1. User Program: ***U=jobname** (as on the JOB job control card)

   Example: *U=TEST3

2. User Phase: ***P=phase-name** (as on the linkage editor output)

   Example: *P=RPGII024

3. Symbiont: ***S=symbiont-name** (as in the load library file)

   Example: *S=JC$$DI00

4. Transient: ***T=transient-number** (decimal number of transient)

   Example: *T=24

F.7. TO SPECIFY OPTIONS

The second and succeeding cards in the monitor input specify location options and actions; that is, points in the program where one or more actions are to be taken by the monitor routine.

The first entry in each card specifies an option. This may be followed by one or more actions to be performed at the specified location. The actions to be taken at one location must be completely specified on one card. No continuation to the next card is permitted. If there are duplicate or overlapping options, only the first one specified is processed at execution time. For example, if the same instruction location is specified by two separate cards, the monitor routine performs the actions requested for that location on the first card, then executes the instruction. The second card, therefore, is never considered for that location.

Options may be specified in any sequence. There is no need to list them according to any pattern. Remember that in the case of overlapping options, only the first option matching the current program location is processed.

There are four types of options you can specify:

1. Storage reference (S)
2. Instruction location (A)
3. Instruction sequence (I)
4. Register change (R)

The formats for these options are:

```
{ S {(PR:xv)     }
    {(B/D:bddd)  }
    {(ABS:xv)    }

  A(PR:xv)[Rnn]
  I(xmcd)
  R(n)           }
```

F.7.1. Storage Reference Option (S)

This option requests the monitor routine to take action when the specified storage location is referenced or the data at that location is changed. There are three ways to express the location in a storage reference option:

1. Program relative (PR)
2. Base/displacement (B/D)
3. Absolute (ABS)

F.7.1.1. Program Relative Address (PR)

The format for the storage reference option using a program relative address is:

```
S(PR:xv)
```

where:

xv
: May range from 0_{16} to $FFFFFF_{16}$.

The address (xv) can consist of from one to six hexadecimal characters. For example:

S(PR:3C)

This option is selected if program execution reaches an instruction that references storage at program relative address 3C. The location specified need not be the first byte of a field. For example, a Move instruction from location 2E for 18 bytes would be detected because the specified program relative address 3C falls within the field moved (2E to 3F).

For a user program, a phase, or a symbiont, a program relative address is relative to the start of the linked program. For a transient, it is relative to the start of the transient.

When the storage reference option with a program relative address is specified, it is possible to get two displays from a single S option. The first comes just before the data changes at a specified location, and the second just after the change. In this case, the instruction displayed when the change occurs (identified on the printout as the NEXT INST) is the instruction following the one that caused the change.

F.7.1.2. Base/Displacement Address (B/D)

The format for the storage reference option using a base/displacement address is:

S(B/D:bddd)

where:

b
: May range from 0_{16} to F_{16}.

ddd
: May range from 000_{16} to FFF_{16}.

The address (bddd) must consist of four hexadecimal characters; that is, one character specifies the base register, followed by three characters specifying the displacement. For example:

S(B/D:4B29)

This option requires an instruction to contain a storage reference of 4B29 for a match to occur; that is, a storage address using base register 4 and a displacement of B29.

F.7.1.3. Absolute Address (ABS)

This option is used mainly by system programs for symbiont and transient routines that can refer to locations outside of their areas.

The format for the storage reference option using an absolute address is:

S(ABS:xv)

where:

xv
May range from 0_{16} to $FFFFFF_{16}$.

The address (xv) can consist of from one to six hexadecimal characters. For example:

S(ABS:35AE)

This option is selected if the program reaches an instruction that references storage at absolute address 35AE.

F.7.1.4. Alternate Address Mnemonics

For convenience, you can specify the type of address expression using only the first letter of the address mnemonic. For example:

Type of Address	Mnemonic
Program Relative	PR or P
Base/Displacement	B/D or B
Absolute	ABS or A

F.7.2. Instruction Location Option (A)

This option requests the monitor routine to take action when the specified instruction location is reached. It uses a program relative address (as with the storage reference option). However, you can also add a range to continue the monitor actions for the number of bytes specified. The format for the instruction location option is:

A(PR:xv)[Rnn]

where:

xv
May range from 0_{16} to $FFFFFF_{16}$.

nn
May range from 02_{16} to FF_{16}.

