GC28-0627-0
File No. S370-36

Systems

OS/VS2 System Programming Library: Job Management

VS2 Release 3

IBM

First Edition (February, 1975)

This edition applies to release 3 of OS/VS2 and to all subsequent releases of VS2 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 **Virtual Storage Supplement to IBM System/360 and System/370 Bibliography**, GA20-0001, for the editions that are applicable and current.

This edition, with **OS/VS2 System Programming Library: Supervisor**, GC28-0628, and **OS/VS2 System Programming Library: TSO**, GC28-0629, obsoletes **OS/VS2 System Programming Library: Job Management, Supervisor, and TSO**, GC28-0682.

JES3 and Mass Storage Systems information contained in this publication is for planning purposes only until the availability of the products.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Systems Publications, Department D58, Building 706-2, PO Box 390, Poughkeepsie, N.Y. 12602. Comments become the property of IBM.

Preface

This publication describes job management facilities that can be influenced by the system programmer.

Part I, the allocation section, discusses volume attributes, considerations in allocating direct access storage devices, and the function of dynamic allocation. Dynamic allocation can be invoked through DYNALLOC macro instruction or through dynamic allocation interface routine (DAIR). This publication discusses the DYNALLOC macro instruction and its associated parameter structure, and reference information for the parameter structure.

Part II, the JES2 section, describes those aspects of JES2 processing that can be affected during system generation, during system initialization, or by user programming. The discussion is intended to tell the system programmer how he can affect JES2 processing but not to tell him how to code the system generation macro instructions and other initialization parameters.

Part III, the miscellaneous job management section, discusses restarting support, assigning special program properties, limiting user region size, changing the system log processing, updating MSTRJCL data set, and the external writer.

JES3 and Mass Storage Systems information contained in this publication is for planning purposes only until the availability of the products.

Publications referenced:

- *OS/VS2 System Programming Library: Initialization and Tuning Guide*, GC28-0681.
- *OS/VS2 System Programming Library: System Generation Reference*, GC26-3792.
- *OS/VS2 Scheduler and Supervisor Logic*, SY28-0624,SY28-0625,SY28-0626. (3 volumes)
- *OS/VS2 JCL*, GC28-0692.
- *OS/VS Data Management Macro Instructions*, GC26-3793.
- *OS/VS Checkpoint/Restart*, GC26-3784.
- *OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor*, GC28-0648.
- *OS/VS2 IBM 3540 Programmer's Reference*, GC24-5111.
- *OS/VS2 System Programming Library: Service Aids*, GC28-0674.

Contents

Figures

Summary of Amendments for GC28-0627-0 OS/VS2 Release 3

This edition, with **OS/VS2 System Programming Library: Supervisor**, GC28-0628, and **OS/VS2 System Programming Library: TSO**, GC28-0629, obsoletes **OS/VS2 System Programming Library: Job Management, Supervisor, and TSO**, GC28-0682-0.

Multi-Access Spool

The operation of single systems operating as members of a multi-access spool complex is discussed.

MSS

New return codes 0248, 049C, 04A0, 0498, and text unit 005E are discussed.

Miscellaneous

Allocation Services
Dynamic Allocation
Program Properties Table
IEALIMIT
External Writer
Controlling System Output

Part I: Allocation

Allocation Services: This topic describes how volume mount attributes (permanently resident, reserved, and removable) and volume use attributes (public, private, and storage) affect allocation of a volume. It also describes how the system satisfies specific and nonspecific volume requests.

Dynamic Allocation: Executing programs can allocate, unallocate, concatenate, and deconcatenate data sets. This topic describes these facilities and explains how to invoke them. It also describes how to write a routine to validate dynamic allocation requests.

Dynamic Allocation Parameter Structure Fields: This section explains informational and error reason codes and describes keys used to specify dynamic allocation requests.

The allocation routines assign units, volumes, and data sets in support of job processing. They allocate resources in response to JCL DD statements at step initialization, and they also permit data set allocation for jobs in progress (dynamic allocation), a capability formerly available only to TSO users.

The considerations and rules for coding DD statements are in *OS/VS2 JCL*, GC28-0692.

