Dick

IBM Technical Newsletter

File Number	S360-20
Re: Order No.	GC28-6550-8
This Newsletter No.	GN28-2452
Date	January 1, 1971
Previous Newsletter Nos.	N28-2437

IBM System/360 Operating System: System Programmer's Guide

This Technical Newsletter, a part of release 20 of IBM System/360 Operating System, provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent releases unless specifically altered. Pages to be inserted and/or removed are:

Cover,2
Contents
9,10,10.1
55,56
63,64,64.1
69,70
97-104
107-110
125-128
181-186,186.1-.2-.3-.4
189,190
191-196
199-202
207,208
211-214,214.1
225,226
229-234,234.1-.2
247-248
263-266
273-276
279,280
283,284
305,306
Index

A change to the text or a small change to an illustration is indicated by a vertical line to the left of the change; a changed or added illustration is denoted by the symbol • to the left of the caption.

Summary of Amendments

This Technical Newsletter provides information related to Release 20. See the Summary of Major Changes for a description of the items in 20.

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Programming Systems Publications, P.O. Box 390, Poughkeepsie, N.Y. 12602

PRINTED IN U.S.A.

File No. S360-20
Order No. GC28-6550-8

OS

Systems Reference Library

IBM System/360 Operating System:

System Programmer's Guide

This publication consists of self-contained chapters, each of which provides information on how to modify, extend, or implement capabilities of the IBM System/360 Operating System control program. It is designed primarily for system programmers responsible for maintaining, updating, and extending the operating system features.

Topics:

- Catalog and VTOC Maintenance
- Adding SVC Routines
- Message Routing Exit Routines
- Adding Accounting Routines
- System Management Facilities
- IECDSECT, IEFJFCBN, and IEFUCBOB Macro Instructions
- The Must Complete Function of ENQ/DEQ
- The EXCP Macro Instruction
- The XDAP Macro Instruction
- The Tracing Routine
- Implementing Data Set Protection
- PRESRES Volume Characteristic List
- Residency Options and Link Pack Area
- Job Queue Format
- System Macro Instructions
- Adding System Output Writer Routines
- Output Separation
- System Reader, Initiator, and Writer Cataloged Procedures
- Writing Rollout/Rollin Appendages
- Adding a UCS Image to the System Library
- The Shared Direct Access Device Option
- The Time Slicing Facility
- Graphic Job Processor Procedures
- Satellite Graphic Job Processor Procedures

Information for TSO (Time Sharing Option), the Model 195, and the Model 165 is for planning purposes only.

Preface

This publication consists of self-contained chapters, each of which provides information on how to modify, extend, or implement capabilities of the IBM System/360 Operating System control program. Although the information in one chapter is sometimes related to information in another, all chapters have been written as separate and complete units. It is assumed that users of this publication are thoroughly familiar with the design of the operating system and its features. Each chapter contains its own introductory section and list of prerequisite publications. This organization has been used to reduce cross-referencing and to facilitate the addition of new chapters.

Eighth Edition (June, 1970)

This is a major revision of, and obsoletes, GC28-6550-6. Other changes to the text and small changes to illustrations, are indicated by a vertical line to the left of the change; changed or added illustrations are denoted by the symbol • to the left of the caption.

This edition, with the addition of Technical Newsletter GN28-2452, applies to release 20 of IBM System/360 Operating System, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/360 SRL Newsletter, Order No. GN20-0360, for the editions that are current and applicable.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Systems Publications, Department D58, PO Box 390, Poughkeepsie, N. Y. 12602

Contents

Illustrations

Figures

Tables

Summary of Major Changes--Release 20

Item	Description	Chapter Affected
PROTECT Macro Instruction	A new macro instruction that can be used to maintain the password data set has been added.	Data Set Protection.
STAE Macro Instruction	Two new parameters have been added.	System Macro Instructions.
ASCII	The macro definitions for the UCB and JFCB have been modified to include ASCII. In addition, restrictions against using ASCII data sets in the reader input stream have been added.	IECDSECT,IEFJFCBN, and IEFUCBOB Macro Instructions. System Reader, Initiator, and Writer Cataloged Procedures.
Models 155/165	New devices have been added to the device type characteristics description.	System Macro Instructions.
Dedicated Data Sets	Additional information on the disposition of dedicated data sets, by allocation/termination, has been added.	System Reader, Initiator, and Writer Cataloged Procedures.
Direct System Output Writer	The description of the direct system output writer procedure has been changed to omit the separator function.	System Reader, Initiator, and Writer Cataloged Procedures.
2150 Console	The 2150 console has been removed from the device type characteristics description.	System Macro Instructions.
SYS1.MANX and SYS1.MANY	Addition of SYS1.MANX and SYS1.MANY to list of data sets that cannot be shared.	The Shared Direct-Access Device Option.
GSP Routines	Addition of reenterable GSP routines to group of modules that can be put in the MFT link pack area.	Resident Routines Option.
System Management Facilities	Modifications to the MDL= and OPI= parameters.	System Management Facilities.
Procedure INITD	Removal of ABEND DD statement from INITD procedure.	System Reader, Initiator, and Writer Routines.

Output From Accounting Routines

You can write output in three ways: by issuing console messages; by using the standard system output; by using an IBM-supplied accounting data set writer.

1. Console messages -- You can use Write to Operator (WTO) or Write to Operator with Reply (WTOR) macro instructions.

2. System output -- You must assemble the following calling sequence into your routine. The contents of register 12 must be the same as when your accounting routine was entered, and register 13 must contain the address of an area of 32 fullwords.

   When writing an accounting routine for inclusion in the job scheduler, you must be aware that register saving conventions within the control program are different from those for problem programs. In the job scheduler, registers are saved in the sequence 0-14 in a 15-word save area. There is no place provided to save register 13. You must provide some other means of saving register 13; you may either save it in another register or provide additional save area that is not known to the control program. This can be done by adding a word to the end of the save area that is provided and is addressed as SAVE + 60.

Name	Operation	Operand	
	MVC	36(4,12),MSGADDR	MOVE MESSAGE ADDRESS AND
	MVC	42(2,12),MSGLEN	LENGTH TO SYSTEM TABLE
	L	REG15,VCONYS	BRANCH AND LINK TO MESSAGE
	BALR	REG14,REG15	ROUTINE
	.		
	.		
MSGADDR	DC	A(MSG)	
MSG	DC	C'text of message'	
MSGLEN	DC	H'two character length of message'	
VCONYS	DC	V(IEFYS)	

3. Accounting Data Set Writer -- This writer places accounting records you have constructed in your accounting routine in a data set named SYS1.ACCT. The data set must reside on a permanently resident direct access device. You must provide, in your accounting routine, linkage to the writer, and pass the beginning address of the record to be written, to it.

   Appendix A of this chapter discusses the use of the data set writer.

Sample Accounting Routine

A sample accounting routine, showing use of the data set writer, output to system output, and issuance of console messages, is stored under the member name SAMACTRT in the SYS1.SAMPLIB data set furnished with the starter operating system.

Inserting an Accounting Routine Into the Control Program

Your accounting routine can be inserted in the control program in two ways; by placing the routine on the SYS1.CI505 data set used in system generation or by placing the routine in the appropriate load module of the control program after system generation. The effect of either action is to replace a dummy accounting routine with your accounting routine.

Insertion at System Generation

To insert your accounting routine into the control program during system generation, you must, prior to the start of the system generation process, place your routine in the SYS1.CI505 data set, using the linkage editor. The SYS1.CI505 data set (furnished with the starter operating system) contains load modules which are combined during the system generation process to form the load modules composing the control program. In response to the specification made in the system generation SCHEDULR macro instruction, your accounting routine is incorporated in the appropriate load modules for the system being generated.

You must place your accounting routine in the SYS1.CI505 data set under the name IEFACTRT. You will be replacing the dummy accounting routine -- also named IEFACTRT.

Insertion After System Generation

To insert your accounting routine into the control program after system generation you place the routine in load modules of the scheduler component of the generated control program, using the linkage editor. The scheduler load modules are in the linkage library (SYS1.LINKLIB data set) of the generated system. The affected load modules of the three PCP schedulers (18K, 44K, 100K), the MFT schedulers (30K, 44K), and the MVT scheduler are as follows:

PCP Configurations

18K Scheduler

load module IEFSELCT -- step initiation
load module IEFSTERM -- step termination
load module IEFJTRM1 -- job termination

44K Scheduler

load module IEFSTERM -- step initiation/termination
load module IEFJTERM -- job termination

100K Scheduler

load module GO -- step initiation/termination and job termination

MFT Configurations

30K Scheduler

load module IEFSD520 -- step initiation
load module IEFSD515 -- step/job termination

44K Scheduler

load module IEFW21SD -- step initiation
load module IEFSD515 -- step/job termination

Getting SMF Into Your System

Getting SMF into your system requires that you include the necessary parameters in your system generation statements, and that you define how you want to use SMF. If you are using the SMF data set (and you should, to gather most quickly the data made available by SMF) you must allocate space for it. If you are taking advantage of the user exits, your exit routines must be added to the system.

HOW TO INCLUDE SMF IN YOUR SYSTEM GENERATION

To include SMF in your system, you add the SMF parameter to the scheduler definition in your system generation procedure. Its format is:

```
     SCHEDULR    ...,ACCTRTN=SMF,...
```

To be effective, the control program definition must include MFT, MVT, or M65MP, the CPU definition must include the timer, and the supervisor definition must include the job step timer. (This holds whether or not you plan to use step data.)

Including SMF in your system adds the following code to the named system data sets:

- The SMF data collection routines - SYS1.LINKLIB and SYS1.NUCLEUS.
- The SVC routines for the SMFWTM macro instruction - SYS1.SVCLIB.
- The SMF use definition (SMFDEFLT) - SYS1.PARMLIB.

The IBM publication System/360 Operating System: Storage Estimates (GC28-6551) shows the amount of space required for each.

HOW TO DEFINE YOUR USE OF SMF

You define your use of SMF in an SMF use definition statement. You enter or change the use definition in one of these ways:

- You may add your use definition to the parameter library (SYS1.PARMLIB) as member SMFDEFLT before or after system generation. Unless your use of SMF is only sporadic, you should do this to save having to define your use in the IPL procedure.

- In the use definition in the parameter library you may permit the operator to change the definition at IPL time. In this way, your use of SMF can be varied to meet changes in the factors that define the use of SMF.

- If you included SMF in your system generation definition but did not add your use definition to the parameter library, your operator will be asked to define your use of SMF at IPL time. If this is the case, be sure he has a copy of your use definition statement or instructions about how you plan to use SMF.

The SMF Use Definition

You define your use of SMF in the form of repeated KEYWORD=VALUE definitions in either the parameter library or in a reply from the operator at IPL time. The definitions may be stated in any order; all except the first are preceded by a comma. In the following, each use factor is shown with its definition and the type or effect of its value. Figure 1 summarizes these definitions, Figure 2 shows an illustrative definition, Figure 3 shows how to add an SMF use definition to the SYS1.PARMLIB data set.

Use Factor	Definition	Type or Effect of Value
Your System/360 Model	,MDL=nn	Your identification of the System/360 model in use.

nn - any two alphameric characters you choose to identify the System/360 model that you are using.

<u>Three-Digit Model Numbers</u>: The MDL=n value may be specified as the hexadecimal equivalent, or as any two alphameric characters. For example: Model 195 could be specified as MDL=C3 or MDL=AA.

Two characters must be shown.

Use Factor	Definition	Type or Effect of Value
Serial Identification Number	,SID=kk	User identification for different uses of SMF.

kk - Any two alphameric characters you may wish to use to distinguish different machines or other differences in your use of SMF.

Two characters must be shown.

Use Factor	Definition	Type or Effect of Value
Range of SMF Use Option	$,OPT=\left\{\begin{matrix}1\\ \underline{2}\end{matrix}\right\}$	Whether the use of SMF extends to step data.

1 - System and job data only is to be collected.
2 - System, job, and step data is to be collected.

You may omit this use factor, in which case 2 is assumed.

Use Factor	Definition	Type or Effect of Value
Change of the Use Definition by the Operator at IPL Time	,OPI=$\left\{\begin{array}{l}\text{YES}\\ \underline{\text{NO}}\end{array}\right\}$	Whether the SMF use definition can be changed by the operator at: IPL time, START TS time, or MODIFY TS time.

YES - The operator is to be allowed (by means of message IEE357A) to change the SMF use definition at IPL, or the operator is to be allowed (by means of the SMF= keyword) to change the SMF use definition at START TS time or at MODIFY TS time.

No - No change in the use definition by the operator is to be allowed.

This use factor may only be defined in the SMFDEFLT member of the SYS1.PARMLIB data set (not by the operator himself at IPL time). You may omit this use factor from SMFDEFLT, in which case NO is assumed.

Illustrations of Use Definitions

Figure SMF 1 summarizes the use factors to be described, the keywords and values used to define them, and the type or effect of the values. Figure SMF 2 shows two forms of a use definition statement. One form shows the use definition in columns 1 through 71 of a data statement for use with the IEBUPDTE program. The other form shows the use definition between apostrophes ('...') for use in an operator reply at IPL time. Figure SMF 3 shows the job control statements necessary to use the IEBUPDTE program to add a use definition to the parameter library.

Use Factor	Key-word	Value	R,D* (1)	Type or Effect of Value
Your System/360 Model	,MDL=	nn	R	Any two alphameric characters
User Serial Identification Number	,SID=	kk	R	2 alphameric characters
Range of SMF Use Option	,OPT=	1		System and Job data only
		2	D	System, Job, and Step data
Change of Use Definition by Operator at IPL time	,OPI=	YES		Definition may be changed
		NO	D	Definition may not be changed
Data Set and Volume Activity Recording	,DSV=	0	D	Record neither
		1		Record volume activity
		2		Record data set activity
		3		Record both
Data Set Activity Recording Extension	,REC=	0	D	None
		1		Include error statistics by volume
		2		Include scratching of temporary data sets
		3		Include both
Use of User Exits	,EXT=	NO		No user exits are to be taken
		YES	D	User exits are to be taken
Job Wait Time (Minutes)	,JWT=	n	R	1 -3 decimal numbers
Use of SMF Data Set	,MAN=	NONE		Not to be used at all
		USER		To be used for user records only
		ALL	D	To be used for SMF and user records
Size of Buffer for SMF Data Set	,BUF=	m	R	A number, from 400 to 65532
SMF Data Set Definition				
Direct Access Storage				
First Data Set (SYS1.MANY)	,PRM=	no	R	Volume Serial Number
		,address		Address of Device
Second Data Set (SYS1.MANX)	,ALT=	no	R	Volume Serial Number
		,address		Address of Device
Magnetic Tape				
Data Set (SYS1.MANX)	,PRM=	no	R	Volume Serial Number
		,address		Address of Device
		,lbl		Label Processing

* R - Required keyword
D - Default value

•Figure SMF 1. SMF Use Definition

SMF Use Definition in Form of a Data Statement for Use with IEBUPDTE Program

```
                                                                          Col.
                                                                           72
     MDL=50,SID=2B,OPT=2,DSV=3,REC=3,OPI=YES,EXT=YES,                      X
            JWT=30,MAN=ALL,BUF=2048,PRM=123456,ALT=123456
```

SMF Use Definition in Form of Operator Reply Text to a Define SMF Message at IPL Time

```
'MDL=50,SID=2B,OPT=2,DSV=3,REC=3,EXT=YES,JWT=30,MAN=ALL,BUF=2048,PRM=123456,ALT=123456'
```

Figure SMF 2. Sample SMF Use Definition

```
//...          JOB        ...
//...          EXEC       PGM=IEBUPDTE
//SYSPRINT     DD         SYSOUT=A
//SYSUT1       DD         DSNAME=SYS1.PARMLIB,DISP=(OLD,KEEP)
//SYSUT2       DD         DSNAME=SYS1.PARMLIB,DISP=(OLD,KEEP)
//SYSIN        DD         DATA
./ ADD LIST=ALL,NAME=SMFDEFLT,LEVEL=01,SOURCE=0

            IEBUPDTE Data Statements

            (See Figure SMF 2)

/*
```

Figure SMF 3. JCL Statements Needed to Add SMFDEFLT to SYS1.PARMLIB

Notes on the use of fields:

51 Completion Code.

gh hh - 4 hexadecimal digits in a 2-byte field taken from the 8-digit field of the TCBCMP when the job was abnormally terminated, or from register 15 when the job terminated normally.

g - Flag Digit.

0 - hhh is a completion code indicating abnormal termination by the control program (in this case gh hh are digits 1,2,3,4 of the TCBCMP), or hhh is a user-established condition code taken from register 15 when the job terminated normally and returned control to the control program.

8 - hhh is a completion code indicating abnormal termination by the user (in this case gh hh are digits 4,5,6,7 of the TCBCMP).

You can determine whether the job ended normally or was abnormally terminated by looking at bits 5 and 6 in the field for the job's end condition starting at byte 62 of this record.

53 Job Priority.

Normally, the user assigned priority (0 - 13). If the job fails while being scheduled, for example, during device allocation, this field shows a priority of 14, reflecting ABEND processing.

62 Job's Ending Condition. Job ended during:

1...	SYSOUT Limit Expiration (IEFUSO Exit)
.1..	JCL Validation (IEFUJV Exit)
..1.	Job Initiation (IEFUJI Exit)
...1	Step Initiation (IEFUSI Exit)
.... 1...	Step Ending (IEFACTRT Exit)
.... .1..	Time Limit Expiration (IEFUTL Exit)
.... ..0.	Normal Ending
.... ..1.	Abnormal Ending
....x	A Reserved Bit

68 Flag field.

1...	System restart
.xx. ..xx	(Reserved bits)
...1	A checkpoint was taken for a step of this job.
.... 1...	A checkpoint restart occurred for this job.
.... .1..	A step restart occurred for this job.

113 CPU time used by the job.

Time used for the problem program by the CPU between job initiation and job termination. Includes time used by, for example, the supervisor program. Excludes time used by, for example, the scheduler, reader and writer programs.

0	.b.	Reserved - Zero
1	.b.	Record Type: 4
2	.b.	Time of End of Step
6	.p.	Date of End of Step
10	.e.	User Serial Identification No.
12	.e.	System Model No.
14	.e.	Job Name
22	.b.	Job Time Stamp
26	.p.	Job Date Stamp
30	.e.	SMF Record User Data
38	.b.	Step No.
39	.b.	Time of Start of Step
43	.p.	Date of Start of Step
47	.b.	No. of User Data Cards in SYSIN
51	.b.	Step Completion Code
53	.b.	Step Priority (0-255)
54	.e.	Name of Executed Program
62	.e.	Name of Executed Step
70	.b.	No. of Blocks of 1K Bytes of Requested Processor (Hierarchy 0) Storage Region or Partition
72	.b.	No. of Blocks of 1K Bytes of Requested Hierarchy 1 Storage Region or Partition
74	.b.	No. of Blocks of 1K Bytes of Obtained Processor (Hierarchy 0) Storage Region or Partition
76	.b.	No. of Blocks of 1K Bytes of Borrowed Processor (Hierarchy 0) Storage Region
78	.b.	No. of Blocks of 1K Bytes of Obtained Hierarchy 1 Storage Region or Partition
80	.b.	No. of Blocks of 1K Bytes of Borrowed Hierarchy 1 Storage Region
82		Reserved
102	.b.	Length Factor
104 +0	.b.	SMF Device Description
+4	.b.	Count of Accesses to This Device
		There is one such eight-byte entry for each data set defined by a DD statement. m
m+1	.b.	Length Factor
m+2	.b.	CPU Time Used by the Step
m+5	.b.	Length Factor
m+6		Step Accounting Fields

Figure SMF 24. Step Record (Type 4)

Notes on the use of fields:

39 Time of Start of Step
The time when the step was selected for initiation. For MVT, this time does not indicate when the step was started.

51 Completion Code.
gh hh - 4 hexadecimal digits in a 2-byte field taken from the 8-digit field of the TCBCMP when the job was abnormally terminated, or from register 15 when the job terminated normally.

g - Flag Digit.

0 - hhh is a completion code indicating abnormal termination by the control program (in this case gh hh are digits 1,2,3,4 of the TCBCMP), or hhh is a user-established condition code taken from register 15 when the job terminated normally and returned control to the control program.

8 - hhh is a completion code indicating abnormal termination by the user (in this case gh hh are digits 4,5,6,7 of the TCBCMP).

You can determine whether the job ended normally or was abnormally terminated by looking at bits 5 and 6 in the field for the job's ending condition starting at byte 62 of this record.