For convenience you can use the mnemonic P instead of PR. The address can consist of from one to six hexadecimal characters. For example:

A(PR:C02) or A(P:C02)

This option is selected if program execution reaches the instruction at program relative address C02.

You can continue monitor action for up to 254 bytes by specifying a range. For example:

A(PR:C02)R7E

This option begins monitor action when the instruction at program relative address C02 is reached and continues for 126 bytes (until program relative address C80 is reached). Note that you must use two hexadecimal characters for the range even when it can be expressed in one.

F.7.3. Instruction Sequence Option (I)

This option requests the monitor routine to take action when the exact instruction sequence specified is reached. The monitor routine compares the machine code specified in the option entry to the actual instruction sequence of each instruction to be executed in the program being monitored, and takes action when an exact match occurs. The format for the instruction sequence option is:

I(xmcd)

where:

xmcd
May consist of from 2 to 64 hexadecimal characters (1 to 32 bytes of machine code).

You can specify a string of instructions, a single instruction, or just the operation code of an instruction. For example:

I(47)

This option is selected whenever a branch instruction with any condition is reached.

I(4780)

This option is selected whenever a branch on equal instruction is reached.

I(D205C006E0141B4405E0)

This option is selected when the program reaches the following sequence of instructions.

Assembler Code		Machine Code
MVC	6(6,12),20(14)	D205C006E014
SR	4,4	1B44
BALR	14	05E0

F.7.4. Register Change Option (R)

This option requests the monitor routine to take action when the contents of the specified register are changed. The monitor routine compares the current register contents to the previous contents, and takes action when a change occurs. The instruction displayed when the change occurs (identified on the printout as the NEXT INST) is the instruction following the one that caused the change.

The format for the register change option is:

R(n)

where:

n
Is a hexadecimal character (0 to F) representing the number (0 to 15) of the register to be tested.

For example:

R(5)

This option is selected whenever the contents of register 5 change.

Because register contents may be changed frequently during program execution, this option could produce a considerable amount of printout.

F.7.5. Default Option

If you omit the option specifications (if your monitor input consists only of the *U, *P, *S, or *T card, and the $ card), the monitor routine interrupts each instruction in the task prior to its execution and prints out pertinent information at that point in the program. The processor then executes the interrupted instruction (identified on the printout as the NEXT INST). Then the succeeding instruction is interrupted, the printout produced, and the instruction executed. This interrupt, printout, and execution pattern is performed for each instruction processed. This could required excessive processor time and could produce a huge printout of unneeded information. Therefore, you would use the default option only for special cases.

The program information printed is the same for the default display described in F.8.1.3, except that the option causing the printout doesn't apply and is omitted.

F.8. TO SPECIFY ACTIONS

The second and succeeding entries on option cards specify actions to be taken when that location is reached. Actions for one option must be completely specified on one card. No continuation to the next card is permitted. If there are duplicate or overlapping options, only the first one specified is processed at execution time, and any action specified on the second card for the same option is never considered.

There are three actions you can specify:

1. Display (D)

   Prints out program information (including specified registers or storage) and continues monitor processing.

2. Halt (H)

   Prints out program information and suspends the job.

3. Quit (Q)

   Prints out program information and deactivates the monitor routine.

A fourth possibility is to omit the action specification. In this case the monitor uses the default printout; that is it prints out program information (including changed registers and storage) and continues monitor processing.

Each of the actions produces a printout of program information. This is summarized in Table F—1.