A discussion of how the design of the allocation routines relates to their performance is in *OS/VS2 System Programming Library: Initialization and Tuning Guide*, GC28-0681. This publication is of particular value to the installation system programmer interested in suggesting coding practices and selecting procedures to maximize allocation efficiency. Allocation treats MSS devices (3330V) as direct access storage devices.

How Allocation Satisfies Requests

The allocation routines attempt to improve system throughput by filling allocation requests in a way that results in as little serialization as possible. There are two types of serialization to be considered. Allocation must serialize to ensure that the status of the devices eligible for allocation remains static while devices are being selected. Also, some devices must be used in a serial manner.

The allocation routines try to satisfy requests in this order (from least serialized to most serialized):

1. allocating data set requests that require no specific units or volumes, for example, dummy, VIO, and SYSIN/SYSOUT data sets (not serialized.)
2. allocating data set requests to sharable units, that is, direct access units with permanently resident or reserved volumes mounted on them (not serialized.)
3. allocating teleprocessing devices (only serializes requested teleprocessing devices.)
4. allocating to mounted volumes and devices that do not need volumes (serialized on only the set of devices eligible to satisfy the request.)
5. allocating to online, unallocated devices that need volumes mounted by the operator or by MSS (serialized on only the set of devices eligible to satisfy the request.)
6. allocating all remaining requests, for example, requests that need offline devices and/or devices allocated to other jobs which can not be used concurrently (serialized on only the set of devices eligible to satisfy the request.)

Note: For items 4, 5, and 6 the order in which device types are serialized is controlled by the installation's Device Precedence List.

Volume Attributes

Volume attributes determine a volume's eligibility for demounting and volume sharing, and control the type of data set that can be allocated to the volume. (Volume sharing is the allocation of a volume to two or more data sets defined in the same job step, or the allocation of a direct access volume to two or more data sets defined in different job steps that are executing concurrently.)

The attributes that are assigned to tape and direct access volumes are the mount attribute and the use attribute. The nonsharable attribute may be assigned to direct access volumes only. The next two topics describe these attributes.

Mount and Use Attributes

Every volume is assigned a mount and use attribute either at IPL via a VATLIST or when first used by a job. The mount attribute controls volume demounting. The use attribute is one of the factors that controls allocation of mounted volumes to data set requests. The mount and use attributes are as follows:

- Mount
  - — Permanently resident
  - — Reserved
  - — Removable
- Use
  - — Public
  - — Private
  - — Storage

A private volume is one that can only be allocated when its volume serial numbers are explicitly or implicitly specified.

A public volume is a direct-access volume that is eligible for allocation of temporary data sets when no specific volume is requested and PRIVATE is not specified. It can also be allocated when its volume serial number is specified.

A storage volume is a direct-access volume that is eligible for allocation of both non-temporary and temporary data sets when no specific volume is requested and PRIVATE is not specified. Storage volumes usually contain non-temporary data sets, but temporary ones will be assigned to storage volumes if they cannot be assigned to public volumes.

The following points list the mount attributes and describe how the mount and use attributes get assigned to a volume:

- **Permanently resident** volumes cannot be demounted. Only direct access volumes can be permanently resident. Although the user may designate all direct access volumes as permanently resident in the "volume attribute list" (VATLSTxx) in SYS1.PARMLIB, the following volumes are always permanently resident:
  - — all volumes that cannot be physically demounted, such as drum storage volumes
  - — the IPL volume
  - — the volume containing the system data sets, such as SYS1.LINKLIB and SYS1.PROCLIB.

  An installation can assign a permanently resident volume the use attribute of public, private, or storage in the VATLST member of SYS1.PARMLIB; it is public by default.

- **Reserved volumes** remain mounted until the operator issues an UNLOAD command. Both direct access and tape volumes can be reserved volumes. A volume becomes reserved as a result of a MOUNT command or a VATLST entry (for direct access devices only). A volume is usually designated as a reserved volume to avoid repeated mounting and demounting of the volume when it is to be used by many jobs.

  An installation can assign a reserved *direct access* volume the use attribute of public, private, or storage. The use attribute is assigned to the volume either in the VATLST member of SYS1.PARMLIB or in the parameter of the MOUNT command, depending on how the volume becomes reserved.

  A reserved *tape* volume is always assigned the use attribute of private.