102 Length Factor.
Length of the following series of fields, including the length factor field, in bytes.

104
+4 Count of Accesses to this Device
For MFT, this count may vary from run to run for PGM=*.DD data sets.

m+1 Length Factor.
Total number of bytes left to the end of the record.

m+2 CPU time used by the step.
Time used for the problem program by the CPU between step initiation and step termination.

m+5 Length Factor.
Total number of fields remaining to the end of the record.

SMF User Exits

Exits in the SMF program to user routines provide you with exit data additional to the data in the SMF records as well as an opportunity to control job, step and SYSOUT processing, progress, and SMF records. This section describes the exit data, the processing you may want to do (by indicating the sample programs supplied), and how to use the SMF data set.

EXIT DATA

The SMF feature provides exits to user routines that can control and audit job, step and SYSOUT processing. If your use definition includes the EXT=YES description, these exits will be taken whether or not you have provided user routines. If there is no routine for an exit, control simply returns to the SMF program.

If you have used the OPT=1 description, there will be six job processing conditions that will cause an exit to be taken. If you have used the OPT=2 description, four step processing conditions as well as the six job processing conditions will cause an exit to be taken. In addition, if there is an OUTLIM=n entry in the DD statement for a SYSOUT data set, the SYSOUT limit expiration exit is taken.

At all exits you may write in the SMF data set (if defined), to the operator, and in your own data set. Writing in your own data set, however, entails opening and closing it during the exit. To permit you to write data of your own choosing without the penalty of conducting your own data management and delaying the processing of the job or step, SMF allows user records to be written in the SMF data set with a simple write-to-the-SMF-data-set macro instruction. Descriptions of the macro instruction and the format of the SMF user record follow the exit descriptions.

The exits and the names of the routines that gain control are enumerated in the following. To add your routines to your system before system generation, add them to the SYS1.CI505 data set. To add your routines after system generation, add them to the SYS1.LINKLIB data set unless you are using MFT. If you are using MFT, two of these routines (IEFUTL and IEFUSO) must be added to SYS1.NUCLEUS(IEANUC)1).

Exit data common to all SMF exits is in the SMF Common Exit Table. The contents and format of this table is shown in Figure 25.

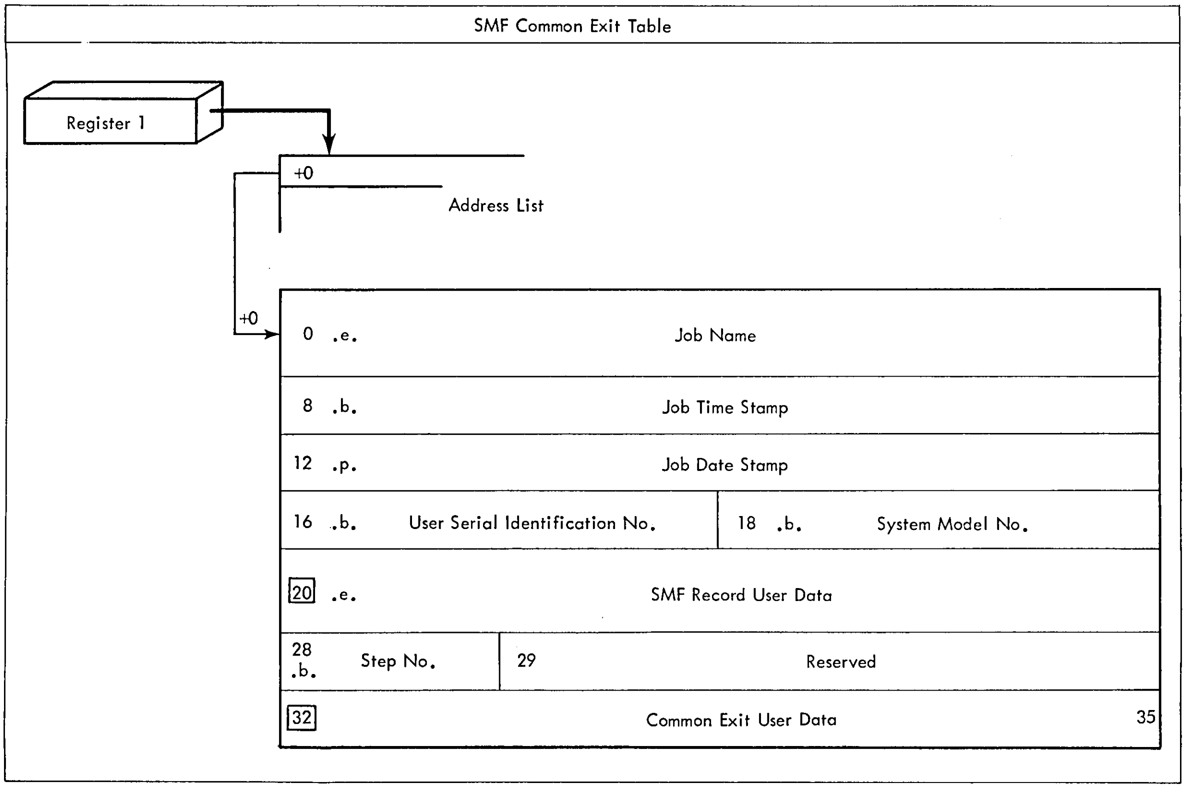


Figure SMF 25. SMF Common Exit Table

Notes on the use of fields:

20 SMF places this data in all subsequent records for this job.

32 Data accessible to the user at every subsequent exit. (This data is not used by SMF.)

OPT=1 EXITS

If your SMF use definition includes OPT=1 and EXT=YES, the exits and the names of your user routines that gain control are:

IEFUJV - The JCL Statements Validation Exit.
An SMF exit in the OS reader/interpreter routine.

Exit Conditions:

- Before interpreting a JCL statement (except a null statement). Commands in the system input stream do not cause an exit. A PROC= statement is presented before the procedure statements called for. A statement that overrides one in a procedure is presented immediately before the one it overrides.

- Before enqueueing the job.

The information available and the return codes expected are shown in Figures SMF 25 and 26.

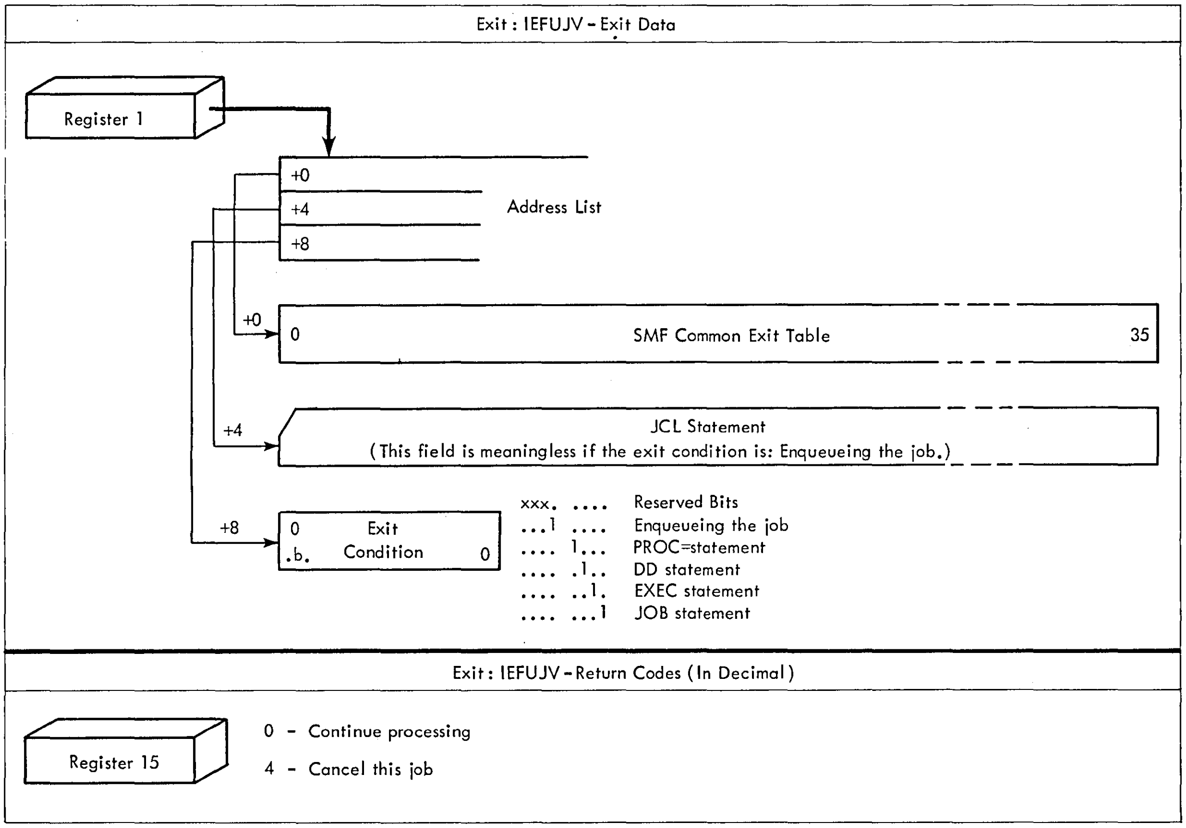


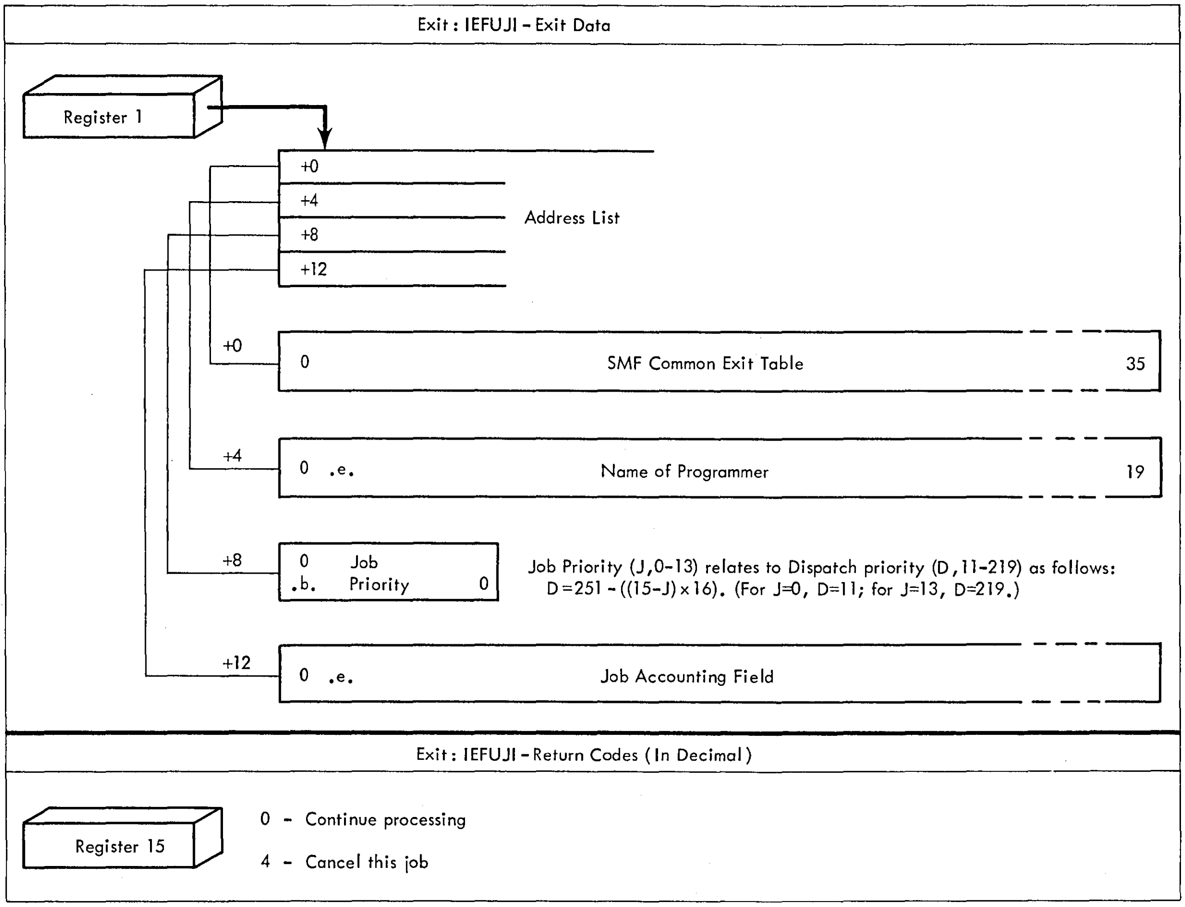
Figure SMF 26. IEFUJV Exit Data and Return Codes

IEFUJI - The Job Initiation Exit.
This is an SMF exit in the initiator/terminator routine.

Exit Condition:

- Selection of a job for initiation.

The information available and the return codes expected are shown in Figures SMF 25 and 27.



Job Priority (J,0-13) relates to Dispatch priority (D,11-219) as follows: D=251-((15-J)×16). (For J=0, D=11; for J=13, D=219.)

0 - Continue processing

4 - Cancel this job

Figure SMF 27. IEFUJI Exit Data and Return Codes

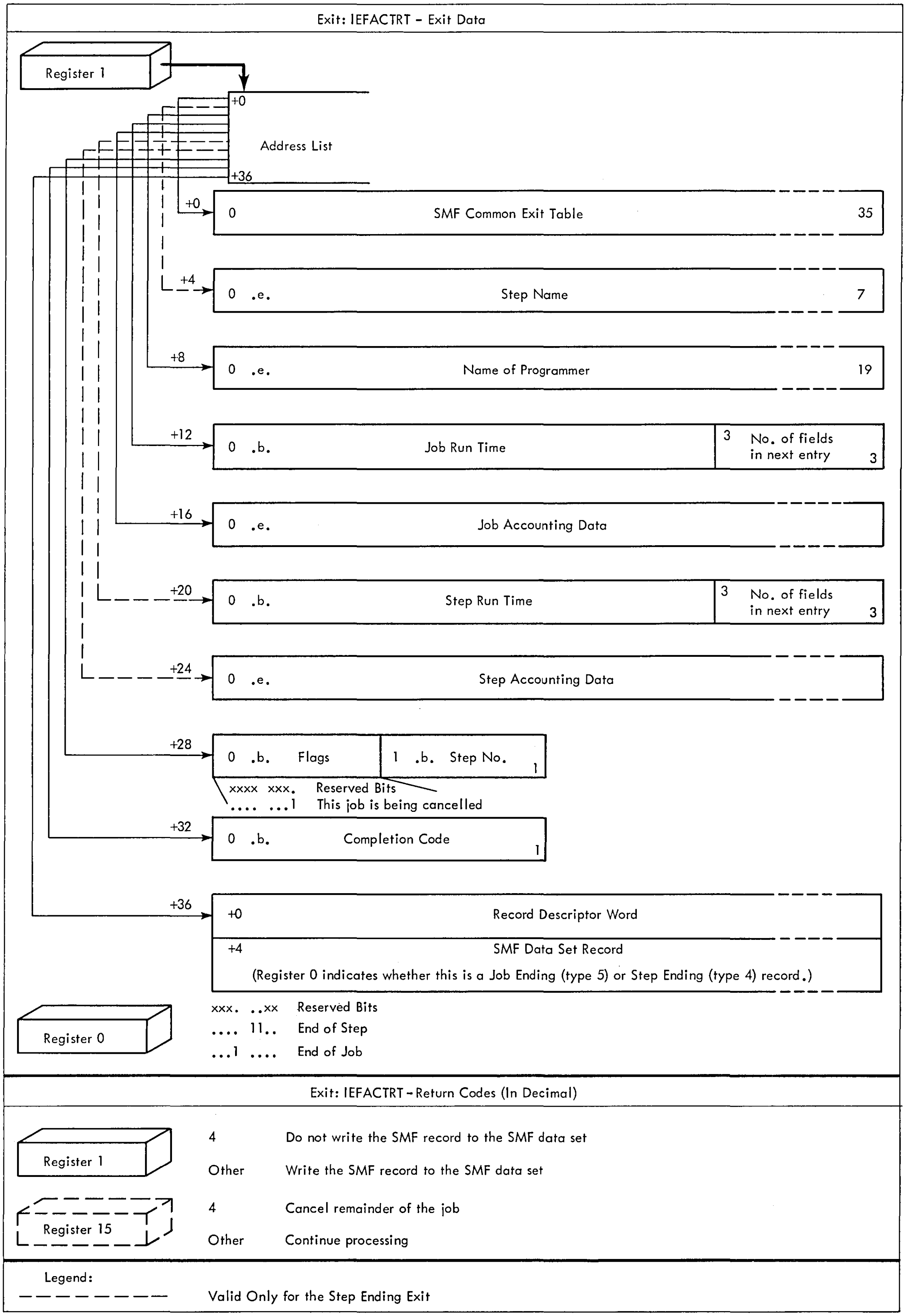
IEFACTRT - The Job Termination Exit.
This is an SMF exit in the initiator/terminator routine.

Exit Condition:

- End of processing of job by OS.

An already existing user accounting routine of this name can be used.

The information available and the return codes expected are shown in Figures SMF 25 and 28.



• **Figure SMF 28. IEFACTRT Exit Data and Return Codes**

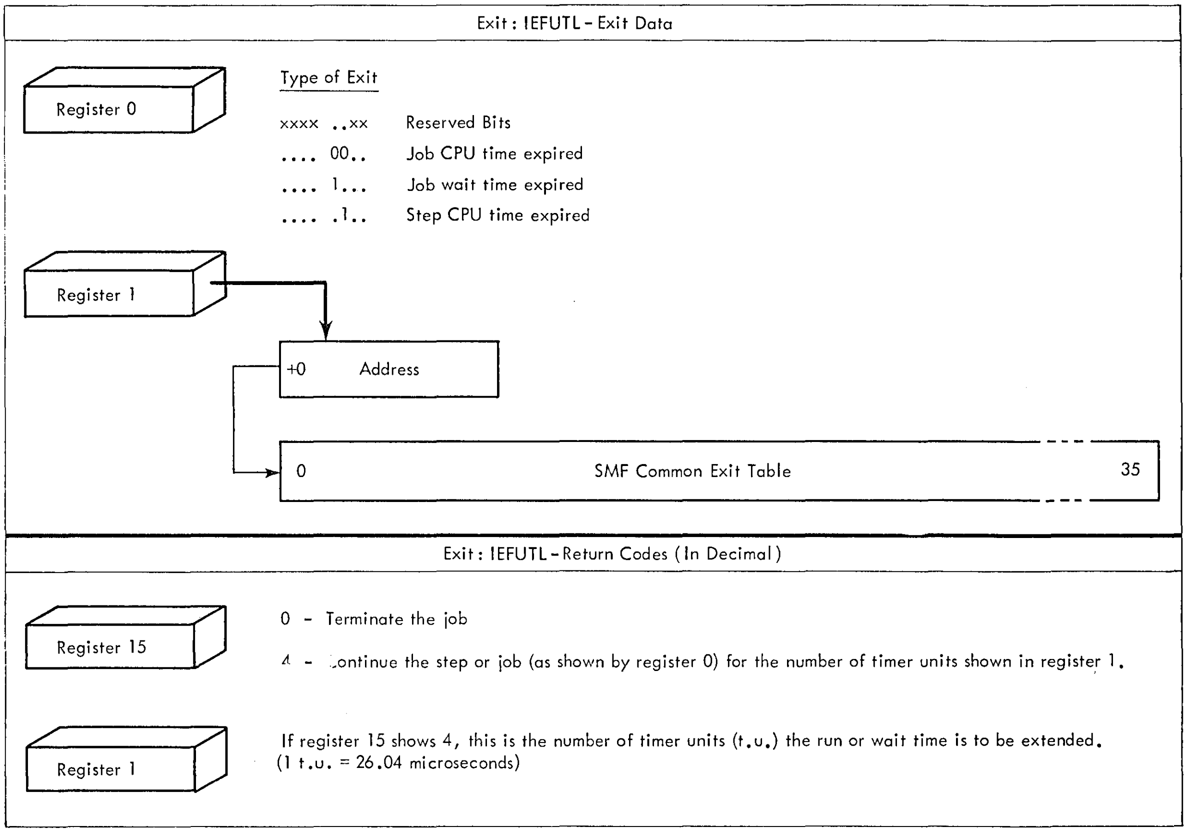
IEFUTL - The Time Limit Exit.
This is an SMF exit in the supervisor routine.

Exit Conditions:

- Expiration of the job CPU time (defined in the TIME parameter entry in the JOB statement).