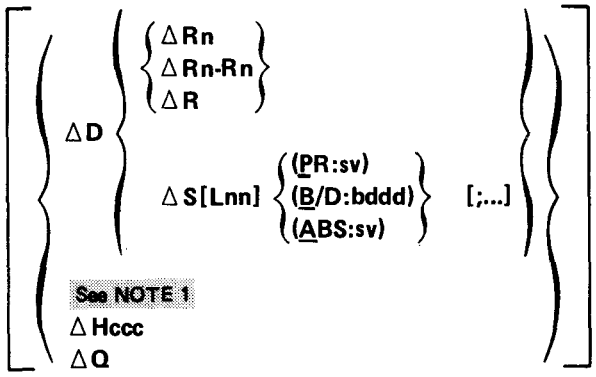
Table F—1. Summary of Actions and Program Information Printed

	Program Information Printed	Action				
		Display Register (DR)	Display Storage (DS)	Default Display	Halt (H)	Quit (Q)
1.	Job name*	x	x	x	x	x
2.	TCB address*	x	x	x	x	x
3.	Program base address*	x	x	x	x	x
4.	PSW contents	x	x	x	x	x
5.	Next instruction to execute	x	x	x	x	x
6.	Option causing this printout	x	x	x	x	x
7.	Contents of specified registers	x				
8.	Contents of specified storage		x			
9.	Contents of changed registers			x		
10.	Contents of referenced storage			x		
11.	HALT message				x	

*These items are included only for the first option that causes a printout.

Note that the job name, TCB address, program base address, PSW contents, next instruction to execute, and option causing this printout (items 1 to 6) are printed for all actions. In addition, the printout for each of the actions (except Quit) includes program information specific to that action (items 7 to 11).

The formats for these actions are:



NOTE 1:

If no option is specified, the monitor routine produces the default display (See F.8.1.3.)

These actions are described on the following pages.

F.8.1. Display Action

There are three types of display specifications:

1. Register display (D R)
2. Storage display (D S)
3. Default display

These three actions have similar functions; that is, program information is printed, then the instruction causing the printout is executed and program processing continues under monitor control. The printouts are the same except for a printout of registers, storage locations, or both depending upon the type of display action requested.

F.8.1.1. Register Display (D R)

For this action the monitor routine prints out the following program information:

- Job Name
- TCB Address
- Program Base Address

} For the first option that causes a printout

- PSW Contents
- Next Instruction to Execute
- Option Causing this Printout
- Contents (4 bytes) of the Specified General Registers

The instruction causing the printout is executed and program processing continues under monitor control.

You can specify one or more registers or all 16 general registers. There are three ways to specify a register printout:

Format	Specifies
D Rn	Register n
D Rn-Rn	Register n to register n, inclusive
D R	All 16 registers

where:

n
: Is a hexadecimal character (0 to F) representing the number (0 to 15) of the register to be printed.

For example:

Format	Specifies
D R1	Prints register 1
D R3—R6	Prints registers 3, 4, 5, and 6
D R	Prints all 16 registers

F.8.1.2. Storage Display (D S)

For this action the monitor prints out the following program information:

- Job Name
- TCB Address
- Program Base Address

} For the first option that causes a printout

- PSW Contents
- Next Instruction to Execute
- Option Causing this Printout
- Contents of the Specified Storage Locations

The instruction causing the printout is executed and program processing continues under monitor control.

You can specify up to 256 consecutive bytes of main storage or you can omit the length, in which case the monitor prints 8 consecutive bytes starting at the specified storage location. Except for the length option, the formats for the storage display action are similar to those for the storage reference option (described in F.7.1).

The formats for the storage display actions are:

D S [Lnn](PR:xv)

D S [Lnn](B/D:bddd)

D S [Lnn](ABS:xv)

where:

L
Indicates an optional length specification.

nn
Specifies the number of bytes and may range from 01_{16} to FF_{16}.

xv
May range from 0_{16} to $FFFFFF_{16}$.

b
May range from 0_{16} to F_{16}.

ddd
May range from 000_{16} to FFF_{16}.

The addresses are expressed in the same way as for the storage reference option. For example:

Address	Meaning
D S (PR:3C)	Displays 8 bytes starting at program relative address 3C
D S (B/D:1B29)	Displays 8 bytes starting at the address using base register 1 and a displacement of B29
D S (ABS:35AE)	Displays 8 bytes starting at absolute address 35AE

You can change the number of bytes to print by using the length option. For example:

Address	Meaning
D SL14(PR:3C)	Displays 20 bytes starting at program relative address 3C
D SL1E(B/D:B29)	Displays 30 bytes starting at the address using base reigster 1 and a displacement of B29
D SL04(ABS:35AE)	Displays 4 bytes starting at absolute address 35AE

Note that you must use two hexadecimal characters for the length even when it can be expressed in one.