- **Removable volumes** are those that are neither permanently resident nor reserved. Removable volumes can be demounted either after the end of the job in which they are last used or when the unit on which the volume is mounted is needed for another volume.

  The use attribute of public or private can be assigned to a removable *direct access* volume as follows. The use attribute of public is assigned when the JCL PRIVATE volume subparameter is not coded. The use attribute of private is assigned when the PRIVATE volume subparameter is coded.

  A removable *tape* volume can be assigned the use attribute of public or private. The use attribute of public is assigned when the PRIVATE subparameter is not coded, a nonspecific volume request is made, and the data set is temporary (a system-generated data set name or a disposition of DELETE.) The use attribute of private is assigned when the PRIVATE subparameter is coded, a specific volume request is made, or the data set is nontemporary (a non system-generated data set name or a disposition other than DELETE.)

Figure 1 summarizes the type of volume that can be assigned to satisfy a specific or nonspecific volume request for a temporary or nontemporary data set; how these attributes are assigned; and how the volume is demounted.

Volume State	Temporary Data Set	Nontemporary Data Set	How Assigned	How Demounted
	Type of Volume Request			
Public/ Permanently Resident[1]	Nonspecific or Specific	Specific	VATLST Entry or by default	Always[2] mounted
Private/ Permanently Resident[1]	Specific	Specific	VATLST Entry	Always[2] mounted
Storage/ Permanently Resident[1]	Nonspecific or Specific	Nonspecific or Specific	VATLST Entry	Always[2] mounted
Public/ Reserved[1]	Nonspecific or Specific	Specific	VATLST Entry or MOUNT command	UNLOAD or VARY OFFLINE commands
Private/ Reserved (Tape and direct access)	Specific	Specific	VATLST Entry or MOUNT command (MOUNT command only for tape.)	UNLOAD or VARY OFFLINE commands
Storage/ Reserved[1]	Nonspecific or Specific	Nonspecific or Specific	VATLST Entry or MOUNT command	UNLOAD or VARY OFFLINE commands
Public/ Removable (Tape and direct access)	Nonspecific or Specific	Specific	VOLUME=PRIVATE is **not** coded on the DD statement. (A nonspecific request and a temporary data set for tape also causes this assignment.)	When unit is required by another volume.
Private/ Removable (Tape and direct access)	Specific	Specific	VOLUME=PRIVATE is coded on the DD statement. (Specific request or a nontemporary data set for tape also causes this assignment.)	At job termination or when the unit is required by another volume.

[1] Direct access volumes only.

[2] Note that a VARY OFFLINE effectively accomplishes dismount.

Figure 1. Combinations of Mount and Use Attributes

Nonsharable Attribute

Allocation assigns the nonsharable attribute to direct access volumes that may require demounting during a step's execution. When a volume has the nonsharable attribute, the volume cannot be assigned to any other data set until the nonsharable attribute is removed. It is removed at the end of the step that was using it as nonsharable.

The nonsharable attribute never gets assigned to a permanently resident or reserved volume. It is always assigned to a volume used to satisfy any of these requests:

- specific volume request that specifies more volumes that devices.
- a nonspecific volume request if it specifies PRIVATE and a volume count greater than the number of devices. For MSS devices, MSVGP may also be used in place of PRIVATE.
- a request for unit affinity with an earlier data set defined in the job step when the data sets reside on different volumes.
- a request for deferred mounting of the volume on which the data set resides.

Figure 2 shows the system action for sharable and nonsharable requests.

The Request is:	The Volume is Allocated:	
	Sharable	Nonsharable
Sharable	allocate the volume	wait[1]
Nonsharable	wait[1]	wait[1]
[1]The operator has the option of failing the request. The request will always fail if waiting is not allowed.		

Figure 2. Sharable and Nonsharable Volume Requests

As an example of when the nonsharable attribute is set, suppose JOBA has indicated a need for two volumes but only one unit is specified. In this case, the operator will later have to mount JOBA's second volume. JOBB requests to share the first volume mounted. If JOBA were to request the mounting of the second volume while JOBB were processing, JOBB would fail. To avoid this problem, the system marks JOBA's volume request as nonsharable so that no other job can use those volumes while JOBA is executing.