- Expiration of the job wait time (defined in the SMF use definition).

The information available and the return codes expected are shown in Figures SMF 25 and 29. Note that the system wait time record (type 1) is not available at this exit.



Exit: IEFUTL - Exit Data

Type of Exit

xxxx ..xx	Reserved Bits
.... 00..	Job CPU time expired
.... 1...	Job wait time expired
.... .1..	Step CPU time expired

Exit: IEFUTL - Return Codes (In Decimal)

0 - Terminate the job

4 - Continue the step or job (as shown by register 0) for the number of timer units shown in register 1.

If register 15 shows 4, this is the number of timer units (t.u.) the run or wait time is to be extended. (1 t.u. = 26.04 microseconds)

•Figure SMF 29. IEFUTL Exit Data and Return Codes

OPT=2 EXITS

If your SMF used definition includes OPT=2 and EXT=YES, the following exits and user routines gain control in addition to those for OPT=1:

IEFUSI - The Step Initiation Exit.
This is an SMF exit in the initiator/terminator routine.

Exit Condition:

- Initiation of first and later steps.

The information available and the return codes expected are shown in Figures SMF 25 and 30.

The exits and corresponding procedures are shown in the following:

To use a user data set in this user routine:	Add a DD statement for the user data set to this procedure:
IEFUJV	Cataloged reader/interpreter procedure[1]
IEFUJI, IEFACTRT, IEFUTL, IEFUSI, IEFUSO	Cataloged initiator/terminator procedure. (MVT only)

[1] If you are using SMF with RJE, add the DD statement to the RJE procedure.

If you are using SMF with GJP, add the DD statement to the GJP procedure.

HOW TO WRITE USER RECORDS INTO THE SMF DATA SET

To write a user record in the SMF data set during a user exit, you use the SMFWTM macro instruction. Its format is:

[label]	SMFWTM	address (R)

address
    Address of the user record to be written.

(R)
    Register containing the address of the user record to be written. Unless this is register 1, the macro instruction will reload the address into that register.

Registers 15, 0, and 1 are used by SMF in the execution of the macro instruction without restoring them.

Note: The macro instruction is ineffective in routines other than SMF exit routines or a user-written SYSOUT writer routine.

To be compatible with other records in the SMF data set, the record to be written should be of the format shown in Figure SMF 35 , with the shown heading fields. Figure SMF 32 also shows the return codes the SMF program uses to show execution of the macro instruction.

SMFWTM Macro Instruction - User Record

Halfword boundary

0 .b. Length of Record		2 .b. Zeroes
4 .b. Zeroes	5 .b. Record Type (128-255)	6 .b. Time of Day
		10 .p. Date
		14 .e. User Serial No.
16 .e. System Model No.		
User Data		

SMFWTM Macro Instruction - Return Codes

Register 15

Return Codes from SMF to User Routines

xxx. ..x.	Reserved Bits
...1	Record not written because SMF data set is full or use of SMF was defined as MAN=NONE.
.... 1...	Record not written because length is less than 14.
.... 11..	Record not written because protection key is not zero.
.... .1..	Record was written but truncated to the length of the space in the empty SMF data set.
....0	Record written without error.

• Figure SMF 32. User Record

Notes on the use of fields:

0 Length, including the length field. Cannot be more than 31999. Lengths greater than one half the SMF buffer size will cause segmentation, that is, the buffer will be written in two blocks.

5 User Record Type Number.
Use a type number from 128 to 255.

6, 10
Time of day, date.
To parallel the contents of the SMF type records, use the TIME BIN macro instruction to obtain the time and date of recording and place it here. Alternately you may wish to use the job time and date stamp available in the SMF common exit table.

14, 16
User serial number, system model number.
This data may be copied from corresponding fields (+16, +18) in the SMF common exit table.

HOW TO ADD YOUR SMF USER ROUTINES TO YOUR SYSTEM

You add your SMF user routines either to the SYS1.CI505 data set before system generation (for MFT and MVT), or to the linkage library (the SYS1.LINKLIB data set) after system generation (for MVT). To add your user routines after system generation in MFT, routines IEFUTL and IEFUSO must be added to SYS1.NUCLEUS, the others to SYS1.LINKLIB. In either case, you use the linkage editor to add object modules of your routines. Figure SMF 33 shows the JCL statements needed for the linkage editor program. Note that because of differing PARM= entries, user routines IEFUJV and IEFUTL must be added to the linkage library in a different job step than other SMF user routines.

Figure SMF 34 shows the linkage editor data statements for adding the routines before system generation. Figure SMF 35 shows those needed after system generation.

A set of illustrative user routines can be found in the SYS1.SAMPLIB data set. Also in the same data set is an SMF exit simulator program, called TESTEXIT, that you can use to test your routines before you add them to the system.

```
//...          JOB      ...
//...          EXEC     PGM=IEWL,PARM=(LET,LIST,RENT,NCAL,itema,itemb)

//SYSPRINT     DD       SYSOUT=A
//SYSLMOD      DD       DSNAME=data set name,DISP=(OLD,KEEP)
//SYSUT1       DD       UNIT=SYSDA,DISP=(,DELETE),SPACE=(TRK,(20,5))
//SYSLIN       DD       *

Linkage editor data statements
/*
```

Figure SMF 33. JCL Statements for Using Linkage Editor to Add SMF User Routines

Entries in capital letters
    Code as shown.

    datasetname
        For use with the system generation data set use: SYS1.CI505.
        For use with the linkage library use: SYS1.LINKLIB.

    ,itema
        For adding to SYS1.CI505: ,DC.
        For adding to SYS1.LINKLIB: ,XREF.

    ,itemb
        For adding IEFUJV and IEFUTL to SYS1.LINKLIB: ,REFR.
        For all other conditions: Omit.

You need the following sequence of statements for eacn routine to be added to SYS1.CI505:

```
Object Deck of the User Routine
NAME routine name (R)
```

Figure SMF 34. Linkage Editor Data Statements Needed to Add User Routines to the SYS1.CI505 Data Set

NAME
    Code as shown.

routinename
    Name of the user routine to be added.

(R)
    Code as shown.
    Replacement operation. Your routine replaces a dummy routine that returns control to the SMF program.

Your input to the linkage editor program is as many sequences of object decks and NAME statements as there are routines to be added.

You need the following sequence of statements for each routine to be added to SYS1.LINKLIB:

```
Object Deck of the Routine
INCLUDE SYSLMOD (load module name)
ENTRY entry point name
ALIAS alias name
NAME load module name (R)
```

Figure SMF 35. Linkage Editor Data Statements to Add SMF User Routines to SYS1.LINKLIB

Entries in capital letters
    Code as shown.

entrypointnamer, aliasname, loadmodulename
    The name of the load module into which the object deck is to be edited and its entry point and alias name.

For these user routines	The entry point and load module name is	The alias name is
for MVT		
IEFUJV	IEFUJV	none
IEFUJI, IEFUSI, IEFACTRT	IEFSD061	IEFSD104, IEFSD065, IEFW42SD, IEFV4221
IEFUTL, IEFUSO	IEFSD263	none
for MFT		
IEFUJV	IEFUJV	none
IEFUJI, IEFUSI	IEFSD512	none
IEFACTRT	IEFSD510	GO, IEFSD511, IEFW42SD, IEFSD516, IEFV4221
IEFUTI, IEFUSO	IEANUC01 (*)	none

*After adding your IEFUTL or IEFUSO routines to the MFT nucleus, you must rename the new nucleus SYS1.NUCLEUS and scratch the old one.

IEFUCBOB Macro Instruction

This macro instruction defines the symbolic names of all fields in the unit control block (UCB). Code this macro instruction with blank name and operand fields, and precede it with a DSECT statement.

Name	Operation	Operand
	IEFUCBOB	

Control Statements Required

```
//jobname      JOB     {parameters}
//stepname     EXEC    PGM=IEBUPDTE,PARM=NEW
//SYSPRINT     DD      SYSOUT=A
//SYSUT2       DD      DSNAME=SYS1.MACLIB,DISP=OLD
//SYSIN        DD      DATA
./       ADD   NAME=IEFUCBOB,LIST=ALL
               .
               .
               .
               IEFUCBOB Macro Definition
               .
               .
               .
./       ENDUP
/*
```

IEFUCBOB Macro Definition

```
          MACRO
          IEFUCBOB
UCBOB     EQU *                 UNIT CONTROL BLOCKS
          DS    0F
SRTEJBNR  DS    XL1                  JOB INTERNAL NUMBER
SRTECHAN  DS    XL1             ALLOC.CHANNEL MASK
UCBID     DS    XL1             UCB IDENTIFICATION
SRTESTAT  DS    XL1             STATUS BITS
SRTEONLI  EQU   128                ONLINE
SRTECHGS  EQU   64                 CHANGE ONLINE/OFFLINE
SRTERESV  EQU   32                 RESERVED DEVICE
SRTEUNLD  EQU   16                 UNLOAD THIS DEVICE
SRTEALOC  EQU   8                  BIT 4 ALLOCATED
SRTEPRES  EQU   4                  BIT 5 PERMANENTLY RESIDENT
SRTESYSR  EQU   2                  BIT 6 SYSRES, OR
*                                       PRIMARY CONSOLE
SRTEDADI  EQU   1                  BIT 7 DADSM INTERLOCK, OR
*                                       TAPE CONTAINS STANDARD LABELS, OR
*                                       ALTERNATE CONSOLE
UCBCHA    DS    XL1             FLAG1 AND CHANNEL ADDRESS
UCBUA     DS    XL1             UNIT ADDRESS
UCBFL2    DS    XL1             FLAG2
UCBDTI    DS    XL1             DEVICE TABLE
UCBETI    DS    XL1             ERROR TABLE
UCBSTI    DS    XL1             STATUS TABLE
UCBLCI    DS    XL1             LOGICAL CHANNEL TABLE
UCBATI    DS    XL1             ATTENTION TABLE
UCBWGT    DS    XL1             WEIGHT
```

```
UCBNAME   DS    CL3        UNIT NAME IN 3 EBCDIC CHARACTERS
UCBTYP    DS    XL4        DEVICE TYPE
UCBTBYT1  EQU   UCBTYP        BYTE 1 OF UCBTYPE-MODEL
UCB1FEA0  EQU   128             BIT 0 OF OPTION FIELD
UCB1FEA1  EQU   64              BIT 1 OF OPTION FIELD
UCB1FEA2  EQU   32              BIT 2 OF OPTION FIELD
UCB1FEA3  EQU   16              BIT 3 OF OPTION FIELD
UCB1FEA4  EQU   8               BIT 4 OF OPTION FIELD
UCB1FEA5  EQU   4               BIT 5 OF OPTION FIELD
UCB1FEA6  EQU   2               BIT 6 OF OPTION FIELD
UCB1FEA7  EQU   1               BIT 7 OF OPTION FIELD
UCBTBYT2  EQU   UCBTYP+1      BYTE 2 OF UCBTYPE-OPTIONS
UCBTBYT3  EQU   UCBTYP+2      BYTE 3 OF UCBTYPE-CLASS
UCB3TAPE  EQU   128             BIT 0 OF CLASS - TAPE
UCB3COMM  EQU   64              BIT 1 OF CLASS - COMMUNIC.
UCB3DACC  EQU   32              BIT 2 OF CLASS - DIRECT AC
UCB3DISP  EQU   16              BIT 3 OF CLASS - DISPLAY
UCB3UREC  EQU   8               BIT 4 OF CLASS - UNIT REC.
UCB3CHAR  EQU   4               BIT 5 OF CLASS - CHAR.READ
UCBTBYT4  EQU   UCBTYP+3      BYTE 4 OF UCBTYPE-DEVICE
UCBLTS    DS    XL2        LAST 12*
UCBSNS    DS    XL6        SENSE INFORMATION
UCBUCSID  EQU   UCBSNS+2   UCS CHARACTER SET-ID
SRTEVOLI  DS    CL6        VOLUME SERIAL
UCBUCSOP  EQU   UCBSNS+6   UCS OPTIONS
UCBUCSO1  EQU   128        DEFAULT CHARACTER SET
UCBUCSO2  EQU   64         BUFFER LOADED in FOLD MODE
SRTESTAB  DS    XL1        STATUS B
SRTEBSVL  EQU   128             BIT 0 SHARED VOLUME
SRTEBVSC  EQU   64              BIT 1 NOT USED
SRTEBALB  EQU   32              BIT 2 ADDIT.VOL.LABEL PROC
SRTEBPRV  EQU   16              BIT 3 PRIVATE
SRTEBPUB  EQU   8               BIT 4 PUBLIC
SRTEBVQS  EQU   4               BIT 5 STORAGE FOR DIRECT
*                                  ACCESS
STREASCI  EQU   SRTEVBQS        BIT 5 AMERICAN  NATIONAL STANDARD LABEL
                                   FOR TAPE DATA SETS
SRTEBJLB  EQU   2               BIT 6 JOBLIB VOLUME
SRTEBNUL  EQU   1               BIT 7 CONTROL VOLUME
SRTEDMCT  DS    XL1        DATA MANAGEMENT COUNT
SRTEFSCT  DS    XL2        FILE SEQ. COUNT
SRTEFSEQ  DS    XL2        FILE SEQ. NUMBER
UCBSQC    DS    2F         SEEK QUEUE CONTROL WORD
UCBSKA    DS    2F         MBBCCHHR FOR LAST SEEK
SRTEUSER  DS    XL1        CURRENT NUMBER OF USERS
SRTEECBA  DS    XL3        DA ECB ADDRESS

*THE FOLLOWING DESCRIBES ONE OF THE 10 SUB-UCBS  FOR THE 2321--

          ORG   SRTEUSER
DATACELL  DS    0CL16      10 OF THESE ARE PRESENT FOR 2321
DCELBBNR  DS    XL2        BIN NUMBER
DCELSTAB  DS    X          STATUS B
DCELSTAT  DS    X          STATUS A
DCELVOLI  DS    CL6        VOLUME SERIAL NUMBER
DCELJBNR  DS    X          INTERNAL JOB NUMBER
DCELDMCT  DS    X          DATA MANAGEMENT COUNT
DCELVTOC  DS    XL3        TTR OF VTOC START
DCELUSER  DS    X          CURRENT NUMBER OF USERS
          MEND
```

IEFJFCBN Macro Instruction

This macro instruction defines the symbolic names of all fields in the job file control block (JFCB). Code this macro instruction with blank name and operand fields, and precede it with a DSECT statement.

Name	Operation	Operand
	IEFJFCBN	

Control Statements Required

```
//jobname       JOB      (parameters)
//stepname      EXEC     PGM=IEBUPDTE,PARM=NEW
//SYSPRINT      DD       SYSOUT=A
//SYSUT2        DD       DSNAME=SYS1.MACLIB,DISP=OLD
//SYSIN         DD       DATA
./       ADD    NAME=IEFJFCBN,LIST=ALL
                .
                .
                .
                IEFJFCBN macro definition
                .
                .
                .
./       ENDUP
/*
```

IEFJFCBN Macro Definition

```
         MACRO
         IEFJFCBN
INFMJFCB EQU   *
JFCBDSNM DS    CL44          DATA SET NAME
JFCBELNM DS    CL8           ELEMENT NAME OR VERSION
JFCBTSDM DS    CL1           TASK SCHEDULER - DATA
*                            MANAGEMENT INTERFACE BYTE
JFCBSYSC DS    CL13          SYSTEM CODE
JFCBLTYP DS    CL1           LABEL TYPE AND USER'S-LABEL
*                            INDICATOR
JFCBUFOF DS    CL1           BUFFER OFFSET FOR TAPE DATA SETS
JFCBFLSQ DS    CL2           FILE SEQUENCE NUMBER
JFCBVLSQ DS    CL2           VOLUME SEQUENCE NUMBER
JFCBMASK DS    CL8           DATA MANAGEMENT MASK
JFCBCRDT DS    CL3           DATA SET CREATION DATE
JFCBXPDT DS    CL3           DATA SET EXPIRATION DATE
JFCBIND1 DS    CL1           INDICATOR BYTE 1
JFCBRLSE EQU   64             BITS 0 AND 1 - EXTERNAL
*                             STORAGE RELEASE INDICATOR
JFCBLOCT EQU   16             BITS 2 AND 3 - DATA SET
*                             HAS BEEN LOCATED
JFCBNEWV EQU   4              BITS 4 AND 5 - NEW VOLUME
*                             ADDED TO DATA SET
JFCBPMEM EQU   1              BITS 6 AND 7 - DATA SET IS
*                             A MEMBER OF A PODS OR GDG
JFCBIND2 DS    CL1           INDICATOR BYTE 2
```

```
JFCBSTAT EQU   64         BITS 0 AND 1 - DATA SET
*                         STATUS (NEW, OLD, OR MOD)
JFCBSCTY EQU   16         BITS 2 AND 3 - DATA SET
*                         SECURITY INDICATOR
JFCBUFNO DS    0AL1
JFCBUFRQ DS    AL1
JFCBFTEK DS    0BL1
JFCBFALN DS    BL1
JFCBUFL  DS    AL2
JFCEROPT DS    BL1
JFCTRTCH DS    0BL1
JFCKEYLE DS    0AL1
JFCMODE  DS    0BL1
JFCCODE  DS    0BL1
JFCSTACK DS    0BL1
JFCPRTSP DS    BL1
JFCDEN   DS    BL1
JFCLIMCT DS    AL3
JFCDSORG DS    BL2
JFCRECFM DS    BL1
JFCOPTCD DS    BL1
JFCBLKSI DS    AL2
JFCLRECL DS    AL2
JFCNCP   DS    AL1
JFCNTM   DS    AL1
JFCRKP   DS    AL2
JFCCYLOF DS    AL1
JFCDBUFN DS    AL1
JFCINTVL DS    AL1
JFCCPRI  DS    BL1
JFCSOWA  DS    AL2
JFCBNTCS DS    CL1       NUMBER OF OVERFLOW TRACKS
JFCBNVOL DS    CL1       NUMBER OF VOLUME SERIAL NUMBERS
JFCBVOLS DS    CL30      VOLUME SERIAL NUMBERS (THE FIRST FIVE)
JFCBEXTL DS    CL1       LENGTH OF BLOCK OF EXTRA
*                        VOLUME SERIAL NUMBERS
*                        (BEYOND FIVE)
JFCBEXAD DS    CL3       TRACK ADDRESS OF BLOCK OF
*                        EXTRA VOLUME SERIAL NUMBERS
JFCBPQTY DS    CL3       PRIMARY QUANTITY OF D.A. STORAGE REQUIRED
JFCBCTRI DS    CL1       INDICATES WHETHER CYLINDERS, TRACKS, OR RECORDS
*                        ARE SPECIFIED IN JFCBPQTY AND JFCBSQTY
JFCBSQTY DS    CL3       SECONDARY QUANTITY OF D.A. STORAGE REQUIRED
JFCBIND3 DS    CL1       INDICATOR BYTE 3
JFCBCNTG EQU   64        BITS 0 AND 1 - CONTIGUOUS STORAGE INDICATOR
JFCBMXIG EQU   16        BITS 2 AND 3 - MAXIMUM
*                        AVAILABLE EXTENT INDICATOR
JFCBALXI EQU   4         BITS 4 AND 5 - ALL EXTENTS INDICATOR
JFCBRNDC EQU   1         BITS 6 AND 7 - ROUND CYLINDER INDICATOR
JFCBDQTY DS    CL3       QUANTITY OF D.A. STORAGE
*                        REQUIRED FOR A DIRECTORY
JFCBSPNM DS    CL3       CORE ADDRESS OF THE JFCB
*                        WITH WHICH CYLINDERS ARE SPLIT
JFCBABST DS    CL2       RELATIVE ADDRESS OF FIRST
*                        TRACK TO BE ALLOCATED
JFCBSBNM DS    CL3       CORE ADDRESS OF THE JFCB
*                        FROM WHICH SPACE IS TO BE SUBALLOCATED
JFCBDRLH DS    CL3       AVERAGE DATA RECORD LENGTH
JFCBVLCT DS    CL1       VOLUME COUNT
JFCBSPTN DS    CL1       NUMBER OF TRACKS PER CYLINDER TO BE USED BY
*                        THIS DATA SET WHEN SPLIT CYLINDERS IS INDICATED
JFCBLGTH EQU 176      LENGTH OF JFCB
JFCBEND  EQU   *
         MEND
```

Data Set Protection

To use the data set protection feature of the operating system, you must create and maintain a password data set consisting of records that associate the names of the protected data sets with the password assigned to each data set. There are two ways to maintain the password data set: you can write your own routines to maintain it or you can use the facilities of the PROTECT macro instruction to maintain it.