F.8.1.3. Default Display

You can omit the action specification; that is, you can enter an option without specifying the related action you want taken at that point in the program. In this case, the monitor routine prints out the following program information:

Job Name
TCB Address
Program Base Address
} For the first option that causes a printout

PSW Contents

Next Instruction to Execute

Option Causing this Printout

Contents of any Registers that were Changed since the last Printout

Contents of the Storage Locations Referenced by the Instruction Causing the Printout

The instruction causing the printout is executed and program processing continues under monitor control.

For example, assume your monitor input includes the following card:

S(B/D:4B29)

When program execution reaches an instruction that references an address using base register 4 and a displacement of B29, the monitor produces a default printout. Then the instruction causing the printout is executed and program processing continues under monitor control.

This printout is the same for the default option (where no option cards are included in the monitor input) except that the default option printout does not include the option causing the printout.

F.8.2. Halt Action (H)

For this action the monitor routine prints out the following program information:

Job Name
TCB Address
Program Base Address
} For the first option that causes a printout

PSW Contents

Next Instruction to Execute

Option Causing this Printout

HALT Message

The halt message is printed in the following format:

HALT cccc. TYPE-IN GO jobname TO RESUME

Program execution is suspended until the operator types GO and the job name at the console. Then the instruction causing the printout is executed and program processing continues under monitor control.

The format for the halt action is:

Hccc

where:

ccc
Is a 3-character code in EBCDIC that you specify to identify this particular halt.

One use for this option is to permit the operator to initiate a main storage dump. You would instruct him to take a dump when a specific halt message is printed. For example, assume that you include the following option card in your job named TESTM:

A(PR:1B4) H-DU

When program execution reaches the instruction at program relative address 1B4, the monitor routine prints out the program information and the following message:

HALT -DU. TYPE-IN GO TESTM TO RESUME

You would instruct the operator to take a dump when he sees the HALT -DU message on the printer. After using the DUMP console command to take a dump, he then types GO TESTM to reactivate the interrupted job. The instruction at program relative address 1B4 is executed and program processing continues under monitor control.

F.8.3. Quit Action (Q)

For this action the monitor routine prints out the following program information:

Job Name
TCB Address
Program Base Address
} For the first option that causes a printout

PSW Contents

Next Instruction to Execute

Option Causing this Printout

The instruction causing the printout is executed and program processing continues without monitor intervention.

The format for the quit action is:

Q

For example:

A(PR:F18) Q

When program execution reaches the instruction at program relative address F18, the monitor routine prints out the program information. Then the instruction at program relative address F18 is executed and program processing continues without monitor intervention.

If this action is omitted from the monitor input, monitor processing continues until the job step being monitored terminates.

F.9. CANCEL OF MONITOR

If the monitor routine is terminated abnormally, either by a CANCEL command or by a program exception within the monitor routine, all programs requesting the monitor routine will continue normal program processing without any type of monitor intervention. The monitor routine itself will cause a dump and leave the system.

Glossary

B

BCW (buffer control word)

A control block that specifies the operation, data address, I/O storage protection key, byte count, and other control information used by the integrated peripheral channels and the multiplexer channel.

block number processing

A technique to ensure correct tape positioning whereby a block sequence number is written on output tape files and checked on input tape files. This block count is recorded on tape in the first three bytes of the block, and consists of a 4-bit tape mark count and a 20-bit block number count.

breakpoint

A point in a spool subfile where the subfile is closed then reopened so that the contents of the subfile can be output to the physical device before the job step terminates.

C

canned message

A generalized message kept in a cataloged message file on disc that may be displayed on the system console, and written to the system log file.

CAW (channel address word)

A control block that specifies the location of the first CCW to be used in the execution of an I/O operation.

CCB (command control block)

A control block that is used as a bidirectional communications medium between the user program and the PIOCS routines in the supervisor.