Satisfying Specific Volume Requests

In the following cases the system can satisfy a request for a specific volume that is already mounted:

- The volume is permanently resident or reserved. (The volume is assigned regardless of the requested use attribute, and the use attribute is not changed by the allocation.)
- The direct access volume is a removable volume that does not have the nonsharable attribute and is being used by a concurrently executing step. (If your request would make the volume nonsharable, the system waits to assign you that volume until all other job steps using the volume have terminated.)
- The direct access volume is removable but not allocated. The use attribute (private or public) assigned to the volume when it is allocated is determined by the presence or absence or the PRIVATE subparameter.
- The tape volume is a scratch volume and is not in use. The use attribute of private is assigned to the volume if the request is for a permanent data set or if PRIVATE is coded.

Figure 3 shows the affect on use attributes of the user's request.

The Request is:	The Volume is:	
	Private	Public
Private	stays private	changes to private
Public	stays private	stays public

Figure 3. Private and Public Volume Requests

Satisfying Nonspecific Volume Requests

There are four possible types of nonspecific volume requests:

- a private volume for a temporary data set
- a private volume for a nontemporary data set
- a non-private volume for a temporary data set
- a non-private volume for a nontemporary data set

The system satisfies these different types of requests as described below. Since the system satisfies the first two types of requests in the same way, these two requests are described together.

1. For a nonspecific volume request for a private direct access or tape volume, the system requests the operator to mount a volume. The operator should mount a volume whose space is unused; this gives the user control over all space on the volume. Once mounted, the volume is assigned the use attribute of private.
2. For a nonspecific volume request for a non-private direct access volume that is to contain a temporary data set, the system attempts to assign a public or storage volume that is already mounted, or, if no space is available, it requests the operator to mount a removable volume.

   If the system selects a mounted volume, its use attribute remains the same. If a removable volume is mounted, the system assigns it the use attribute of public.

   For a nonspecific volume request for a non-private *tape* volume that is to contain a temporary data set, the system assigns a scratch volume that is already mounted, or it request the operator to mount a tape volume. Once mounted, the system assigns the volume the use attribute of public.
3. For a nonspecific volume request for a non-private *direct access* volume that is to contain a nontemporary data set, the system assigns a storage volume if one is mounted on an eligible device. Otherwise, the system treats the request as a nonspecific volume request for a private volume.

   For a nonspecific volume request for a non-private *tape* volume that is to contain a nontemporary data set, the system treats the request as a nonspecific volume request for a private volume.

MSS Devices

There are six possible types of nonspecific volume requests for MSS devices:

- a private volume for a temporary data set
- a private volume for a nontemporary data set
- a private, MSS group volume for a temporary data set
- a private, MSS group volume for a nontemporary data set
- a non-private volume for a temporary data set
- a non-private volume for a nontemporary data set

The system satisfies these different types of requests as described below. (Coding MSVGP=grpname implies PRIVATE and identifies to the system an installation-defined group of MSS volumes.)

1. The system handles the first two types identically. It defaults to a private, MSS group request with a default name of SYSGROUP. If the installation has defined one or more volumes in this group, the system will select one with sufficient space to satisfy the request and cause the volume to be mounted.
2. The third and fourth requests are also handled in the same fashion. A volume from the specified group, which has sufficient space, will be selected and mounted.
3. For a nonspecific request for a non-private 3330V volume that is to contain a temporary data set, the system attempts to assign a public or storage 3330V volume that is already mounted. If none is mounted, the request defaults to SYSGROUP and is handled as in (1) above.

4. For a nonspecific request for a non-private 3330V volume that is to contain a nontemporary data set, the system assigns an already mounted storage 3330V volume. If none is mounted, the request defaults to SYSGROUP and is handled as in (1) above.

Determining Numbers of Volumes/Units per Request

Before assigning volumes and units for a job step or for an allocation request through dynamic allocation, the allocation routines must determine

- the maximum number of volumes per request.
- the maximum number of units per request.
- the number of units per job step.

The maximum numbers are calculated because more units than specified may actually be used. The rules for determining unit requirements are explained under the topic "Units per Job Step".

Volumes per Request

The maximum number of tape volumes or direct access volumes required to satisfy any request is the greater of

- the volume count specified in the VOLUME parameter.
- the number of volume serials available.