This chapter is divided into two sections. The first section describes the general features of data set protection, including the use of your own routines to maintain the password data set. It provides the information you need to create the data set and it describes the record format and characteristics of the data set. The second section discusses the PROTECT macro, it provides the programming information you need to use the macro and it discusses the difference between using the PROTECT macro and using your own routines to maintain the password data set.

RECOMMENDED PUBLICATIONS

The IBM System/360 Operating System: Data Management Services publication (GC28-3746) contains a general description of the data set protection feature.

The IBM System/360 Operating System: Messages and Codes publication (GC28-6631) contains a description of the operator messages and replies associated with the data set protection feature.

The IBM System/360 Operating System: Job Control Language Reference publication (GC28-6704) contains a description of the data definition (DD) statement parameter used to indicate that a data set is to be placed under protection.

Documentation of the operating system routines supporting data set protection can be obtained through your IBM Branch Office.

Data Set Protection

To prepare for use of the data set protection feature of the operating system, you place a sequential data set, named PASSWORD, on the system residence volume (containing SYS1.NUCLEUS and SYS1.SVCLIB). Note: If the routines that you write to maintain the password data set use the basic direct access method (BDAM), you must place a BDAM data set named PASSWORD on the system residence volume. This data set must contain one record for each data set placed under protection. In turn, each record contains a data set name, the password for that data set, a counter field, a protection mode indicator, and a field for recording any information you desire to log. On the system residence volume, these records are formatted as a "key area" (data set name and password) and a "data area" (counter field, protection mode indicator, and logging field). The data set is searched on the "key area."

You can write routines to create and maintain the PASSWORD data set. (If you use the PROTECT macro instruction to maintain the password data set, see the section in this chapter called USING THE PROTECT MACRO INSTRUCTION TO MAINTAIN THE PASSWORD DATA SET.) These routines may be placed in your own library or the system's linkage editor library (SYS1.LINKLIB). You may use a data management access method or EXCP programming to handle the PASSWORD data set.

If a data set is to be placed under protection, it must have a protection indicator set in its label (DSCB or header 1 tape label). This is done by the operating system when the data set is created. The protection indicator is set in response to an entry in the LABEL= parameter of the DD statement associated with the data set being placed under protection. The Job Control Language Reference publication describes the entry. Note: Data sets on magnetic tape are protected only when standard labels are used.

Users who wish to have the password supplied by some method other than operator key-in may replace the password reading module with their own routine. The READPSWD source module may be used as a base for writing a new module. In this case, the new object module replaces module READPSWD on the SVCLIB.

The balance of this chapter discusses the PASSWORD data set characteristics and record format, the creation of protected data sets, and operating characteristics of the data set protection feature.

Password Data Set Characteristics and Record Format

The PASSWORD data set must reside on the same volume as your operating system. The space you allocate to the PASSWORD data set must be contiguous, i.e., its DSCB must indicate only one extent. The amount of space you allocate is dependent on the number of data sets your installation desires to place under protection. The organization of the PASSWORD data set is physical sequential, the record format is unblocked, fixed length records (RECFM=F). These records are 80 bytes long (LRECL=80) and form the data area of the PASSWORD data set records on direct access storage. In these direct access storage records, the data area is preceded by a key area of 52 bytes (KEYLEN=52). In main storage, the 52 byte key field (which contains the data set name and the password) and the 80 byte data field (whose first three bytes contain a counter and a protection indicator) together form a 132 byte buffer. Figure PSWD1 shows the password records as you would build them in a 132 byte work area. Explanation of the fields follows the illustration.

The name of the protected data set being opened and the password entered by the operator are matched against the 52-byte "key area." The data set name and the password must be left-justified in their areas and any unused bytes filled with blanks (X'40'). The password assigned may be from one to eight alphameric characters.

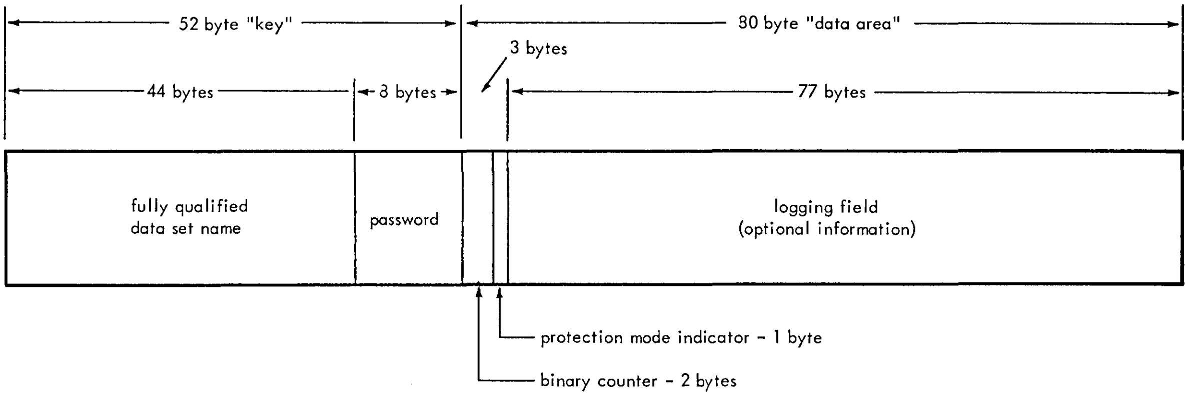

52 byte "key"
80 byte "data area"
3 bytes
44 bytes
8 bytes
77 bytes
fully qualified data set name
password
logging field (optional information)
protection mode indicator - 1 byte
binary counter - 2 bytes

Figure PSWD1. Password Record

The operating system increments the binary counter by one each time the data set is successfully opened (except for performance of SCRATCH or RENAME functions on the data set). When you originate the password record, the value in the counter may be set at zero (X'0000') or any starting value your installation desires.

The protection mode indicator is set to indicate that the data set is to be read-only, or that it may be read or written. Read only and read/write protection for a data set can be attained by including the same data set name in the password data set twice and giving it different passwords. You set the indicator as follows:

- To zero (X'00') if the data set is to be read-only.
- To one (X'01') if the data set may be read or written.

You may use the 77-byte logging field to record any information about the data set under protection that your installation may desire, e.g., date of counter reset, previous password used with this data set, etc.

Protecting the Password Data Set

You protect the PASSWORD data set itself by creating a password record for it when your program initially builds the data set. Thereafter, the PASSWORD data set cannot be opened (except by the operating system routines that scan the data set) unless the operator enters the password.

PSWD

Creating Protected Data Sets

A data definition (DD) statement parameter (LABEL=) is used to indicate that a data set is to be placed under protection. You may create a data set, and set the protection indicator in its label, without entering a password record for it in the PASSWORD data set. However, once the data set is closed, any subsequent opening results in termination of the program attempting to open the data set, unless the password record is available and the operator can honor the request for the password. Operating procedures at your installation must ensure that password records for all data sets currently under protection are entered in the PASSWORD data set.

Protection Feature Operating Characteristics

This section provides information concerning actions of the protection feature in relation to termination of processing, volume switching, data set concatenation, SCRATCH and RENAME functions, and counter maintenance.

Termination of Processing

Processing is terminated when:

1. The operator cannot supply the correct password for the protected data set being opened.

2. A password record does not exist in the PASSWORD data set for the protected data set being opened.

3. The protection mode indicator setting in the password record, and the method of I/O processing specified in the open routine do not agree, e.g., OUTPUT specified against a read-only protection mode indicator setting.

4. There is a mismatch in data set names for a data set involved in a volume switching operation. This is discussed in the next section.

Volume Switching

The operating system end-of-volume routine does not request a password for a data set involved in a volume switch. Continuity of protection is handled in the following ways:

Input Data Sets - Tape and Direct Access Devices

Processing continues if there is an equal comparison between the data set name in the tape label or DSCB on the volume switched to, and the name of the data set opened with the password. An unequal comparison terminates processing.

Output Data Sets - Tape Devices

The protection indicator in the tape label on the volume switched to is tested:

1. If the protection indicator is set ON, an equal comparison between the data set name in the label and the name of the data set opened with the password allows processing to continue. An unequal comparison results in a call for another volume.
2. If the protection indicator is OFF, processing continues, and a new label is written with the protection indicator set ON.
3. If only a volume label exists on the volume switched to, processing continues, and a new label is written with the protection indicator set on.

Output Data Sets - Direct Access Devices

For existing data sets, an equal comparison between the data set name in a DSCB on the volume switched to, and the name of the data set opened with the password allows processing to continue. For new output data sets, the mechanism used to effect volume switching ensures continuity of protection and the DSCB created on the new volume will indicate protection.

Data Set Concatenation

A password is requested for every protected data set that is involved in a concatenation of data sets, regardless of whether the other data sets involved are protected or not.

SCRATCH and RENAME Functions

An attempt to perform the SCRATCH or RENAME functions on a protected data set results in a request for the password. The protection feature issues an operator's message (IEC301A) when a protected data set is the object of these functions. The *Messages and Codes* publication discusses the message.

Counter Maintenance

The operating system does not maintain the counter in the password record and no overflow indication will be given (overflow after 65,535 openings). You must provide a counter maintenance routine to check and, if necessary, reset this counter.

Using the Protect Macro Instruction to Maintain the Password Data Set

To use the PROTECT macro instruction, your password data set should be on the system residence volume. The PROTECT macro can be used to:

- Add an entry to the password data set.
- Replace an entry in the password data set.
- Delete an entry from the password data set.
- Provide a list of information about an entry in the password data set; this list will contain the security counter, access type, and the 77 bytes of security information in the "data area" of the entry.

In addition, the PROTECT macro, will update the DSCB of the protected data set, for a direct access device, to reflect its protected status; this feature eliminates the need for you to use job control language whenever you place a data set under protection.

PASSWORD DATA SET CHARACTERISTICS AND RECORD FORMAT WHEN YOU USE THE PROTECT MACRO

When you use the PROTECT macro, the record format and characteristics of the password data set should be the same as the record format and characteristics when you use your own routines to maintain it, with two exceptions: the number of records that you establish for each protected data set and the values of the protection mode indicator.

Number of Records for Each Protected Data Set: When you use the PROTECT macro, the password data set must contain at least one record for each protected data set. The password (the last 8 bytes of the "key area") that you assign when you place the data set under protection for the first time is called the control password, in addition, you may create as many secondary records for the same protected data set as you need. The passwords assigned to these additional records are called secondary passwords. This feature is helpful if you want several users to have access to the same protected data set, but you also want to control the manner in which they can use it. For example: one user could be assigned a password that allowed the data set to be read and written, and another user could be assigned a password that allowed the data set to be read only.

PSWD

Note: The PROTECT macro will update the DSCB of the protected data set only when you issue it for adding, replacing or deleting a control password.

Protection Mode Indicator: You can set the protection mode indicator in the password record to four different values:

- X'00' to indicate that the password is a secondary password and the protected data set is to be read only.
- X'80' to indicate that the password is the control password and the protected data set is to be read only.
- X'01' to indicate that the password is a secondary password and the protected data set is to be read and written.
- X'81' to indicate the password is the control password and the protected data set is to be read and written.

Since the DSCB of the protected data set is updated only when the control password is changed, it is possible to request protection attributes for secondary passwords which conflict with the protection attributes of the control password.

If the control password has read only protection, its secondary passwords may have read only or read write protection. A request for a secondary password with read without password protection will result in a secondary password with read write protection. A read only control password may be changed to a read write control password without affecting any secondary passwords, but if a read only control password is changed to a read without password control password all secondary passwords will automatically become read without password secondary passwords.

If the control password has read write protection, its secondary passwords may have read only or read write protection. A request for a secondary password with read without password protection will result in a secondary password with read write protection. A read write control password may be changed to a read only control password without affecting any secondary passwords, but if a read write control password is changed to a read without password control password all secondary passwords will automatically become read without password secondary passwords.

If the control password has read without password protection, its secondary passwords must also have read without password protection. A request for a read only or for a read write secondary password will result in a read without password secondary password. If a read without password control password is changed to either a read only or read write control password all its secondary passwords will automatically become read write secondary passwords.

PROGRAMMING CONVENTIONS FOR THE PROTECT MACRO INSTRUCTION

The format of the PROTECT macro is:

```
        ( (1)        register 1 with the address of a parameter list      )
PROTECT < (REG)      a register with the address of a parameter list      >
        ( list addr  address of location containing the parameter list    )
```

When you issue the PROTECT macro, you should have already established the parameter list. Its size and contents depend on the function that you want the macro to perform. In any case, the first byte of the parameter list is an entry code that indicates the function:

- X'01' for adding an entry to the parameter list.

- X'02' for replacing an entry in the parameter list.

- X'03' for deleting an entry from the parameter list.

- X'04' for listing the information in a password data set entry. For a complete discussion of the contents of the parameter lists, see figures PSWD2 to PSWD5 and the notes explaining each of these figures.

PROTECT Macro Parameter Lists

The parameter lists, their formats and contents are:

PARAMETER LIST FOR ADD FUNCTION

0	X'01'	1	00 00 00
4	Data Set Length	5	Pointer to Data Set Name
8	00	9	00 00 00
12	00	13	Pointer to Control Password
16	Number of Volumes	17	Pointer to Volume List
20	Protection Code	21	Pointer to New Password
24	String Length	25	Pointer to String

Figure PSWD2. Parameter List for Add Function

Explanatory Notes for Figure PSWD2.

0 X'01'
Entry code indicating add function.

13 Pointer to control password.
The control password is the password assigned when the data set was placed under protection for the first time. This can be a string of zeros if the new password is the control password.

16 Number of volumes.
If the data set is not cataloged and you want to have it flagged as protected, you have to specify the number of volumes in this field. A zero indicates that the catalog information should be used.

17 Pointer to volume list.
If the data set is not cataloged and you want to have it flagged as protected, you provide the address of a list of volume serial numbers in this field. Zeros indicate that the catalog information should be used.

20 Protection code.
A one-byte number indicating the type of protection: X'00' indicates default protection (for the add function, the default protection is the type of protection specified in the control password record of the data set), X'01' indicates that the data set is to be read and written, X'02' indicates that the data set is to be read only and X'03' indicates that the data set can be read without a password, but a password is needed to write into it. The

PSWD

PROTECT macro will use the protection code value, specified in the parameter list, to set the protection mode indicator in the password record.

21 Pointer to new password.
If the data set is being placed under protection for the first time, the new password is the same as the control password. If you are adding a secondary entry, the new password is different from the control password.

24 String length.
The length of the character string (maximum 77 bytes) that you want to place in the optional information field of the password record. If you don't want to add information, set this field to zero.

25 Pointer to string.
The address of the character string that is going to be put in the optional information field. If you don't want to add additional information, set this field to zero.

Parameter List for Replace Function

0 X'02'	1 00 00 00
4 Data Set Length	5 Pointer to Data Set Name
8 00	9 Current Password
12 00	13 Pointer to Control Password
16 Number of Volumes	17 Pointer to Volume List
20 Protection Code	21 Pointer to New Password
24 String Length	25 Pointer to String

Figure PSWD3. Parameter List for Replace Function

Explanatory Notes for Figure PSWD3.

0 X'02'
Entry code indicating REPLACE function

9 Pointer to current password.
The address of the password that is going to be replaced.

13 Pointer to control password.
The address of the password assigned to the data set when it was first placed under protection. This can be zero if the current password is the control password.

16 Number of volumes.
If the data set is not cataloged and you want to have it flagged as protected, you have to specify the number of volumes in this field. A zero indicates that the catalog information should be used.

17 Pointer to volume list.
If the data set is not cataloged and you want to have it flagged as protected, you have to provide the address of a list of volume serial numbers in this field. If this field is zero, the catalog information will be used.

20 Protection code.
A one-byte number indicating the type of protection: X'00' indicates that the protection is default protection (for the replace

function the default protection is the protection specified in the current password record of the data set), X'01' indicates that the data set is to be read and written, X'02' indicates that the data set is to be read only, and X'03' indicates that the data set can be read without a password, but a password is needed to write into the data set.

21 Pointer to new password.
The address of the password that you want to replace the current password.

24 String length.
The length of the character string (maximum 77 bytes) that you want to place in the optional information field of the password record. Set this field to zero if you don't want to add additional information.

25 Pointer to string.
The address of the character string that is going to be put in the optional information field of the password record. Set the address to zero if you don't want to add additional information.

Parameter List for Delete Function

0 X'03'	1 00 00 00
4 Data Set Length	5 Pointer to Data Set Name
8 00	9 Pointer to Current Password
12 00	13 Pointer to Control Password
16 Number of Volumes	17 Pointer to Volume List

Figure PSWD4. Parameter List for Delete Function

Explanatory Notes for Figure PSWD4.

0 X'03'.
Entry code indicating delete function.

9 Pointer to current password.
The address of the password that you want to delete. You can delete either a control entry or a secondary entry.

13 Pointer to control password.
The address of the password assigned to the data set when it was placed under protection for the first time. This can be zeros if the current password is also the control password.

16 Number of volumes.
If the data set is not cataloged and you want to have it flagged as protected, you have to specify the number of volumes in this field. A zero indicates that the catalog information should be used.

17 Pointer to volume list.
If the data set is not cataloged and you want to have it flagged as protected, you have to provide the address of a list of volume serial numbers in this field. If this field is zero, the catalog information will be used.

Parameter List for List Function

0 X'04'	1 Address of 80 Byte Buffer
4 Data Set Length	5 Address of Data Set Name
8 00	9 Pointer to Current Password Name

Figure PSWD5. Parameter List for List Function

Explanatory notes for using Figure PSWD5.

0 X'04'.
Entry code indicating list function.

1 Address of 80-byte buffer.
The address of a buffer where the list of information can be returned to your program by the macro instruction.

9 Pointer to current password name.
The address of the password of the record that you want listed.

Return Codes from the PROTECT Macro

When the PROTECT macro finished processing, register 15 will contain a return code that indicates what happened during the processing. Table PASS1 contains the return codes and their explanations.

Table PASS1. Return Codes from The PROTECT Macro

Register 15	Explanation
0	The updating of the password data set was successfully completed.
4	The password of the data set name was already in the password data set.
8	The password of the data set name was not in the password data set.
12	A control password is required or the one supplied is incorrect.
16	The supplied parameter list was incomplete or incorrect.
20	There was an I/O error in the password data set.
**24	The password data set was full.
28	The validity check of the buffer address failed.
*32	The LOCATE macro failed. LOCATE's return code is in register 1 and the number of indexes searched is in register 0.
*36	The OBTAIN macro failed. OBTAIN's return code is in register 1.
*40	The DSCB could not be updated.
44	The password data set does not exist.
*48	Tape data set can not be protected.

*For these return codes, the password data set has been updated, but the DSCB has not been flagged to indicate the protected status of the data set.

**For this return code, a message is written to the console indicating that the password data set is full.

The device type is defined by:

A four digit number designating the type of direct access device on which the volume resides, e.g., the IBM 2311 Disk Storage Drive is indicated by the notation 2311. Note that this field only indicates the basic device type for the associated volume. You must advise the operator if the device requires special features (such as track overflow) to process the data on the designated volume.

The mount priority field is used to suppress mount messages at IPL time for a volume; the alphabetic character N should be inserted in this field to suppress the mount message. This field allows the user to list seldom used volumes in the PRESRES list without having a mount message issued at each IPL. When these volumes are required, they may be mounted and attributes will be set from the PRESRES list entry. If the user does not wish to have the mount message suppressed, he may omit the mount priority field and the preceding comma.

The optional information field contains:

Any descriptive information about the volume that you may wish to enter. This information is not used by the system, but will be available to you on a printout of the list. If necessary, comments may start in the second byte after the mount priority field or if the mount priority field is omitted, in the second byte following the comma after the device type field.

Embedded blanks are not permitted in the volume serial, mount, allocation, or device type fields.

Operational Characteristics

Upon receiving control from the nucleus initialization program (NIP), the scheduler compares the volume serial numbers in the PRESRES characteristics list with those of currently mounted direct access volumes. Each equal comparison results in the assignment to the mounted volume of the characteristics noted in the PRESRES entry. (Fields in the unit control block for the device on which the volume is mounted are set to reflect the desired characteristics.) If the volume is: the IPL volume; the volume containing the data sets SYS1.LINKLIB, SYS1.PROCLIB, SYS1.SYSJOBQE; or a physically nondemountable volume (such as a 2301 drum storage unit), the mount characteristic (permanently resident) has already been assigned and only the allocation characteristic is set.