CCW (channel command word)

A control block that specifies the operation, data address, byte count, and other control information used by the selector channel.

checkpoint

A point in a program or routine where a recording of the contents of main storage and the state of peripheral units is made so that, in the event of machine failure or some other interruption, a job can be restarted at an intermediate point rather than from the beginning.

communication region
A 12-byte field in the job preamble used to pass information from one job step to the next.

control stream
A sequence of control statements that define one or more jobs to the operating system and may include source code or data as required by the jobs.

current ID
A field in the PCA table that contains the starting address of the logical partition or the address of the current record being processed.

D

DTF (define the file) table
A control block that contains the file name and operating and physical characteristics of a file used by the SAT routines.

dump
To copy the contents of all or part of main storage. Also, the data resulting from the process.

E

ECB (event control block)
A control block that identifies a subtask and indicates status to the other tasks within a job step. An ECB is created for each subtask.

embedded data set
Data in the form of card images entered into the system with the job control stream.

end of data ID
A field in the PCA table that contains the address of the last logical record of the partition.

F

FCB (file control block)
File and device information compiled by job control and stored by the supervisor in the PIOCB.

file identifier
The physical label of a file. It is specified in the LBL job control statement and, for a disc file, may have up to 44 bytes. It is used to locate the VTOC entry that contains control information for this disc file.

filelock
A lock to prevent unauthorized access to a file. This lock is set at the logical level. Only macro instructions specifying a required password prefixed to the file name can access the file.

file name
The name of the logical file. It is specified in the LFD job control statement and may have up to eight alphanumeric characters. It must be entered in the label field of the DTFPF macro instruction when you define a partitioned file, or in the label field of the PIOCB macro instruction when you want to use the PIOCS function of the supervisor. It is the logical file name used to access a file and to locate the FCB block for a file.

I

interlace

A technique whereby blocks on a partitioned file are spaced on the track so that more than one I/O operation may be performed per disc revolution. The interlace factor is specified by the LACE keyword parameter of the PCA macro instruction.

island code

Closed routines by which you may handle interrupts, contingencies, or events not normally processed by the supervisor. These island code routines are activated when your program is interrupted for:

- Abnormal termination

  An error occurs, making continuation of the program impossible.

- Interval timer

  The time interval previously specified by a SETIME macro instruction elapses.

- Operator communications

  An unsolicited message is entered at the system console by the operator.

- Program check

  An operation in the problem program causes a hardware program check interrupt, such as an addressing violation, an arithmetic overflow, or an operation exception.

J

job

A total processing application comprising one or more processing steps. Each job is divided into job steps (programs) that are executed serially. With the exception of disc space, resources are allocated on a job basis.

job region

An area in main storage reserved for each job containing the control information (prologue) and the program code for the job step being executed.

job step

The unit of work associated with one processing program. A job step is an executable program consisting of one or more tasks and requiring a specific amount of the hardware resources of the system.

L

library search order

The default order of search employed by the loader is:

1. Load library file (YLOD)

2. Job run library file (YRUN)

If the job run library file (YRUN) is specified on the EXEC job control card, the order of search is:

1. Job run library file (YRUN)
2. Load library file (YLOD)

If an alternate library is specified on the EXEC job control card, the order of search is:

1. Alternate load library
2. Load library file (YLOD)
3. Job run library file (YRUN)

To minimize search time, the loader always begins searching a library at the last root phase loaded from that library for that job. This means that it is generally more efficient to link modules together than to create a series of smaller, separately linked load modules.

M

main storage consolidation

The repositioning of jobs in main storage in order to run a job requiring more contiguous space than is currently available.

monitor

A supervisor routine that interrupts each instruction in a program, or a part of a program, so that a trace of program execution can be made.

multijobbing

The concurrent scheduling, loading, and execution of more than one job at a time. Up to seven jobs can be processed concurrently, with each job consisting of one or more job steps.

multitasking

The concurrent processing of many tasks asynchronously. Multitasking applies to the switching of processor control among two or more tasks on a priority or rotational basis. Job steps with more than one task are capable of using multitasking.