The number of volume serials available is one of the following:

- The number of volume serials specified.
- The number of volumes obtained through VOL=REF (only if VOL=REF was coded).
- The number of volume serials that the data set resided on when it was passed (only if the request is for an existing data set that was passed from a prior step, and neither volume serials nor VOL=REF was specified).
- The number of volume serials obtained from the catalog (only if the request is for an existing data set that was not passed from a prior step, and neither volume serials nor VOL=REF was specified).
- The number of volume serials minus the volume sequence number + 1 (only if the request is for an existing data set in which the volume sequence number specified is not greater than the number of volume serials). For example, if 8 volume serials are calculated to be used and a volume sequence number of 4 is specified, then the number of volume serials to be allocated would be 5 (8 - 4 + 1); in this case, the first three volume serials will be discarded, and the fourth volume would become the first volume allocated.
- The unit count specified in the UNIT parameter (only if the unit count specified is greater than the number of volume serials calculated in the previous statement, or if the request is for a new nonspecific direct access volume that does not specify VOLUME=PRIVATE).

When the required number of volume serials for a request is greater than the number of specific volume serials obtained through specified volume serial numbers, VOL=REF, from a passed data set, or from the catalog, the remainder of the volumes are assumed to be requests for nonspecific volumes.

Units per Request

The maximum number of tape units or direct access units required to satisfy any request is equal to the greater of

- the unit count specified in the UNIT parameter.
- the total number of volumes required (if parallel mounting is requested).

When UNIT=AFF is specified, the unit requirements are obtained from the referenced request. The number of units shared with the referenced request is the number of units used by the referenced request.

For direct access volumes, the number of units required to satisfy a request specifying a generation data group (GDG) name is dependent upon the unit requirements of each member of that GDG. Therefore, each member is handled as a single request.

For direct access volumes, the number of units required to satisfy a VSAM data set is dependent upon the unit/volume configuration of the data set. If the data set spans multiple device types, the total number of units required is determined by catalog management. Additional tables will then be generated by the scheduler to cause the allocation of the required number of units. For VSAM data sets, a specified unit count or parallel mount may be overridden by the system once the unit requirements for the data set are determined.

Units per Job Step

The number of units required for a job step is not necessarily the sum of the unit requirements for each request.

The following rules tend to reduce the total unit requirements for a step:

- A volume can only be allocated to one unit. Therefore, if more than one request asks for the same volume, all requests will get allocated the same unit.
- For direct access, storage and/or public requests can be allocated on the same volume. Therefore, two or more such requests may be satisfied with one unit.
- For tape, if VOL=REF is specified, more than one public request can be allocated on the same volume. Therefore, two or more public requests may be satisfied with one unit.

The following rules tend to increase the total unit requirements for a step:

- A permanently resident or reserved volume cannot be demounted. Therefore, a volume which is permanently resident or reserved will be assigned its own unit (where it is mounted) even if, through JCL specification, it was to share a unit with one or more other volumes.
- For direct access, when more than one request within a job step requires the same volume, that volume must be shared. Therefore, a direct access volume which is required by more than one request will be assigned its own unit even if, through JCL specification, it was to share a unit with one or more other volumes.
- For direct access, a VSAM data set will require additional units if the data set resides on more than one device type.
- For direct access, an additional unit is required for a private catalog volume if it is associated with and/or used to retrieve volume information about a particular data set.
- For direct access, when a GDG name is specified, additional units may be required to satisfy the device type requirements of each individual member of the GDG.
- For tape, when conflicting unit assignments are specified for tape volumes, the volume involved in the conflict will be assigned its own unit. For example, such a conflict would exist for VOLUM2 in the following DD statements:

```
//DD1    DD  UNIT=2400,VOL=SER=(VOLUM1,VOLUM2)
//DD2    DD  UNIT=2400,VOL=SER=(VOLUM2,VOLUM3)
```

In this case, three units, one for each volume, would be assigned. If the user had requested via unit affinity that the same tape unit be used for both DD1 and DD2, then only one unit would have been assigned.

The allocation performed in response to JCL at step allocation (or at LOGON for time sharing users) may be altered prior to step unallocation or LOGOFF by invoking dynamic allocation. Because device requirements may not be fully known prior to execution, dynamic allocation routines (SVC99) provide the facility to acquire resources as the need develops. It also allows resources to be used more efficiently, because they can be acquired just before use and/or released immediately after use. (The term "resource" means a ddname-data set combination with its attendant volumes and devices, if any.)