A mounting list is issued for the volumes in the PRESRES characteristics list that are not currently mounted (except those for which mounting messages have been suppressed) and the operator is given the option of mounting none, some, or all of the volumes listed. The mount and allocation characteristics for the volumes mounted by the operator are set according to the PRESRES list entry for the volume. The operator selects the unit on which the volume is to be mounted.

The Messages and Codes publication describes the operator messages and responses associated with the use of the PRESRES volume characteristics list.

After the scheduler has finished PRESRES processing reading of the job input stream begins, and the PRESRES list is not referred to again until the next IPL.

Volume characteristics assigned by a PRESRES list entry are inviolate. They cannot be altered by subsequent references to the volume in the input stream.

Note:

1. A PRESRES entry identifying a physically nondemountable volume will appear in the mount list issued to the operator if the volume (device) is OFFLINE or is not present in the system.

2. Use of the PRESRES list can only be suppressed by deleting the member from the parameter library (SYS1.PARMLIB).

3. Only the first 102 volumes on the PRESRES list can be placed on the mount list.

Programming Considerations

The only way to assign an allocation characteristic other than "public" to volumes whose mount characteristic is "permanently resident" is through a PRESRES characteristic list entry.

Selection of the volumes for which PRESRES entries are to be created should be done so that critical volumes are protected. Since the combination of mount and allocation characteristics assigned to a specific volume determine the types of data sets that can be placed on the volume and its usage, you can exercise effective control over the volume through a PRESRES list entry.

Resident Routines Options

The resident routines options are the BLDL feature, the resident reenterable modules feature, and the RSVC and RERP features. These features permit preloading into main storage routines (or at least their addresses) that otherwise would be repeatedly loaded each time the routines are requested. The Link list feature, also described in this chapter, permits references to the Link library to be extended to other data sets. Figures RRO 1, 2, and 3 describe all these features.

There are three sections to this chapter. Section 1 discusses the PCP and MFT use of the features, section 2 the MVT use, and section 3 the Link list feature.

Section 1 of this chapter discusses the BLDL Table, reenterable modules, and RSVC and RERP and provides guidelines for their use. The purpose of these options is to improve performance by reducing or eliminating the access time required to obtain the routines with which these options are concerned. You may incorporate these options in the PCP or MFT configurations of the operating system.

Section 2 of this chapter discusses the inclusion of SVC routines, reenterable load modules, and linkage library directory entries in the Link Pack Area of the MVT configuration of the operating system.

Section 3 of this chapter discusses the link library list and provides guidelines for its use. The purpose of the link library list is to allow concatenation of data sets for SYS1.LINKLIB. The link library list must be included in the system.

PREREQUISITE PUBLICATIONS

The IBM System/360 Operating System: System Generation publication (GC28-6554) describes how to specify the options and content of the link pack area at system generation time.

RRO

The IBM System/360 Operating System: Supervisor Services publication (GC28-6646) contains a general description of the BLDL function.

The IBM System/360 Operating System: Utilities publication (GC28-6586) contains a description of the IEBUPDTE utility which you use to construct lists of load module names in the parameter library (SYS1.PARMLIB).

The IBM System/360 Operating System: Storage Estimates publication (GC28-6551) provides storage requirement information for the options and link pack area.

The IBM System/360 Operating System: Messages and Codes publication (GC28-6631) contains the operator message and replies associated with the options and link pack area.

Feature :	BLDL	Link List	RSVC	RAM	RERP
RESIDNT= (a)	BLDLTAB		TRSVC	RENTCODE	ERP
IPL (b)	BLDL=		RSVC=	RAM=	RERP=
Name of List	IEABLD..	LNKLST00	IEARSV..	IEAIGG..	IEAIGE..
Contents of List	Names of Routines	Names of Data Sets	Names of Routines	Names of Routines	Names of Routines
Subject Routines	Link Library		Type 3 and 4 SVC Routines	Access Method and Link Library	Error Rcvy Procedure
Library Residence	SYS1.LINKLIB	Any volume	SYS1.SVCLIB	SYS1.SVCLIB, SYS1.LINKLIB	SYS1. SVCLIB
Operation of Feature	Builds a table of addresses	Concatenates other datasets with the Link Library	Loads Named Routines	Loads Named Routines	Loads Named Routines

(a) - Entry for the SUPRVSOR macro instruction in the system generation procedure.
(b) - Entry for the operator reply to the IPL time message SPECIFY SYSTEM PARAMETERS.

Figure RRO 1. Resident Routines Options - PCP

Option	BLDL	Link List	RSVC	Reenterable Routines		RERP
				Access Methods	Link Library	
RESIDNT=	(a) BLDTAB		TRSVC	ACSMETH	RENTCODE	ERP
IPL	(b) BLDL=		RSVC=	RAM=	RAM=	RERP=
Name of List	IEABLD..	LNKLST00	IEARSV..	IEAIGG.. (c)	IEAIGG.. (c) User-Written	IEAIGE..
Names on the List	Link Library Routines	Data Sets	Type 3 and 4 SVC Routines	Access Method Routines	Reenterable Link Library Routines, Reenterable GSP routines, and the OS Loader	Names of Routines
Residence	SYS1.LINKLIB	Any Volume	SYS1.SVCLIB	SYS1.SVCLIB	SYS1.LINKLIB	SYS1. SVCLIB
Use of Option	To Build a Table of Addresses	To Concatenate other datasets with Link	To Load Named Routines	To Load Named Routines	To Load Named Routines	Loads Named Routines

(a) - Entry for the SUPRVSOR macro instruction in the system generation statements.
(b) - Entry for the operator reply to the IPL time message SPECIFY SYSTEM PARAMETERS.
(c) - Though similarly named, each IEAIGG.. list can have names of only one of the two kinds of routines. Each list must have a unique name.

• Figure RRO 2. Resident Routines Options - MFT

Feature :	BLDL	Link List	RSVC	RAM	RERP
RESIDNT= (a)	BLDTAB		TRSVC	ACSMETH	ERP
IPL (b)	BLDL=		RSVC=	RAM=	RERP=
Name of List	IEABLD..	LNKLST00	IEARSV..	IEAIGG..	IEAIGE..
Contents of List	Names of Routines	Names of Data Sets	Names of Routines	Names of Routines	Names of Routines
Subject Routines	Link Library		Type 3 and 4 SVC Routines	Access Method	Error Rcvy Procedure
Library Residence	SYS1.LINKLIB	Any volume	SYS1.SVCLIB	SYS1.SVCLIB	SYS1. SVCLIB
Operation of Feature	Builds a table of addresses	Concatenates other datasets with the Link Library	Loads Named Routines	Loads Named Routines	Loads Named Routines

(a) - Entry for the SUPRVSOR macro instruction in the system generation procedure.
(b) - Entry for the operator reply to the IPL time message SPECIFY SYSTEM PARAMETERS.

Figure RRO 3. Resident Routines Options - MVT

Section 1: Nucleus Resident Library Routines (PCP and MFT)

The BLDL, reenterable modules, RSVC and RERP options, when included in a PCP or MFT configuration of the operating system, enable you to place in the nucleus area of main storage (make resident):

1. All, or a selection of, Link library directory entries.

2. A selected group of access method routines.

3. A selected group of type 3 and 4 SVC routines.

4. A selected group of error recovery procedures.

5. For MFT, user-written reenterable routines from the Link library, the OS Loader, and reenterable GSP routines.

Placement occurs during the initial program load (IPL) process. The main storage area that these resident routines occupy becomes part of the "fixed storage" area of the system. In effect, the nucleus is expanded.

These options are included in the system when it is generated. The System Generation publication describes the procedure. The resident SVC routine option requires that the Transient SVC Table option also be included in the system. If you wish to exercise control over the other options at IPL time, you must also specify the operator communication facility for these options when the system is generated.

You specify the Link library (SYS1.LINKLIB) routines and directory entries, the access method routines, the type 3 and 4 SVC routines, and the error recovery procedures to be made resident through lists of linkage library, access method, SVC routine, and the error recovery procedures load module names placed in the parameter library (SYS1.PARMLIB).

A standard list and alternative lists of load module names may exist for the options. The standard list (so called because its member name in the parameter library is predefined) is automatically referred to during the IPL process when the operator communication facility is not included in the system with the options. When the operator communication facility is included, the operator must designate which list is to be used. IBM provides suggested standard lists for the resident access method modules and resident SVC routine options. These lists are in the starter system parameter library.

Inclusion of the operator communication facility enables full control over all the options at IPL time, i.e., selection of alternative or standard lists, and suppression of the options until the next IPL. Otherwise, the options are in effect at every IPL, using the standard lists. The operator communication facility is required for the resident Link library modules option of MFT. Unless the operator refers to load list (or lists) for this option in his RAM= reply, none of the modules named on a load list is made resident.

The balance of this chapter discusses the function of each option, the creation of the parameter library lists, and, lists the content of the resident access method modules and resident type 3 and 4 SVC routines standard lists. The Messages and Codes publication describes the message (message number IEA101A) and replies associated with the options.

The Resident BLDL Table Option

System issued ATTACH, LINK, LOAD, or XCTL macro instructions requesting load modules from partitioned data sets cause a search of the data set directory for the location of the requested module (the BLDL table operation) and a fetch of the module. The resident BLDL table option eliminates the directory search required during execution of these macro instructions when a load module (whose directory entry is resident) is requested from the linkage library.

This option builds, in the system nucleus, a list of linkage library directory entries for use by ATTACH, LINK, LOAD, or XCTL macro instructions requesting linkage library load modules. During execution of the BLDL operation in the macro instruction routines, the linkage library directory is searched only when the directory entry for the requested load module is not present in the resident BLDL table.

You list, in a member of SYS1.PARMLIB, the names of those linkage library load modules whose directory entries are to be made resident. The member name for the standard list is IEABLD00. The load module names must be listed in the same order as they appear in the directory; that is, they must be in ascending collating sequence. Creation of parameter library lists is discussed later in this chapter. The next section provides guidelines for choosing the content of the list.

Note: Directory entries in the resident table are not updated as a result of updating the load module in the linkage library. The old version of the load module is used until an IPL operation takes place and the new directory entry for the module is made resident.

SELECTING ENTRIES FOR THE RESIDENT BLDL TABLE

Any load module in the linkage library may have its directory entry placed in the resident BLDL table. Other items you should consider are:

1. Table size. (Each entry requires 40 bytes of storage with PCP and MFT; MVT requires 56 bytes for each entry.)

2. Frequency of use of the load module.

Table Size

The resident BLDL table is incorporated in the system nucleus. The additional storage required is governed by the number of table entries and is acquired by reducing the amount of dynamic storage area available, i.e., the system nucleus expands. Each installation using the resident BLDL table option must determine the amount of storage it can afford for the resident BLDL table.

Frequency of Use

Short of placing the entire linkage library directory in the resident BLDL table, you make the option effective by selecting directory entries representing the load modules which are called most frequently. Your choice will depend on the system configuration and the operating practices of your installation. You should give load modules of the scheduling components of the system, linkage editor, and language processor(s) thorough consideration.

List IEABLD00

The IBM supplied standard list IEABLD00 is:

```
SYS1.LINKLIB    IEBCOMPR,IEBGENER,IEBPTPCH,IEBUPDTE,IEHLIST,IEHMOVE,     X
                IEHPROGM,LINKEDIT,SORT
```

Resident Reenterable Modules Option

The resident reenterable modules options make it possible to pre-load reenterable access method modules (in PCP and MFT) and user-written reenterable Link library modules, the OS Loader (in MFT only) and reenterable GSP routines. These two otherwise independent options -- the resident access method modules option and the resident Link library modules option -- use similarly named load lists (IEAIGG..) and share an operator reply (RAM=) at IPL time to refer to their separate lists.

The resident access methods modules option uses a list or lists (named IEAIGG..) to name the modules to be preloaded and the RESIDNT=ACSMETH entry (in the system generation statements) to cause use of the pre-loaded modules when a DCB is being completed. The sytem comes with a standard list (IEAIGG00) which is used, unless you have changed it or ask for the use of another list in the operator reply.

The resident Link library modules option uses a list or lists (also named IEAIGG.., but ending in a pair of characters other than the ones used to name the resident access methods modules option lists) to name the modules to be pre-loaded. The RESIDNT=RENTCODE entry (in the system generation statements) causes the pre-loaded modules to be used when a LINK, ATTACH, or XCTL macro instruction refers to the name of a resident module. Because there is no standard list, no modules are loaded unless you provide such a list.

To use both access method modules and Link library modules options, the system generation statement entry is: RESIDNT=ACSMETH,RENTCODE and the operator reply is RAM=kk,ll,mm,nn. Each pair of letters is a pair of numbers (like 01) that identify a list of either access method or user-written Link library modules and the OS Loader.

THE RESIDENT ACCESS METHOD MODULES OPTION

This option places access method load modules in the system nucleus and creates a resident list of these modules. (In MVT, these load modules are placed in the link pack area. If the system includes IBM 2361 Core Storage and Main Storage Hierarchy Support, modules may also be placed in the secondary link pack area in Hierarchy 1 using the "HRAM=" reply to "Specify System parameters.") A LOAD macro instruction requesting any access method module first scans the resident list. If the module is listed, no fetch operation is required.

You list, in a member of SYS1.PARMLIB, the load module names of access method load modules to be made resident. The member name for the standard list is IEAIGG00. A standard list of most frequently used access method modules is supplied by IBM, and is in SYS1.PARMLIB of the starter system under the standard member name. The content of the list is tabulated at the end of this description.

RESIDENT LINK LIBRARY MODULES OPTION (MFT)

This option permits, in MFT, preloading user-written reenterable Link library modules, the IBM-supplied OS/360 Loader, and reenterable GSP routines. If you chose to implement this option, the use of a LINK, ATTACH, LOAD, or XCTL macro instruction causes the contents supervision routines to find out whether the module is main storage resident already. If it is, the module already resident is used for that partition in which the macro instruction was used. If it is not, the module is loaded from the Link library into the requesting partition.

IBM supplied modules, except those of the OS/360 Loader, and GSP routines cannot be used with this option. Any user written routine that is reenterable may be used, for example, a user-written reader routine that is reenterable.

How to Include the Resident Link Library Option in Your System

To include the option in your system:

- Code RESIDNT=RENTCODE in your system generation statements to have the contents supervision routines find out whether the load module is main storage resident already.

- Code OPTION=COMM in your system generation statements to allow the operator to have the modules preloaded at IPL time.

- Add to the Parameter library, a list or lists of names of reenterable modules to be preloaded. Each module name must be followed by its alias names (separated by commas).

- Have the operator specify your list or lists in his RAM= reply at IPL time.

You code RESIDNT=RENTCODE and OPTION=COMM to include certain IBM supplied coding in your system.

You name the list of reenterable Link library modules IEAIGGxx. The final two characters (xx) of the name may be any EBCDIC character but should be different from any pair used to name a list of modules for the resident access method modules option. (The lists for the latter option are also named IEAIGG..). You add the list, or lists, to the Parameter library as described later in this chapter.

The modules are finally and actually preloaded if your operator includes the last two characters of the list name in his answer RAM= at IPL time. (Say, for example, you named your list of names of reenterable Link library modules that you want preloaded, IEAIGGAA. The operator's reply must be of the form: RAM=..,..,AA,..) Since there is no standard list for this option, no modules are loaded unless you have constructed a list of names, added it to the Parameter library, and the operator refers to it (as described) in his RAM= response.

The Resident SVC Routines Option

This option places any of the type 3 and 4 SVC routine load modules in main storage. (In MVT, these load modules are placed in the link pack area. If the system includes IBM 2361 Core Storage and Main Storage Hierarchy Support, modules may also be placed in the secondary link pack area in Hierarchy 1 using the "HSVC=" replay to "Specify System Parameters.") Some, or all, of the modules associated with a SVC service routine may be made resident. Placing the most frequently used SVC load modules of a system service routine, such as OPEN, in main storage improves system performance. For type 3 SVC load modules and initial type 4 SVC load modules, the SVC table entries associated with these modules are adjusted to reflect an entry point address rather than a relative track address. A resident SVC load list is used by the XCTL macro instruction for transfer of control between resident type 4 SVC load modules.

You list, in a member of SYS1.PARMLIB, the type 3 and 4 SVC load modules to be made resident. The member name for the standard list is IEARSV00. Such a standard list (shown below) is provided by IBM in SYS1.PARMLIB of the starter system. The creation of parameter library lists is discussed later in this chapter.

If your system includes the Multiple Console Support (MCS) function, to improve MCS performance you should add to the standard list (or include in a list of your own) IGC0007B, the name of the first load module of the SVC 72 routine.

Storage Requirements

The Storage Estimates publication provides the byte requirements of type 3 and 4 SVC routines eligible to be made resident. The byte requirement of the code supporting the option is also provided.

List IEARSV00

The content of the IBM supplied standard list IEARSV00 is:

Module Name	Function
IGC0001F	Purge Routine
IGC0001I	Open - Initial Load - Part 1
IGC0199X	Open - Initial Load - Part 2
IGC0005E	EOV - Initial Load
IGC0002_ (b,?)*	Close - Initial Load
IGG0190L	Open - Merge and Access Method Determination
IGG0190M	Open - Merge and DCB Exit Routine
IGG0190N	Open - Final Load
IGG0190S	Open - Rewrite JFCB
IGG0191A	Open - DEB Construction (First Load)
IGG0196A	Open - DEB Construction (Second Load)
IGG0191B	Open - Main Executor (First Load)
IGG0196B	Open - Main Executor (Second Load)
IGG0191D	Open - Direct Access Executor
IGG0191G	Open - Tape and Unit Record Executor
IGG0191O	Open - Tape/Unit Record Executor
IGG01910	Open - Load Executor
IGG01917	Open - Second Load of Load Executor
IGG01911	Open - IOB and Buffer Construction
IGG0199M	Open - JFCB Merge
IGG0200A	Close - Read JFCB and DSCB
IGG0200F	Close - Direct Access Routine
IGG0200G	Close - Delete Routine
IGG0200H	Close - Second Load of Delete Routine
IGG0200Y	Close - Direct Access Processing
IGG0200Z	Close - Second Load
IGG0201Y	Close - Release Work Areas and Buffers
IGG0201Z	Close - SAM Executor (SAM - Common sequential access methods modules)
IGG0209Z	Close - XCTL

*The last (eighth) character is a 12 and 0 punch. In EBCDIC this is b (the blank character), in BCD ? (the question mark).

THE RESIDENT ERROR RECOVERY PROCEDURE OPTION

This option places error recovery procedures in main storage. Some, or all, of the modules associated with the handling of an I/O error may be made resident. If an I/O device frequently requires ERP processing, system performance improves if the error recovery procedures are made resident. The list of those error recovery procedures that may be made resident in main storage is contained in the Storage Estimates publication. An I/O supervisor request for an error recovery procedure will result in a search of the resident error recovery procedure list. If the error recovery procedure is resident, no fetch operation is required.

You list, in a member of SYS1.PARMLIB, the module names of error recovery procedures to be made resident. The member name for the standard list is IEAIGE00. After system generation, you will have the option of indicating which error recovery procedures are to be made resident. The error recovery procedures should be listed by expected frequency of use; the least used module is first in the list. Note: The format of the IBM-supplied IEAIGE00 list contains the required library name, SYS1.SVCLIB, and no error recovery procedure names. After system generation, IEAIGE00 can be updated to indicate which error recovery procedures are to be made resident or an alternate list can be created. Until this update is performed, no error recovery procedures will be made resident during the IPL process. The creation of parameter library lists is discussed later in this chapter.

Storage Requirements

The *Storage Estimates* publication provides the byte requirements of error recovery procedures that may be made resident. The byte requirement of the code supporting the option is also provided.

Creating Parameter Library Lists

You use the IEBUPDTE utility program to construct the required lists of load module names in the parameter library. Standard member names for these lists are:

IEABLD00 for the BLDL table option

IEAIGG00 for the access method option

IEARSV00 for the SVC routine option

IEAIGE00 for the error recovery procedure option

LNKLST00 for the link library list option

These are the member names that the nucleus initialization program recognizes at IPL time in the absence of any other specification, i.e., when the operator communication facility is not incorporated.

Note: The nucleus initialization program (NIP) will search the system catalog to locate the SYS1.PARMLIB data set. If it is not found in the catalog, SYS1.PARMLIB is assumed to reside on the IPL volume. If no VTOC entry can be found, the operator will receive message IEA211I "OBTAIN FAILED FOR SYS1.PARMLIB DATA SET". Message IEA208I "fff FUNCTION INOPERATIVE" will follow either of these messages. The fff parameter - RAM, BLDL, RSVC, or RERP - shows which of the functions cannot be implemented. Processing will continue; however, any resident functions dependent on parameter lists contained in the parameter library will be omitted from the system nucleus.