P

PCA (partition control appendage)

A control block that contains the characteristics of a partition within a file used by the SAT routines.

PIOCB (physical input/output control block)

A buffer reserved in main storage into which the FCB information for a file is stored by the supervisor.

PIOCS (physical input/output control system)

A set of resident routines that controls the activity between the processor and all peripheral devices connected to the multplexer, selector, and integrated channels.

preamble
The portion of the prologue in the job region containing control information for a job.

processor
A unit of the computer that includes the circuits controlling the interpretation and execution of instructions. Synonymous with central processing unit.

program phase (load module phase)
A program segment that can perform one or more specific processing operations. A program phase is output by the linkage editor and stored in a load library, then located and read into main storage by the program loader routine.

prologue
The portion of the job region containing the preamble, TCBs, and disc storage extent control information for a job.

PSW (program status word)
A field in main storage that contains an instruction address and control information for the program currently being executed. When an interrupt occurs, the supervisor stores the PSW in a PSW save area, suspends the task, processes the interrupt, and then uses the stored PSW to resume the interrupted task.

PUB (physical unit block)
A control block in main storage for each I/O device containing the physical address, characteristics, status, and other control information for the device.

R

restart
To resume processing a job from some intermediate point (called a checkpoint) following an interruption.

rollout/rollin
The temporary transferring of jobs from main storage to disc to make room for a job with a preemptive scheduling priority.

S

SAM (sequential access method)
A data management program that reads and writes records on a physical device serially.

SAT (system access technique)
An input/output control system that provides a standard interface for tape and disc subsystems between OS/3 data management and the PIOCS.

SIB (system information block)
An area in main storage, which is part of the resident supervisor, containing system control information.

spooling (simultaneous peripheral operation on line)

A technique that increases the throughput of a system by buffering data files for low speed input and output devices to a high speed storage device independently of the program that uses the input data or generates the output data. Data from card readers or from remote sites is stored on disc for subsequent use by the intended program. Data output by the program is stored on disc for subsequent punching or printing.

subtask

A task that is created by a user ATTACH macro instruction and executes concurrently with the parent task.

symbiont

A system utility routine that operates concurrently with other system programs and with user programs and is executed in the same manner as a job step.

T

task

A unit of work capable of competing with other tasks for control of the central processor. A task is a logical point of control rather than a physical set of instructions. Each job step has at least one task (the primary task) and may have additional tasks (subtasks), all of which compete independently for processor time. There may be a maximum of 256 tasks per job.

TCA (tape control appendage)

A control block that contains the characteristics of a magnetic tape file used by the SAT routines.

TCB (task control block)

A control table generated for each task and stored in the job region prologue. A TCB is constructed for each job step submitted by job control for execution. Additional TCBs are constructed for each subtask created by the ATTACH supervisor macro instruction.

trace

A diagnostic technique in the monitor routine that prints program information (PSW and register contents, etc.) at specified points in a program executing in the monitor mode so that errors can be located and corrected.

SPERRY UNIVAC

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

_____

(Document Title)

_____ _____ _____

(Document No.) *(Revision No.)* *(Update No.)*

Comments:

Cut along line.

From:

_____

(Name of User)

_____

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY UNIVAC

ATTN.: SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

CUT

FOLD

Comments concerning this manual may be made in the space provided below. Please fill in the requested information.

System: _____

Manual Title: _____

UP No: _____ Revision No: _____ Update: _____

Name of User: _____

Address of User: _____

Comments:

NOTE: DO NOT USE THIS FORM TO ORDER MANUALS.

FOLD

FIRST CLASS
PERMIT NO. 21
BLUE BELL, PA.

BUSINESS REPLY MAIL NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

SPERRY UNIVAC

P.O. BOX 500
BLUE BELL, PA.
19422

ATTN: SYSTEMS PUBLICATIONS DEPT.

CUT

FOLD

SPERRY UNIVAC

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

_____

(Document Title)

_____ _____ _____

(Document No.) *(Revision No.)* *(Update No.)*

Comments:

Cut along line.

From:

_____

(Name of User)

_____

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY UNIVAC

ATTN.: SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

CUT

FOLD