The term dynamic allocation not only refers to the allocation of resources, but also to all related functions. The major functions of dynamic allocation are:

- dynamic allocation — allocates a resource
- dynamic unallocation — unallocates a resource
- dynamic concatenation — concatenates allocated data sets
- dynamic deconcatenation — deconcatenates concatenated data sets
- dynamic information retrieval — provides retrieval of certain data set information.

A typical use for dynamic allocation is in a program that needs temporary use of a device, volume, or data set for which there is heavy contention. In such a case, dynamic allocation provides the means for a program to tie up the resource for only as long as necessary rather than for the life of the program.

Another common use for dynamic allocation is in a program whose need for allocation resources may vary according to the input. Dynamic allocation permits such programs to dynamically allocate and free only the files necessary to process the input, so the specific resources supporting the required files can be in use for the minimum time.

The SVC99 dynamic allocation routines can be invoked by both batch and time-sharing programs. It can be invoked in two ways - through the DYNALLOC macro instruction or through the dynamic allocation interface routine (DAIR), which was the only interface between time-sharing programs and dynamic allocation prior to MVS. When DYNALLOC is used, the request is specified in the parameter structure discussed under "Requesting Dynamic Allocation Functions." Requests via DAIR use a different parameter structure, but not all of the dynamic allocation options are supported through DAIR. The DAIR interface remains to provide compatibility.

The DAIRFAIL TSO service routine can be used to issue write-to-programmer or TSO PUTLINE failure messages for both DAIR and SVC99 error codes.

This publication discusses allocation functions available through DYNALLOC and how to request them. Refer to *OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor,* GC28-0648, for information on using DAIR and DAIRFAIL.

Allocations

A dynamic allocation user can request one of two types of dynamic allocation:

- allocation of a data set name. This type of allocation is the dynamic allocation equivalent of a JCL DD statement. (Included in this type of request are requests for allocation of a sysout data set, a terminal, a DUMMY data set, or a device, that is, a dsname does not have to be explicitly specified. A ddname may optionally be specified).
- allocation of a ddname. This type of request marks the resource currently associated with the specified ddname as in use. (A resource is considered in use after it has been dynamically allocated, dynamically concatenated, or opened.)

When dynamic allocation is invoked for either type of request, an allocation environment already exists for the user. This allocation environment consists of the user's step and dynamic allocation requests which have been previously processed and which have not been dynamically unallocated. Dynamic allocation considers these existing allocations. For dsname allocation requests, the dynamic allocation routines first check for environmental conflicts, then try to satisfy the request with an existing allocation that matches or can be made to match the request, and finally try a new allocation. If an existing allocation can be used, much of allocation is avoided. For ddname requests, the routines check if the specified ddname is still allocated and is not in use. This processing is described in detail below.

Checking for Environmental Conflicts

If the dsname allocation request is not valid with respect to the user's existing allocation environment, the request is failed. The following is a list of environmental conflicts which cause a request to be failed:

- the specified ddname is associated with an existing allocation which is in use.
- the specified ddname is associated with a group of concatenated data sets that the user defined to be permanently concatenated. (See the topic "The Permanently Concatenated Attribute" for a description of this attribute.)
- the specified ddname is associated with an existing allocation which does not have the convertible attribute (the convertible attribute means that some of its parameters can be changed), is not for the specified dsname, and cannot be changed, or which is for the specified dsname but cannot be used for one of the reasons described in the next topic.
- a status of NEW is specified with a non- & dsname that is the same as that associated with the existing allocation unless different volume serial numbers are specified.
- a status of OLD or SHR is specified for a dsname which is associated with an existing allocation that is not permanently allocated, not is use, and has a disposition of DELETE, unless specified volume serial numbers are different from those associated with the existing allocation. (A permanently allocated resource can be unallocated only when a user specifically requests unallocation or at job step unallocation.)

Using an Existing Allocation

When successive processes usually require the same resource, the overhead of releasing and reobtaining the resource can be avoided. For example, time-sharing command processors often use the same data sets. Therefore, the data sets are not dynamically unallocated at the end of the command process, but are designated "not in use". This avoids unallocation and reallocation processing.