Except for LNKLST00, your input format (to IEBUPDTE) for the lists is the same for all three options, consisting of library identification followed by the load module names. You use eighty character records with the initial or only record containing the library identification. Continuation is indicated by placing a comma after the last name in a record and a nonblank character in column 72. Subsequent records must start in column 16.

The initial record format (with continuation) is:

```
 1                                                72
        SYS1.LINKLIB
[b...]  SYS1.SVCLIB    b...name1,name2,name3,...X
```

Subsequent records do not contain the library name.
SYS1.LINKLIB indicates that linkage library load module names follow.
SYS1.SVCLIB indicates that SVC library module names follow.

You may construct alternative lists for all but the LNKLST00 option and place them in the parameter library. Member names for these alternative lists are of the form:

IEABLDxx for the BLDL option

IEAIGGxx for the resident access method option

IEARSVxx for the resident SVC routine option

IEAIGExx for the resident error recovery procedure option

LNKLST00 for the link library list option

where xx can be any two alphameric characters.

The Messages and Codes publication describes the operator message and responses associated with use of the link pack area.

Programming Notes

A list of the load modules always placed in the link pack area by the system is contained in the Storage Estimates publication. The main storage space requirements of these modules determines the basic (minimum) size of the link pack area. The area is extended by the number of storage bytes needed to accommodate the load modules and BLDL table content specified at IPL time.

Placing the initiator/terminator load module IEFSD061 in the link pack area enables the system to make more efficient use of the dynamic area of storage. The operating system allocates to each job a part of a region not less than the size required to accommodate the initiator-terminator. This allocation is from processor storage (hierarchy 0) and occurs even when the REGION parameter requests less than the required space or no space. After initiation, the part of the region in hierarchy 0 is reduced by as much as 40,000 bytes when the job terminator is resident in the link pack area.

EXAMPLE OF LINK PACK AREA SPECIFICATION

The following example illustrates the extension of the link pack area to contain SVC load modules, other reenterable load modules, and a BLDL table. The RESIDNT field of your system generation SUPRVSOR macro instruction would look like:

```
          .
          .
          .
SUPRVSOR  RESIDNT=TRSVC,RENTCODE,BLDLTAB...
          .
          .
          .
```

If you intend to alter the content of your link pack area, you would also specify: OPTIONS=COMM,... in the SUPRVSOR macro instruction.

Assume that you wish to place five lists on SYS1.PARMLIB. These lists are:

1. IEARSV00, which contains names of modules of the Open SVC routine used for direct access devices.

2. IEARSV20, which contains names of modules of the Close SVC routine.

3. IEAIGG01, which contains names of modules of the basic sequential access method (BSAM).

4. IEABLD00, which contains names of modules of the initiator portion of the job scheduler.

5. IEABLDF0, which contains names of modules of both the FORTRAN compiler and the initiator.

Note that there is no standard list for reenterable modules from the linkage or SVC library (IEAIGG00). This implies that you don't want modules of this type loaded unless a list is explicitly specified.

To place these lists in SYS1.PARMLIB, you could use the IEBUPDTE utility program as shown:

```
//ADDLISTS   JOB      61938,R.L.WILSON
//STEP       EXEC     PGM=IEBUPDTE,PARM=NEW
//SYSPRINT   DD       SYSOUT=A
//SYSUT2     DD       DSNAME=SYS1.PARMLIB,DISP=OLD
//SYSIN      DD       DATA
./           ADD      NAME=IEARSV00,LIST=ALL
./           NUMBER   NEW1=01,INCR=02
SYS1.SVCLIB  IGG0190I,IGG0190L,IGG0190M,                            C
                      IGG0190S,IGG0190Z
./           ADD      NAME=IEARSV20,LIST=ALL
./           NUMBER   NEW1=01,INCR=02
SYS1.SVCLIB  IGC00020,IGG0200A,IGG0200B,IGG0200C,IGG0200F,          C
                      IGG0200G,IGG0200Y
./           ADD      NAME=IEAIGG01,LIST=ALL
./           NUMBER   NEW1=01,INCR=02
SYS1.SVCLIB  IGG019BA,IGG019BB,IGG019BC,IGG019BD,                   C
                      IGG019BE,IGG019BF,IGG019BG,                   C
                      IGG019BH,IGG019BI,IGG019BK,IGG019BL
./           ADD      NAME=IEABLD00,LIST=ALL
./           NUMBER   NEW1=01,INCR=02
SYS1.LINKLIB IEFSD061,IEFSD062,IEFSD065,IEFSD104,                   C
                      IEFUM1,IEFWC000,IEFWD000,                     C
                      IEFW21SD,IEFW41SD,IEFW42SD,IEFXJ000
./           ADD      NAME=IEABLDFO,LIST=ALL
./           NUMBER   NEW1=01,INCR=02
SYS1.LINKLIB IEFSD061,IEFSD062,IEFSD065,IEFSD104,                   C
                      IEFUM1,IEFWC000,IEFWD000,IEFW21SD,            C
                      IEFW41SD,IEFW42SD,IEFX4J000,IEJAAA0,          C
                      IEJEAA0,IEJFAA0,IEJGAA0,IEJJAA0,              C
                      IEJLAA0,IEJNAA0,IEJPAA0,IEJRAA0,              C
                      IEJVAA0,IEJXAA0
                      IEFWD000,IEFW41SD
./           ENDUP
/*
```

Without operator communication only the standard lists IEARSV00 and IEABLD00 would be referred to at IPL time. With operator communication use of all the lists or any combination could be specified at IPL time.

If after a given IPL you intend to extensively use the FORTRAN compiler, and BSAM with direct access devices, you would probably want to use all of these lists -- except IEABLD00 -- to specify the content of your extended link pack area. To do this your operator would specify the following in response to the SPECIFY SYSTEM PARAMETERS operator's message:

```
REPLY id, "RSVC=00,20,RAM=01,BLDL=F0"
```

If, after an IPL you intended to perform general processing without extensive use of any particular compiler or access method, you might want to put just the linkage library directory entries of initiator modules in a BLDL table. In this case, your operator's reply at IPL would be:

```
REPLY id, "RSVC=,RAM=,"
```

Since the list of initiator modules is the standard list, it need not be specified. "RSVC=," must be specified to prevent the use of the standard list of SVC modules. Although you have no standard list of reenterable modules "RAM=," should be specified to prevent NIP from performing unnecessary processing.

Job Queue Format

The job queue format is specified when the system is generated and may be altered during subsequent system start procedures. In MFT and MVT, formatting consists of specifying the number of queue records in a job queue logical track, reserving queue records for initiators, the write-to-programmer routine, and reader/interpreters, and reserving queue records for job cancellation.

This chapter provides guidelines for estimating:

For PCP:

- The number of records to be made resident.

For MFT and MVT:

- The number of queue records in a job queue logical track.
- The number of queue records to be reserved for use by an initiator and reader/interpreter.
- The number of queue records to be reserved for cancellation of job initiation and running when the number of queue records reserved for initiator use is insufficient.
- The number of records to be reserved for the write-to-programmer routine.

REFERENCE PUBLICATIONS

The IBM System/360 Operating System: System Generation publication (GC28-6554) describes the SCHEDULR macro instruction parameters used to initially specify job queue format.

The IBM System/360 Operating System: Operator's Reference publication (GC28-6692) describes the procedure used to alter job queue format.

The IBM System/360 Operating System: Service Aids publication describes the service aids program IMCSQDMP.

The Resident Job Queue Option (PCP only)

This option places a specified number of system job queue records in main storage rather than in external storage (the SYS1.SYSJOBQE data set). The records are taken sequentially from the beginning of the queue. There is one break in the sequence which is noted in the next section "Operational Characteristics."

Operational Characteristics

The job queue is formatted as a series of 176 byte records. The first 42 records form a "fixed group" of job queue records used by the scheduler. These 42 records are always present in the job queue. Of this group, 26 records are used by the interpreter routines of the scheduler as a work area. These 26 records are never made resident by the option. The remaining 16 records in the "fixed group" may be made resident. After the "fixed group" of records, a series of records forming a "variable group" of job queue records is developed. The number of records in the "variable group" fluctuates from job to job reflecting the make-up of the input job stream for the job being read in. All records in the "variable group" may be made resident.

Starting with the first (in sequence) of the 16 eligible "fixed group" records, the option places the specified number of records in main storage. For example, a specification of 5 resident records will place the first 5 of the 16 "fixed group" records in main storage; a specification of 20 resident records will place all 16 of the "fixed group" records in main storage plus the first 4 records from the "variable group."

Reference to a specific job queue record causes a test to be made -- in resident queue or in external storage -- and the record is referred to accordingly.

In an MFT configuration of the operating system only the "variable group" job queue records developed from the input job stream for the lowest priority partition may be made resident.

Determining Resident Job Queue Size

The storage occupied by the resident job queue cannot be allocated to any other use, therefore you must determine the amount of storage your installation can afford to devote to a resident job queue. Since the size of the queue can be varied from IPL to IPL you may want to estimate several sizes -- each estimate reflecting a feasible job queue size in view of the work to be performed after the IPL.

The following formula can be used to estimate the number of resident job queue records developed for a given job. The constant (16) represents the 16 "fixed group" records that are always developed and are eligible for inclusion in the resident job queue.

$$\text{Number of Records} = 16 + \frac{B}{3} + 2C + \frac{E}{28} + \frac{F}{176} + 3G + \frac{H-5}{15} + \frac{J}{22}$$

Where:

B = the number of data sets passed between job steps.

C = the number of steps in the job.

E = the number of volume serial numbers specified in the DD statements for each job step. (Evaluate each job step separately and sum the results to obtain the total value.)

F = the number of characters in data set names, including qualifiers, appearing in DD statements in the parameter VOL=REF=dsname. (Evaluate each job step separately and sum the results to obtain the total value.)

G = the number of DD statements in the job.

H = the number of volume serial numbers specified in each DD statement (if $H \leq 5$, $H-5=0$). (Evaluate each DD statement separately and sum the results to obtain the total value.)

J = the total number of job control language statements used to describe the job, when all messages are to be written on the system output device, otherwise J=1.

Multiplying the number of records by 176 provides the resident job queue size in terms of bytes.

If possible, the entire set of eligible job queue records should be made resident. It is recommended that at least the 16 eligible records from the "fixed group" of job queue records be made resident.

MVT Job Queue Formatting

In MVT and MFT operating system configurations, the basic element of the system job queue (the data set SYS1.SYSJOBQE) is a 176-byte record -- the queue record. The total number of queue records available is fixed by the space allocated to the SYS1.SYSJOBQE data set. Queue records contain the tables, control blocks, and system messages developed by the reader/interpreter, write-to-programmer, and initiator control program routines -- the information used to run a job.

Lack of queue records to work with is not critical for a reader/interpreter routine. In MVT processing of the input job stream assigned to a reader/interpreter is suspended until queue records become available, at which time processing is resumed. In MFT the operator will receive a message if there is insufficient space for a reader/interpreter. He may wait for space or cancel the reader. An initiator, however, must have sufficient queue records available to complete the initiation and running of a job or the job is canceled. Because, in an MVT configuration, one or more reader/interpreters and one or more initiators may be concurrently active, steps must be taken to ensure that queue records are available to each initiator started, so that it may complete its operations. In addition queue records must be reserved for use by initiators in the event job cancellation does take place. The main function of job queue formatting is to reserve queue records for initiator use.

To format the job queue you must:

1. Designate the number of queue records to be contained in a job queue logical track. A logical track consists of a header record (20 bytes) plus the designated number of queue records. Reader/interpreters and initiators are assigned queue records in terms of logical tracks.

2. Designate the number of queue records to be reserved for use by an initiator. Each initiator is allocated this number of records. If the allocation is insufficient for the job currently being processed by the initiator, the job is canceled in MVT.

3. Designate the number of queue records to be reserved for use in case of job cancellation. All initiators that cancel use these queue records. If the allocation is insufficient, the initiator is placed in a WAIT state and a message issued.

4. Designate the number of queue records to be reserved for write-to-programmer routine use for each job that may be started by an initiator.

The balance of the queue (total queue records less the reservations in items 2, 3, and 4) is available for use by the reader/interpreters.

You specify initial values for logical track size, queue record reservation for initiators, and queue record reservation for job cancellation, in the SCHEDULR macro instruction parameters JOBQFMT, JOBQLMT, JOBQTMT, AND JOBQWTP respectively. The System Generation publication describes the procedure.

The service aids program IMCJQDMP provides a formatted dump of the entire job queue, or selected portions of it. The formatted dump includes the master queue control record (QCR) which contains the physical parameters of the job queue. For a complete description of IMCJQDMP, see the publication IBM System/360 Operating System: Service Aids.

There are no comprehensive, foolproof formulas for calculating values of JOBQFMT, JOBQLMT, JOBQTMT, and JOBQWTP. The values to be estimated are dependent upon the requirements and structure of the jobs to be presented to the system, the number of job steps, the number of I/O devices required, the number and type of data sets, the number of

Output for Each Device Type

	UCB Type Field (Word 1, In Hexadecimal)	Maximum Record Size (Word 2, In Decimal)	DEVTAB (Words 3, 4, and 5, In Hexadecimal)
2540 Reader	10 00 08 01	80	Not Applicable
2540 Reader w/CI	10 01 08 01	80	Not Applicable
2540 Punch	10 00 08 02	80	Not Applicable
2540 Punch w/CI	10 01 08 02	80	Not Applicable
1442 Reader-Punch	50 00 08 03	80	Not Applicable
1442 Reader-Punch w/CI	50 01 08 03	80	Not Applicable
1442 Serial Punch	51 80 08 03	80	Not Applicable
1442 Serial Punch w/CI	51 01 08 03	80	Not Applicable
2501 Reader	50 00 08 04	80	Not Applicable
2501 Reader w/CI	50 01 08 04	80	Not Applicable
2520 Reader Punch	50 00 08 05	80	Not Applicable
2520 Reader Punch w/CI	50 01 08 05	80	Not Applicable
2520 B2-B3	11 00 08 05	80	Not Applicable
2520 B2-B3 w/CI	11 01 08 05	80	Not Applicable
1403	10 00 08 08	120*	Not Applicable
1403 w/UCS	10 80 08 08	120*	Not Applicable
1404	10 00 08 08	120*	Not Applicable
1443	10 00 08 0A	120*	Not Applicable
2671	10 00 08 10	32767	Not Applicable
1052	10 00 08 20	130	Not Applicable
2400 (9-track)	30 00 80 01	32767	Not Applicable
2400 (9-track, p.e.)	34 00 80 01	32767	Not Applicable
2400 (9-track, d.d.)	34 20 80 01	32767	Not Applicable
2400 (7-track)	30 80 80 01	32767	Not Applicable
2400 (7-track, d.c.)	30 C0 80 01	32767	Not Applicable
2301	30 40 20 02	20483	000100C85003BA3535000200
2302	30 00 20 04	4984	00FA002E1378511414010219
2303	30 00 20 03	4892	0050000A131C922626000200
2311	30 00 20 01	3625	00CB000A0E29511414010219
2314	30 C0 20 08	7294	00CB00141C7E922D2D010216
2321	30 00 20 05	2000	140A051407D0641010030219
3210 Printer Keyboard	10 00 08 22	130	Not Applicable
3215 Printer Keyword	10 00 08 23	130	Not Applicable

		Maximum	
	UCB Type Field	Record Size	DEVTAB
	(Word 1,	(Word 2,	(Words 3, 4, and 5,
Graphics Devices	In Hexadecimal)	In Decimal)	In Hexadecimal)
1053	14 00 10 04		Not Applicable
2250 (Mod 1)	31 xx 10 02		Not Applicable
2250 (Mod 2)	32 xx 10 02		Not Applicable
2250 (Mod 3)	33 xx 10 02		Not Applicable
2280	30 00 10 05		Not Applicable
2282	30 00 10 06		Not Applicable
2260 (Mod 1)	11 xx 10 03		Not Applicable
2260 (Mod 2)	12 xx 10 03		Not Applicable
3066 (Model 165 System Console)	10 00 10 08		Not Applicable
5450 (Model 85 Operators Console)	10 00 10 07		Not Applicable

Legend

CI-Card Image Feature, d.c.-data conversion, d.d.-dual density, p.e.-phase encoding, UCS-Universal Character Set, w/-with

*Although certain models can have a larger line size, the minimum line size is assumed.

xx = Special Feature (byte 2) configurations may be obtained from the System Control Blocks publication.

Communication Equipment	UCB Type Field	Record Size
1030,1050,83B3, TWX,2250, S360	51xx40YZ	Not Applicable
1060,115A,1130	52xx40YZ	Not Applicable
2780	53xx40YZ	Not Applicable
2740	54xx40YZ	Not Applicable

Y=Adapter Type (Bits 0-3)

Hex Value	Meaning
1	IBM Terminal Adapter, Type I
2	IBM Terminal Adapter, Type II
3	IBM Telegraph Adapter
4	Telegraph Adapter, Type I
5	Telegraph Adapter, Type II
6	World Trade Telegraph Adapter
7	Synchronous Adapter, Type I
8	IBM Terminal Adapter, Type III
9	Synchronous Adapter, Type II

Z=Control Unit (Bits 4-7)

Hex value	Meaning
1	2702
2	2701
3	2703

Exceptional Returns

The following return codes are placed in register 15:

00 - Request completed satisfactorily.
04 - Ddname not found.
08 - Invalid area address. The address of the output area either violates protection, or it is out of the range of main storage.

Examples: The macro instruction in EX1 creates a parameter list for two data control blocks: INVEN and MASTER. In creating the list, both data control blocks are assumed to be opened for input; opt_2 for both blocks is assumed to be DISP. The macro instruction in EX2 reads the system-created JFCBs for INVEN and MASTER from the job queue into main storage, thus making the JFCBs available to the problem program for modification. The macro instruction in EX3 modifies the parameter list entry for the data control block named INVEN and indicates, through the TYPE=J operand, that the problem program is supplying the JFCBs for system use.

```
EX1      RDJFCB (INVEN,,MASTER),MF=L
         .
         .
EX2      RDJFCB MF=(E,EX1)
         .
         .
EX3      OPEN   (,(RDBACK,LEAVE)),TYPE=J,MF=(E,EX1)
         .
         .
```

Programming Notes

Any number of data control block addresses and associated options may be specified in the RDJFCB macro instruction. This facility makes it possible to read job file control blocks in parallel.

An exit list address must be provided in each data control block specified by an RDJFCB macro instruction. Each exit list must contain an active entry that specifies the main storage address of the area into which a JFCB is to be placed. A full discussion of the exit list and its use is contained in the Supervisor and Data Management Services publication. The format of the job file control block exit list entry is as follows:

Type of Exit List Entry	Hexadecimal Code (high-order byte)	Contents of Exit List Entry (three low-order bytes)
Job file control block	07	Address of a 176-byte area to be provided if the RDJFCB or OPEN (TYPE=J) macro instruction is used. This area must begin on a fullword boundary and must be located within the user's region.

The main storage area into which the JFCB is read must be at least 176 bytes long.

The data control block may be open or closed when this macro instruction is executed.

Cautions: The following errors cause the results indicated:

Error	Result
A DD control statement has not been provided.	No action
A main storage address has not been provided.	Abnormal termination of task

L- and E-Form Use: The L and E forms of this macro instruction are written as described in the Supervisor Services and Supervisor and Data Management Macro Instructions publications.

CIRB - Create IRB for Asynchronous Exit Processing

The CIRB macro instruction is included in SYS1.MACLIB and must be included in your system at system generation time if you intend to use it. The issuing of this macro instruction causes a supervisor routine (called the exit effector routine) to create an interruption request block (IRB) if one is not already in existence for the task in question. In addition, other operands of this macro instruction may specify the building of a register save area and/or a work area to contain interruption queue elements, which are used by supervisor routines in the scheduling of the execution of user exit routines.

Name	Operation	Operand
[symbol]	CIRB	{EP=addrx}, KEY=$\left\{\begin{array}{l}\underline{\text{PP}}\\\text{SUPR}\end{array}\right\}$, MODE=$\left\{\begin{array}{l}\underline{\text{PP}}\\\text{SUPR}\end{array}\right\}$, [STAB=code,] SVAREA=$\left\{\begin{array}{l}\underline{\text{NO}}\\\text{YES}\end{array}\right\}$, [WKAREA=value]

EP

specifies the entry point address of the user's asynchronous exit routine.

KEY

specifies whether the user's asynchronous routine will operate with a CPU protection key established by the supervisory program (SUPR) or with a protection key obtained from the task control block of the task for which the macro instruction is issued (PP).

MODE

specifies whether the user asynchronous routine will be executed in the problem program (PP) state or in a supervisory (SUPR) state.

STAB

indicates the status condition of the interruption request block. The 'code' parameter may be either of the following:

(RE) to indicate that the IRB is reusable in its current form.

(DYN) to indicate that the storage area assigned to the IRB is to be made available (i.e., freed) for other uses when the asynchronous exit routine is completed.

SVAREA

specifies whether a register save area (of 72 bytes) is to be obtained from the main storage assigned to the problem program. If it is, the address of this save area is placed in the IRB. The asynchronous exit routine then follows the system register saving convention of using the SAVE and RETURN macro instructions. In this manner, a generalized subroutine can be used as an asynchronous exit routine.

WKAREA

specifies the number of doublewords (given as a decimal value) required for an area in which the routine issuing the macro instruction can construct interruption queue elements.

SYNCH - Synchronous Exits to Processing Program

The SYNCH macro instruction is a system macro instruction that permits control program supervisor call (SVC) routines to make synchronous exits to a processing program.

Name	Operation	Operand
[symbol]	SYNCH	⎧entry-point⎫ ⎩ (15) ⎭

entry-point
: specifies the address of the entry point for the processing program that is to be given control.

  If (15) is specified, the entry-point address of the processing program must have been pre-loaded into parameter register 15 before execution of this macro instruction.

SYNCH Macro Definition

```
          MACRO
&NAME     SYNCH     &EP
          AIF       ('&EP' EQ '').E1
          AIF       ('&EP'(1,1) EQ '(').REG
&NAME     LA        15,&EP                       LOAD ENTRY POINT ADDRESS.
          AGO       .SVC
.REG      AIF       ('&EP' EQ '(15)').NAMEIT
&NAME     LR        15,&EP(1)                    LOAD ENTRY POINT ADDRESS.
.SVC      SVC       12                           ISSUE SYNCH SVC
          MEXIT
.NAMEIT   ANOP
&NAME     SVC       12                           ISSUE SYNCH SVC
          MEXIT
.E1       IHBERMAC  27,405
          MEND
```

Programming Notes: In general, you use the SYNCH macro instruction when a control program in the supervisor state is to give temporary control to a processing program routine, and you expect the processing program to return control to the supervisor state. The program to which control is given must be in main storage when the macro instruction is issued. The use of this macro instruction is similar to that of the BALR instruction in that register 15 is used for the entry point address. When the processing program returns control, the supervisor state bit, the storage protection key bits, the system mask bits and the program mask bits of the program status word are restored to the settings they had before execution of the SYNCH macro instruction.

Example: As a result of an OPEN macro instruction, label processing may be carried out to a point at which a user's processing program indicates that private processing is desired (or necessary). The control program's open routine then will issue a SYNCH macro instruction giving the entry point of the subroutine required for the user's private label processing.

STAE - Specify Task Asynchronous Exit

The STAE macro instruction permits control to be returned to a user exit routine when a task is scheduled for ABEND. When you issue the STAE macro instruction, a STAE control block (SCB) is created and initialized with the address of your user exit routine. If you issue multiple STAE requests within the same program, the SCB associated with the last issued STAE request becomes the active SCB: it will be the first to gain control when an ABEND is scheduled. If the active SCB is cancelled, the preceding SCB, if there is one, will become the active SCB.

Notes:

- You cannot cancel or overlay an SCB not created by your program.
- The execution of a LINK macro instruction does not cancel the active SCB for the program in control.

Name	Operation	Operand
[symbol]	STAE	$\left\{\begin{matrix}0\\ \text{exit address}\end{matrix}\right\} ,\left\{\begin{matrix}\text{OV}\\ \underline{\text{CT}}\end{matrix}\right\} \left[\text{,PARAM=list address}\right] \left[\text{,XCTL=}\left\{\begin{matrix}\text{YES}\\ \underline{\text{NO}}\end{matrix}\right\}\right]$
		$\left[\text{,PURGE=}\left\{\begin{matrix}\underline{\text{QUIESCE}}\\ \text{HALT}\\ \text{NONE}\end{matrix}\right\}\right] \left[\text{,ASYNCH=}\left\{\begin{matrix}\underline{\text{NO}}\\ \text{YES}\end{matrix}\right\}\right]$

exit address
specifies the address of a STAE exit routine to be entered if the task issuing this macro instruction terminates abnormally. If 0 is specified, the last SCB created is canceled and the previously created SCB becomes current. The address may be loaded into one of the general registers (r_1) 2 through 12.

Note: If you use the Execute form of the macro and specify a zero, the exit address in the parameter list will be zeroed.

OV
indicates that the parameters passed in this STAE macro instruction are to overlay the data currently in the SCB.

CT
indicates the creation of a new active SCB.

PARAM=
specifies the address of a parameter list containing data to be used by the STAE exit routine when it is scheduled for execution. The address may be loaded into one of the general registers (r_2) 2 through 12.

XCTL=YES
indicates that the STAE macro instruction will not be canceled if an XCTL macro instruction is issued.

XCTL=NO
indicates that the STAE macro instruction will be canceled if an XCTL is issued.

PURGE=QUIESCE

indicates that all active input/output operations will be purged with the quiesce option. If this fails, active input/output operations will be purged with the halt option.

Note: If you use the execute form of the STAE macro instruction and omit the PURGE parameter, QUIESCE will not be the default; the option specified for the preceding use of STAE will be used.

PURGE=HALT

indicates that all active input/output operations will be purged with the halt option.

PURGE=NONE

indicates that all active input/output operations will not be purged

ASYNCH=NO

indicates that asynchronous exit processing will be prohibited while STAE exit processing is being done.

ASYNCH=YES

indicates that asynchronous exit processing will be allowed while STAE exit processing is being done.

Note: If you use the Execute form of the STAE macro instruction and omit the ASYNCH parameter, the option specified for the preceding use of STAE will be used.

There are several conditions that you should be aware of when you use the PURGE and ASYNCH parameters of the STAE macro instruction.

- If your user exit routine requests a supervisor service that requires asynchronous interruptions to complete its normal processing, you must specify ASYNCH=YES.

- You must specify ASYNCH=YES if you use an access method that requires asynchronous interruptions to complete its normal processing and you have specified PURGE=QUIESCE.

- If you are using the Indexed Sequential Access Method (ISAM) and specify PURGE=HALT, only the I/O event for which the PURGE is done will be posted. Subsequent ECBs will not be posted; this causes the ISAM CHECK routine to treat purged input/output operations as waiting input/output operations and you will never get past the CHECK in your program.

- You must specify ASYNCH=YES when you have the following combination of conditions: an access method that requires asynchronous interruptions to complete its normal processing, a specifications of PURGE=NONE, and a request of CHECK in your user exit routine.

- If you specify PURGE=HALT and an ISAM data set is being updated when a failure occurs, part of the data set may be destroyed.

- If quiesced input/output operations are not restored and you are using ISAM, the ISAM CHECK routine will treat purged input/output operations as waiting input/output operations and part of the ISAM data set may be destroyed if it is being updated when a failure occurs.

- If input/output operations are allowed to complete while your exit routine is in progress and there is a failure in the I/O processing, you will encounter an ABEND recursion when the I/O interrupt occurs. This can be misleading because it will appear that your exit routine failed while the actual cause of the failure was in the I/O processing.

Programming Notes

When control is returned to the user after the STAE macro instruction has been issued, register 15 contains one of the following return codes:

Code	Meaning
00	An SCB is successfully created, overlaid, or cancelled.
04	Storage for an SCB is not available.
08	The user is attempting to cancel or overlay a non-existent SCB, or is issuing a STAE in his STAE exit routine.
0C	The exit routine or parameter list address is invalid.
10	The user is attempting to cancel or overlay an SCB not associated with his level of control.

When a program with an active STAE environment encounters an ABEND situation, control is returned to the user through the ABEND/STAE interface routine at the STAE exit routine address. The register contents are as follows:

- Register 0:

Code	Indication
0	Active I/O at time of ABEND was quiesced and is restorable.
4	Active I/O at time of ABEND was halted and is not restorable.
8	No I/O was active at the time of the ABEND.

- Register 1: Address of a 104-byte work area:

0	STAE exit routine parameter list addr or 0	ABEND completion code
8	PSW at time of ABEND	
16	Last P/P PSW before ABEND	
24	Registers 0-15 at time of ABEND (64 bytes)	

If problem program issued STAE:

88	Name of ABENDing program or 0	
96	Entry point addr of ABENDing program	0

If supervisor program issued STAE:

88	Request Block addr of ABENDing program	0
96	0	

- Registers 2-12: Unpredictable.
- Register 13: Address of a supervisor save area.
- Register 14: Address of an SVC 3 instruction.
- Register 15: Address of the STAE exit routine.

Registers 13 and 14, if used by the STAE exit routine, must be saved and restored prior to returning to the calling program. Standard subroutine linkage conventions are employed.

If storage was not available for the work area, the register contents upon entry to the STAE exit routine are as follows:

- Register 0: 12.
- Register 1: ABEND completion code, as in the TCBCMP field.
- Register 2: Address of STAE exit parameter list.

The STAE exit routine may contain an ABEND, but must not contain either a STAE or an ATTACH macro instruction. At the time the ABEND is scheduled, the STAE exit routine must be resident as part of the program issuing STAE, or brought into storage via the LOAD macro instruction.

The STAE exit routine may perform pre-termination functions, attempt to diagnose the error, or attempt to correct the error by specifying that a retry routine is to be scheduled. If a retry routine is not specified, normal ABEND processing continues by returning control to the ABEND/STAE interface routine with a 0 in register 15. To schedule a retry routine with a purge of the RB chain (not including the STAE user's RB), the STAE exit routine must return to the ABEND/STAE interface routine with a 4 in register 15. Register 0 must contain the address of the retry routine, and register 1, the address of the work area. (The work area is the same as that passed to the exit routine except that the first word may now contain user data.)

In supervisor mode, you may want the failing task to remain in its present status and not be reestablished. A retry routine may be scheduled without a purge of the RB chain by returning to the ABEND/STAE interface routine with an 8 in register 15, and registers 0 and 1 initialized as described above. If the STAE retry routine is scheduled, the system automatically cancels the active SCB and the preceding SCB, if there is one, will become the active SCB. If you want to maintain protection against ABEND, you must re-establish an active SCB within the retry routine, or you must issue multiple STAE requests prior to the time that the retry routine gains control.

Like the STAE exit routine, the STAE retry routine must be in storage when the exit routine determines that retry is to be attempted. If not already resident within your program, the retry routine may be brought into storage via the LOAD macro instruction by either the user's program or exit routine.

Upon entry to the STAE retry routine, register contents are as follows:

- Register 0: 0
- Register 1: Address of the work area, as previously described, except that word 2 now contains the address of the first I/O Block and word 26 now contains the address of the I/O restore chain.
- Registers 2-13: Unpredictable.
- Register 14: Address of an SVC 3 instruction.
- Register 15: Address of the STAE retry routine.

The retry routine should free the 104 bytes of storage occupied by the work area when it is no longer needed.

Again, if the ABEND/STAE interface routine was not able to obtain storage for the work area, register 0 contains a 12; register 1, the ABEND completion code upon entry to the STAE retry routine; and register 2, the address of the first I/O Block on the restore chain, or 0 if I/O is not restorable.

If the STAE retry routine is scheduled with a purge of the RB chain, all RBs are purged from the RB of the program that is being terminated up, but not including, the RB of the program that issued the STAE request. The purge is accomplished by placing an SVC 3 in the old PSW field of the RB. In addition, all open DCBs associated with the purged RBs are closed and all queued I/O requests associated with the DCBs being closed are deleted from the I/O restore chain.

When your STAE exit routine gains control, it can examine the code in register 0 to determine if there were active input/output operations at the time of the ABEND and if the input/output operations are restorable. If there are quiesced restorable input/output operations, you can restore them, in the STAE retry routine, by using word 26 in the work area. Word 26 contains the link field passed as a parameter to SVC Restore. SVC Restore is used to have the system restore all I/O requests on the I/O restore chain. For further information, see the section in this publication on the Restore macro instruction.

You can selectively restore specific I/O requests on the I/O restore chain by using word 2 in the work area. Word 2 contains the address the first I/O block on the I/O restore chain. You can use this address as a starting point for issuing EXCP for the I/O requests that you want to restore.

Note: If the program using the STAE macro instruction terminates via the EXIT macro instruction, the EXIT routine cancels all SCBs related to the terminating program. If the program terminates via the XCTL macro instruction, the EXIT routine cancels all SCBs related to the terminating program except those SCBs that were created with the XCTL=YES option. If the program terminates by any other means, the terminating program must reinstate the previous SCB by canceling all SCBs related to the terminating program.

Output Separation

In the PCP, MFT, and MVT operating system configurations, the system output writer can use the output separator facility to write separation records prior to writing the output of each job. These separation records make it easy to identify and separate the various job outputs that are written contiguously on the same printer or card punch device.

This chapter describes the output separator that is supplied by IBM, and tells how to write your own. A separate section describes the differences between separators for the MFT and MVT configurations and the PCP configuration. Before reading this chapter, you should be familiar with the information contained in the prerequisite publications listed below:

PREREQUISITE PUBLICATIONS

The IBM System/360 Operating System: Assembler Language publication (GC28-6514) contains the information necessary to code programs in the assembler language.

The IBM System/360 Operating System: Data Management Services publication (GC28-3746) describes the queued sequential access method (QSAM) used by the system output writer, and discusses program linkage conventions.

The IBM System/360 Operating System: Supervisor and Data Management Macro Instructions publication (GC28-6647) describes the system macro instructions that can be used in programs coded in the assembler language.

Output Separation – MFT, MVT

In MFT and MVT, both the system output writer and the direct SYSOUT writer may be used by a problem program to channel its output eventually to a printer or punch. When this is done, however, the system output stream goes uninterruptedly from one job to another, making it difficult to separate the output of one job from that of another, unless output separation is provided for.

The output separator facility of the operating system provides a means of identifying and separating the output of various jobs processed by the same output unit. To do this, the separator writes separation records to the system output data set prior to the writing of each job's output.

You can use the output separator that is supplied by IBM, or you can create and use your own output separator programs.

Using an Output Separator

The output separator function operates under control of both the system output writer and the direct SYSOUT writer. To use the function, the separator program must reside in the link library (SYS1.LINKLIB), and its name must be included as a parameter in either of the output writer procedures (the second part of the PARM field in the EXEC statement). Cataloged procedures for both writers are fully described in another chapter of this publication.

Functions of the IBM Output Separator

The IBM-supplied output separator resides in the link library (SYS1.LINKLIB). When its name, IEFSD094, is specified as a parameter in an output writer cataloged procedure, that output writer uses it to separate job output. The type of separation provided by the separator depends on whether the output is punch-destined or printer-destined.

Punch-Destined Output

For punch-destined output, the IBM-supplied separator provides three specially punched cards (deposited in stacker 1) prior to the punched card output of each job. Each of these separator cards is punched in the following format:

Columns 1 to 35	--	blanks
Columns 36 to 43	--	jobname
Columns 44 to 45	--	blanks
Column 46	--	output classname
Columns 47 to 80	--	blanks

Table 6 shows the operator commands that are affected by the aaaa parameter in an MCS environment. The commands are grouped by function. If the command is in a group authorized by the aaaa parameter, it is processed. If the command is not authorized by the aaaa parameter, it is ignored and an error message is sent to the master console.

Note: Informational commands (Group 0) are always valid when entered into the input stream.

Table 6. Operator Command Groups

Command Group	Function	Commands		
0	Informational	BRDCST	LOG	REPLY
		DISPLAY	MSG	SHOW
1	System Control	CANCEL	MODIFY	SET
		CENOUT	QUIESCE	START
		DEFINE	RELEASE	STOP
		HALT	RESET	USERID
		HOLD		WRITELOG
2	I/O Control	MOUNT	UNLOAD	VARY *
3	Console Control	VARY *		
1,2,3	Master Console	All commands are valid, plus VARY MSTCONS VARY HARDCPY VARY CPU VARY STOR VARY CH		

Note: VARY (Group 2) is accepted only to VARY a non-console device online or offline. VARY (Group 3) provides only for console switching and console reconfiguration or secondary consoles.

Bit settings for the aaaa parameter are:

Bytes	Bits	Bit Settings	Meaning
0	0	1	Group 1 commands executed
(aa)	1	1	Group 2 commands executed
	2	1	group 3 commands executed
	3-7	00000	Reserved
1 (aa)	0-7	00000000	Reserved

Example: If you wish to authorize commands from command groups 2 and 3 to be executed when entered into the input stream, code the aaaa parameter: "6000"

ef
MSGLEVEL value in absence of a value in the JOB statement.

If there is no MSGLEVEL= parameter in the JOB statement, job control statements and allocation/termination messages are recorded in the system output data set according to the value of the ef parameter. The values and their effects are:

e
Kinds of job control statements recorded.
0 - JOB statement only.
1 - Input statements, cataloged procedure statements, and symbolic parameter substitution values.
2 - Input statements only.

A blank defaults to a value of 0.

f
Kinds of allocation/termination messages recorded.
0 - None, except in the case of an ABEND condition. (In that event, all messages are recorded.)
1 - All.

A blank defaults to a value of 1.

h
MSGCLASS Default Value (A-Z, 0-9).

If there is no MSGCLASS keyword parameter in the JOB statement, job control statements and allocation/termination messages are recorded according to the message class specified by this character. If the character is blank or absent, A is the default class.

DD Statement for the Input Stream

Your procedure for the reader/interpreter must include a DD statement that describes the input stream. The format for this statement is:

```
//IEFRDER  DD  UNIT=device,LABEL=(,type),                                X

//             VOLUME=SER=SYSIN,                                         X

//             DCB=(list of attributes)[,DSNAME=name,DISP=OLD]
```

This statement must be named IEFRDER, as shown. The IEFRDER statement can be overridden with a START command. The parameter requirements are as follows:

UNIT=device
specifies the device from which the input stream is read. This can be any device supported by the queued sequential access method (QSAM). The device can be specified by its address, type, or group.

LABEL=(,type)
describes the data set label (needed only for tape data sets). If this parameter is omitted, a standard label is assumed.

Note: Label types AL and AUL (American National Standard label types) should not be used.

VOLUME=SER=SYSIN
: specifies the volume containing the input stream. This parameter is required for magnetic tape or direct access volumes. The serial SYSIN is recommended for identification of this volume, but other serials can be used.

Note: The volume serial numbers should not identify a volume that contains a data set written in ASCII.

DCB=(list of attributes)
: specifies the characteristics of the input stream and the buffers. If the BLKSIZE, LRECL, and BUFL subparameters are not specified, an 80-byte value is assigned to each. In MFT, if the procedure is going to be used for transient readers, the input must be unblocked 80 byte records. Other subparameter fields may be specified as needed; otherwise, the QSAM default attributes are assigned, as follows:

BUFNO -- two buffers. (In MFT if the procedure is to be used for transient readers, BUFNO=1 must be specified.)

RECFM -- U-format, with no control characters.

TRTCH -- odd parity, no data conversion, and no translation.

DEN -- lowest density.

PROC

DSNAME=name,DISP=disposition
: specifies the name and disposition of the input stream data set to be read, this keyword should be used only with direct access input stream.

DISP=OLD
: specifies that the input stream is an existing data set.

Note: OPTCD = Q should not be coded.

DD Statement for the Procedure Library

Your procedure for the reader/interpreter must include a DD statement that defines the procedure library. This statement must follow the IEFRDER statement which describes the input stream. The format for this statement is:

```
//IEFPDSI   DD    DSNAME=SYS1.PROCLIB,DISP=SHR
```

This statement must be named IEFPDSI, as shown. The parameter requirements are as follows:

DSNAME=SYS1.PROCLIB
: identifies the procedure library. To concatenate other data sets with the system library, you may follow the IEFPDSI DD statement with other unnamed DD statements thus expanding the system procedure library.

DISP=SHR
: specifies that the procedure library is an existing data set and can be shared with other tasks.

DD Statement for the CPP Data Set

Your procedure for the reader/interpreter must include a DD statement that defines the CPP (concurrent peripheral processing) data set. Two

DCB parameters (BLKSIZE, and buffer number) may be overridden by parameters in the input stream on DD* and DD DATA statements. The CPP data set is used for intermediate storage of input stream data. The format for this statement is:

```
//IEFDATA  DD    UNIT=device,                                    X


//               SPACE=(units,(quantities),RLSE,CONTIG),         X


//               VOLUME=SER=volser,DISP=(status,disp),           X


//               DCB=(list of attributes)
```

This statement must be named IEFDATA, as shown. The parameter requirements are as follows:

UNIT=device
specifies one or more direct access devices on which data sets from the input stream will be written. If more than one device is provided, the different data sets are not necessarily written in a continuous manner from device to device. Instead, the different data sets might be "spread" among the available devices in accordance with a reader/interpreter algorithm that is based on priorities and optimum access. If you want all the input stream data sets written on the same device, use the VOLUME parameter in this DD statement to identify the specific volume. The DEFER option must not be used.

CAUTION: Do not use UNIT group names unless the request is for no more than one device, or the group is defined to have devices of only one type.

SPACE=(units,(quantities),RLSE,CONTIG)
specifies space allocation for the direct access volume. The RLSE subparameter releases all unused space to the system when the data set is closed. The CONTIG subparameter ensures that space is allocated in contiguous tracks or cylinders.

Note: The first space allocation made by the system will be for the reader/interpreter program itself, which does not need or use the space.

VOLUME=SER=volser
identifies a specific direct access volume. This parameter is not required, but you can use it to cause all input stream data sets to be written on the same volume. You should also use this parameter if you specify the DISP parameter.

DISP=(status,disp)
specifies the status and disposition of the CPP data set. This parameter is not required, but can be used to bypass the first space allocation (as explained above). To do this, specify the parameter as DISP=OLD. The system then assumes that the data set exists, and does not allocate space for the reader/interpreter program.. Subsequently, the reader/interpreter forces a DISP=NEW,PASS status for the CPP data set so that space is allocated on it for recording the input stream data sets.

How to Dedicate a Data Set

You dedicate a data set by adding a DD statement (for each data set to be dedicated) to the initiator procedure. The unit must be a DASD; the space may be for a sequential or partitioned data set. (See the publication Storage Estimates, the chapter Job Step Initiation Requirement, for details on the number of DD statements per initiator.) Each DD statement must be of the form:

```
//ddname  DD  UNIT=unitparms,VOL=volparms,
              SPACE=(kind,(amount,increment,dirblks)),DISP=(new,delete)
```

ddname

A user supplied ddname must be given to identify the DD statement. The ddname is used (in the form DSNAME=&ddname) in the DD statement of the problem program job step which is to make use of the dedicated data set.

unitparms

Parameters that describe the unit to be used for the dedicated data set. The unit must be a DASD. The AFF= and DEFER unit parameters may not be used. The unit parameters specified here override those of the job step DD statement for which the dedicated data set is used.

volparms

Volume parameters.
A volume may be specified for each unit specified in the preceding unit parameter entry. The volume parameters specified here override those of the job step DD statement for which the dedicated data set is used.

(kind,(amount,increment,dirblks))

Type and size of space (in terms of CYL, TRK, avgbl, or ABSTR) to be allocated to the data set. If ,dirblks is omitted, the data set request implies sequential organization. If ,,dirblks is used, the data set request implies partitioned organization. If the dedicated data set is going to reside on an IBM 2301, or 2303 drum storage device, do not request space in cylinders.

When a dedicated data set with partitioned organization reaches an EOV condition, the initiator must be restarted. The DD statement in the problem program job step that is to use a dedicated data set must describe a problem program data set of the same organization as the dedicated one. Increments, once allocated, remain allocated until the initiator stops.

new,delete

These disposition parameters may either be coded explicitly or may take effect by default, that is by omitting the DISP= entry.

The effect of new is that the data set is freshly allocated from any available space on the volume, each time a Start Initiator operator command is used or the system is restarted.

The effect of delete is that the data set is not kept when the initiator is stopped and the space is available for reallocation to other jobs.

DSNAME

The allocation procedure for an initiator pre-allocated data set is the same as for any temporary data set. This procedure is simplest with no dsname= entry in the DD statement. That results in a

system assigned data set name of the form:
SYSnumber.Rnumber.procname.RVnumber.

You may also code DSNAME=&name, DSNAME=&&name, or DSNAME=name. These names will override those used in the job step DD statement for which the dedicated data set is used.

DCB parameters:
DCB parameters specified here have no effect.

How to Get to Use a Dedicated Data Set

If you want a dedicated data set to be used for a data set needed temporarily in a job step, define the temporary data set in a DD statement of the form:

```
//ddname   DD  DSNAME=&ddname,

//             SPACE=(avgbl,(amount,increment,dirblks)),

//             UNIT=unitparms,DISP=(new,delete),DCB=dcbparms
```

&ddname
name of the DD statement for the dedicated data set, preceded by an & sign.

(avgbl,(numbr,increment,dirblks))
Space request, in terms of average block length only, needed for this temporary data set.

An attempt to allocate the dedicated data set will be replaced by the normal allocation procedure if one of the following conditions is encountered:

- If the total space (primary and increments) requested here exceeds the total space (primary and increments) available to the dedicated data set.
- If the use of ,dirblks (presence or absence) differs from that in the DD statement of the dedicated data set, (or if ISAM is specified).
- If the space for ,dirblks requested here exceeds the space for ,dirblks specified in the dedicated data set.
- If the space request is shown in other than average block length.

unitparms
Unit parameters
Parameters that describe the unit to be used for the temporary data set, if the dedicated data set is NOT used. Here, the unit may be a magnetic tape unit, as well as a DASD.

(new,delete)
These disposition parameters must either be coded explicitly or may take effect through default.

dcbparms
DCB parameters required for your temporary data set. Unless specified, you may find that a previous user has left the dedicated data set with undesired DCB parameters.

PROCEDURE INITD

Language processor programs (such as FORTRAN compilers) make much use of temporary data sets. To permit ready use of the dedicated data set feature with IBM-supplied processor procedures, IBM supplies the initiator procedure INITD. (It becomes part of the system by including it in the SYS1.PROCLIB at system generation time.)

INITD is an initiator procedure that dedicates five utility data sets commonly used with IBM-supplied processor procedures. To use the dedicated data set facility with these procedures start the INITD initiator.

Before including the INITD procedure in your system, review the space allocations, unit specifications, and ddnames used in the procedure against your requirements. If they are significantly different, you may wish to code your own.

Presently existing procedures can be used under the INITD initiator without changes. Procedures designed for the dedicated data set feature remain operative without the presence of the dedicated data set feature. In short, the procedure will run under any initiator regardless of whether that initiator has dedicated data sets.

The INITD procedure looks as follows:

Procedure: INITD

```
//IEFPROC EXEC PGM=IEFIIC,PARM='A,LIMIT=13'

//SYSUT1 DD DSNAME=&UT1,SPACE=(1700,(200,100),,CONTIG),UNIT=SYSDA

//SYSUT2 DD DSNAME=&UT2,SPACE=(1700,(200,100)),UNIT=(SYSDA,SEP=SYSUT1)

//SYSUT3 DD DSNAME=&UT3,SPACE=(1700,(200,100)),                      C

//          UNIT=(SYSDA,SEP=(SYSUT1,SYSUT2))

//SYSUT4 DD DSNAME=&UT4,SPACE=(460,(700,100)),                       X

//          UNIT=(SYSDA,SEP=(SYSUT1,SYSUT2,SYSUT3))

//LOADSET DD DSNAME=&LOADSET,UNIT=(SYSDA,SEP=SYSUT1),                X

//          SPACE=(3600,(100,10))
```

INITD Procedure Statements

Each of the statements shown in the preceding illustration is explained in detail in the following. In addition to describing the reason for or effect of the use of a parameter, the description distinguishes between those parameters that must be coded as shown and those that you may override or substitute for.

The EXEC Statement

The EXEC statement for the procedure is:

```
//IEFPROC  EXEC  PGM=IEFIIC,PARM='A,LIMIT=13'
```

IEFPROC
The step name. Must be coded as shown.

EXEC
The job control statement name. Must be coded as shown. Defines the beginning of a job step.

PGM=IEFIIC
The program to be executed in this job step. IEFSD060 is the name of the initiator program. Must be coded as shown. Whether dedicated data sets are used depends on the DD statements that follow, not on the name of the program.

PARM='A,LIMIT=13'
Parameter list for the initiator program. A is the class of jobs to be processed, LIMIT=13 is the dispatch priority limit for this initiator. Both of these values can be overridden by values used with the Start operator command for the initiator.

The explicit data set reference (DSNAME=SYS1.COBLIB) requires a search of the catalog in each job step using the procedure. To save the repeated catalog search, move the DD statement to the initiator procedure and replace it in the COBECLG procedure with a DD statement in which the DSNAME=&name entry refers to the ddname of the dedicated data set. Allocation treats this as a dedication request, dedicated if so found. The new DD statement in the COBECLG procedure, after adding the present one to the initiator procedure, is:

```
//SYSLIB   DD  DSNAME=&SYSLIB,DISP=(SHR,KEEP)
```

The result is one catalog search per initiator start instead of one catalog search every job step. However, keep in mind that this COBECLG procedure requires the initiator with the dedicated data set. Using this modified procedure with an unmodified initiator will result in failure to allocate.

Disposition of Temporary Dedicated Data Sets

Allocation/termination routines do not delete temporary dedicated data sets at the end of each job step, but, instead, keeps them until the initiator stops; this occurs even if there is a specification of DISP=(NEW,DELETE) or DISP=(MOD,DELETE) on the DD statement for the data set. Therefore, if you attempt to use such a data set a second time in the same job, it will contain data from the previous use. This can be a problem if you are using cataloged procedures and run the same procedure twice within the same job. For example: assume that you use the procedure PL1LFLCLG twice within the same job and it uses a dedicated data set with a disposition of (MOD,PASS) for the compile step and (OLD,DELETE) for the linkage edit step. When the procedure is entered for the second time, the object module produced by the second compile step will be placed in back of the object module produced by the first compile step. Since both object modules are assigned identical names by the compiler, only the first will be linkage edited.

You can avoid this problem by not using dedicated data sets for jobs that run the same cataloged procedure twice. Alternatively, you could submit each cataloged procedure as separate jobs instead of submitting them as separate job steps within the same job.

You can use the following chart to determine the disposition, by allocation/termination, of temporary dedicated data sets.

If you code DISP=	Allocation/termination treats it as:
NEW	OLD
OLD/SHR	OLD
MOD	OLD
,DELETE	KEEP
,PASS	PASS
,KEEP	KEEP

Output Writer Procedures

A cataloged procedure for output writers requires two job control statements: an EXEC statement and a DD statement.

- An EXEC statement with the step name IEFPROC specifies the output writer program.
- A DD statement named IEFRDER defines the output data set. (In MVT, the attributes of the output data set must remain unchanged for a deferred checkpoint restart if the data set was opened but not completely written. The extents and number of extents do not have to be the same.)

SYSTEM OUTPUT WRITER

The standard output writer procedure supplied by IBM is named WTR. The standard procedure is:

```
                          Procedure:  WTR

//IEFPROC   EXEC  PGM=IEFSD080,REGION=20K,                          X

//                PARM='PA'

//IEFRDER   DD    UNIT=1403,VOLUME=(,,,35),                         X

//                DSNAME=SYSOUT,DISP=(NEW,KEEP),                    X

//                DCB=(BLKSIZE=133,LRECL=133,BUFL=133,              X

//                BUFNO=2,RECFM=FM)
```

PROCEDURE REQUIREMENTS

When creating your own output writer procedure, you must conform to the procedure format and the statement requirements. Use the IBM-supplied procedure as an example. The statement requirements are explained individually in the following paragraphs.

The EXEC Statement

The EXEC statement specifies the output writer program and its region size. It also passes a set of parameters to the output writer program. The format for the EXEC statement is:

```
//IEFPROC   EXEC  PGM=IEFSD080,REGION=nnnnnK,                       X
//                PARM='cxxxxxxxx,seprname'
```

The step name must be IEFPROC, as shown. The parameter requirements are as follows:

PGM=IEFSD080
: specifies the output writer program. The name of the program must be IEFSD080, as shown.

REGION=nnnnnK (MVT configurations only)
: specifies the region size for the output writer. The value nnnnn represents a number from one to five digits that is multiplied by K (K=1024 bytes) to designate the region size. The region

The procedure supplied by IBM is named DSO and is described in the following. If you wish to create your own procedure, follow its format.

```
                         Procedure:  DSO

//IEFPROC   EXEC  PGM=IEFDSO,REGION=8K,PARM=(PA,,A)

//IEFRDER   DD    UNIT=2400,DSN=SYSOUT,DISP=(NEW,KEEP),LABEL=(,SL),

                  VOL=(,,,05),DCB=(BUFNO=3)
```

The EXEC Statement

The EXEC statement specifies the direct SYSOUT writer and the space it requires to start in MVT. It is also used to give the writer program necessary operating information.

```
//IEFPROC   EXEC  PGM=IEFDSO,REGION=8K,PARM=(cx,,jjjjjjjj)
```

PRO

IEFPROC
     Name of the EXEC statement.
     Required as shown.

IEFDSO
     Name of the writer program.

REGION=8K
     Space required by IEFDSO to start in MVT.

PARM=
     Information for the IEFDSO program.

c
     A letter, P for printer or C for card punch, that describes the ultimate hard-copy medium. Must be given.

x
     The SYSOUT class to be processed.
     If stated here, and in the Start command, the latter rules.
     If not stated here, must be given in the Start command.

,jjjjjjjj
     Jobclasses to be processed.
     From zero to eight letters (A - O) showing the job classes to be processed.
     If any job classes are named in the Start command, they overrule all stated here.
     In MFT, if none are named here, then the job classes will be those assigned to the partition for which this writer is started.
     In MVT, if none are named here they must be given in the Start command.

The DD statement

This DD statement describes the kind of volume to be used and the format of the data set.

```
//IEFRDER  DD  UNIT=name,DSN=anyname,DISP=(NEW,KEEP),LABEL=(,SL),
               VOL=(,,,volcount),DCB=(list)
```

IEFRDER

Name of the DD statement.
Required as shown for IEFDSO.

name

Any form of unit identification may be used, for example, 00E, 2400, or TAPE.
Multiple parallel units (UNIT=2400,2) cannot be used.

DSN=anyname

Name of a non-temporary data set.
A name must be given.
If stated here and in the Start command also, the latter rules.
The name is used in the disposition messages at step termination, and must be used to identify the data set if it is to be printed later from tape.

DISP=(NEW,KEEP)

Required disposition.

LABEL=(,SL)

If DSO is being used to write to magnetic tape, standard label tapes are required. The label description may be stated explicitly or may be omitted, in which case SL is assumed.

,,,volcount

1 - 225.
The maximum number of volumes a data set to be processed by this writer will have.
Determines the amount of job queue space allocated to each SYSOUT data set processed by this writer. After the first 5 volumes, each subsequent 15 require another job queue record.
If omitted, 1 is assumed.
If stated here and also in the Start command, the latter rules.
This value cannot be given in a DD statement of a job to be processed.

list

The following DCB parameters gain control only if they are not also given in the SYSOUT DD statement or in the DCB macro instruction (that is, default values can be stated in this procedure):

BFALN, BFTEK, BUFL, BUFNO, BLKSIZE, LRECL, RECFM, NCP, HIARCHY, UCS.

The following DCB parameters, if stated here, override all except those given in a Start command:

CODE, DEN, MODE, OPTCD, PRTSP, STACK, TRTCH.

The Shared Direct-Access Device Option

This chapter describes the Shared Direct Access Storage Device option (Shared DASD) of the System/360 Operating System. It describes the functions of the option, its operating environment, and volume acceptability. Sections also explain operating procedures and data set considerations that the systems programmer must be aware of in using the option. An appendix to the chapter describes a procedure for finding unit control block addresses necessary for using the RESERVE macro instruction: it also shows an assembler language subroutine that issues a RESERVE and can be called by a higher level language.

The IBM System/360 Operating System: Operator's Guide, GC28-6540 provides information on operator responsibility when the Shared DASD facility is being used in a system; this should be read before using the Shared DASD option.

The IBM System/360 Operating System: Concepts and Facilities, GC28-6535 discusses the purposes of the Shared DASD facility.

The IBM System/360 Operating System: Storage Estimates, GC28-6551 provides information on the storage requirements for the option.

SHRI

IBM System/360 Operating System: System Generation, GC28-6554 explains how the option is included in a system. Supervisor and Data Management Macro Instructions, GC28-6647 provides information on the use of the DEQ macro instruction.

The Shared Direct-Access Device Option

The Shared DASD option allows computing systems to share direct access storage devices. Systems can share common data and consolidate data when necessary; no change to existing records, data sets, or volumes is necessary to use the facility. However, reorganization of volumes may be desirable to achieve better performance. Briefly, the sharing is accomplished by a two-channel switch which allows a shared control unit to be switched between two channels from different systems. (With certain hardware configurations sharing between a maximum of four systems is possible.) The switching is controlled by program use of the RESERVE macro instruction which reserves a shared device or volume for the use of one system until it is freed by the program's issuing a DEQ macro instruction. If a RESERVE macro instruction is used before the system in which the macro instruction is used has access to the shared device, the macro instruction will take effect only after the system gains access to the device.

The Shared DASD facility can only be included in a system at system generation time. This facility is shown diagrammatically in Figure 10.

SYSTEM CONFIGURATION

The Shared DASD option can be used with any combination of PCP, MFT, and MVT configurations of the operating system, excluding MVT with Model 65 multiprocessing (M65MP). Identical operating system configurations are not necessary for systems to share devices unless they share the system data set SYS1.LINKLIB. The option requires no additional equipment except the two-channel switch or the IBM 2844 Auxiliary Storage Control unit, which does not require the two-channel switch. Any of your installation's applications data sets can be shared; SYSCTLG can be shared when it does not reside on a systems residence volume. The following system data sets cannot be shared:

SYS1.SVCLIB
SYS1.NUCLEUS
SYS1.LOGREC
SYS1.SYSVLOGX (MFT and MVT)
SYS1.SYSVLOGY (MFT and MVT)

SYS1.SYSJOBQE
PASSWORD data set
SYSCTLG (on system residence volume)
SYS1.ROLLOUT
SYS1.ACCT
SYS1.MANX
SYS1.MANY

DEVICES THAT CAN BE SHARED

The following control units and devices are supported by the Shared DASD option:

1. IBM 2841 Storage Control Unit equipped with two-channel switch -- IBM 2311 Disk Storage Drive, 2303 Drum Storage, and 2321 Data Cell.
2. IBM 2314 Direct Access Storage Facility equipped with the two-channel switch -- IBM 2314 Disk Storage Module.
3. IBM 2314 Direct Access Storage Facility combined with the IBM 2844 Auxiliary Storage Control -- IBM Disk Storage Module. Device reservation and release are supported by this combination with or without the presence of the two-channel switch. Two channels -- one from System A and one from System B -- may be connected to the combination. In addition, the two-channel switch may be installed in either or both of the control units, thus permitting as many as four systems to share the devices.
4. IBM 2820 Control Unit with two-channel switch -- IBM 2301 Drum Storage.

Alternate channels to a device from any one system may only be specified for the IBM 2314 Direct Access Storage Facility.

Index

Indexes to systems reference library manuals are consolidated in the publication IBM System/360 Operating System: Systems Reference Library Master Index, Order GC28-6644. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

INDX

INDX

INDX

READER'S COMMENT FORM

IBM System/360 Operating System
System Programmer's Guide

Order No. GC28-6550-8

- Is the material:

	Yes	No
Easy to read?	□	□
Well organized?	□	□
Complete?	□	□
Well illustrated?	□	□
Accurate?	□	□
Suitable for its intended audience?	□	□

- How did you use this publication?

  □ As an introduction to the subject
  □ For additional knowledge

  Other ....

- Please check the items that describe your position:

  □ Customer personnel
  □ IBM personnel
  □ Manager
  □ Systems Analyst
  □ Operator
  □ Programmer
  □ Customer Engineer
  □ Instructor
  □ Sales Representative
  □ Systems Engineer
  □ Trainee
  Other ....

- Please check specific criticism(s), give page number(s), and explain below:

  □ Clarification on page(s) ....
  □ Addition on page(s) ....
  □ Deletion on page(s) ....
  □ Error on page(s) ....

Explanation:

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Cut Along Line

Fold Fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY ...

IBM Corporation
P.O. Box 390
Poughkeepsie, N.Y. 12602

Attention: Programming Systems Publications
Department D58

Fold Fold

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]