An existing allocation can be used to satisfy only a request for the allocation of an explicitly specified dsname, a request for the allocation of the user's terminal as an I/O device, or a request for the allocation of a DUMMY data set. In addition, if any of the following are specified the request is not eligible to be satisfied by an existing allocation: data set sequence number, label type, unit description (unless the dsname is a & dsname, in which case the unit description is ignored), unit count or parallel mounting, volume sequence number, volume count, volume reference, private volume, or DCB reference. If MSVGP is specified, it will be ignored if an existing allocation is used to satisfy the request.

An existing allocation of the specified dsname, or terminal or DUMMY allocation must have the following properties to satisfy an eligible request:

- it must not be in use.
- it must not be a member of a concatenated group.
- it must have the same volume serial numbers as any that are explicitly specified in the request.
- it must be permanently allocated if it has a disposition of DELETE and the request specifies a status of MOD.

- it must not be a generation data group data set.
- it must have the convertible attribute or, if not, all of the following must be true. (Only the first requirement must be true for requests specifying an & dsname):
  — the request does not specify a ddname or the specified ddname matches the ddname associated with the existing allocation. A terminal request which does not specify a ddname cannot be satisfied by a non-convertible existing allocation.
  — the member name specified in the request is associated with the existing allocation or a member name is not specified in the request and no member name is associated with the existing allocation.
  — DCB parameters, Input Only, or Output Only are not specified in the request.
  — a status of MOD is either specified in the request and associated with the existing allocation, or is not specified and not associated with the existing allocation.
  — the request does not specify that only convertible existing allocations may be used to satisfy the request.
  — the request does not specify that the convertible attribute be assigned to the allocation.

If the specified ddname is associated with an existing allocation of the user which was not selected to satisfy the request, a ddname that is unique within the step is generated and associated with that existing allocation. This ddname consists of the characters 'SYS' followed by five digits.

Changing the Parameters of an Existing Allocation

When dynamic allocation uses an existing allocation to satisfy a request, some of the parameters of the existing allocation may have to be changed to reflect the parameters specified in the request. The only existing allocations that can have parameters changed are those allocated dynamically without the Permanently Allocated attribute or with the Permanently Allocated attribute and the Convertible attribute. Resources allocated via JCL or the TSO ALLOCATE command cannot have their parameters changed (with the exception of status and disposition specified via JCL), but they may be used if no changes are necessary.

Not all parameters of an existing allocation can be changed. The following parameters are eligible for change:

- ddname
- membername
- status
- normal disposition
- conditional disposition
- space
- unallocation at CLOSE
- input only
- output only
- DCB attributes
- password
- permanently allocated attribute

No others are eligible.

Choosing Among Satisfactory Existing Resources

If more than one existing allocation can satisfy the request, dynamic allocation selects:

- the existing allocation which is associated with the specified ddname or, if none,
- the existing allocation for which the in-use attribute has been most recently removed (data sets allocated via JCL are considered to have had their in use attributes removed at step allocation).

Allocation of a Ddname

This type of dynamic allocation request is used to determine if a particular ddname is associated with an existing allocation and, if so, to assign the in-use attribute to that existing allocation. To satisfy a ddname request, an existing allocation with the specified ddname must:

- not be in use
- not have the convertible attribute, or must be permanently concatenated, that is, must have properties that insure that the ddname could not have been disassociated from the existing allocation. (See the topic "The Permanently Concatenated Attribute" for a description of this attribute.)

If the existing allocation with the specified ddname does not meet the above requirements, or if the ddname is not associated with any existing allocation of the user, the request is failed and an error return is made to the user. If the existing allocation meets the above requirements, it is assigned the in-use attribute and the request has been satisfied. If the existing allocation is a member of a concatenated group, all members of the group are assigned the in-use attribute, so the entire group has been allocated.

The user may specify that an indication be returned if the existing allocation which satisfies the request is associated with a DUMMY data set.

New Allocations

A new allocation is attempted when an existing allocation cannot be used to satisfy the request. The dynamic allocation routines do not allow a new allocation while a job step is holding for possible reuse more dynamically allocated resources than it is permitted. The permitted number, called the control value, is equal to the value in the EXEC statement DYNAMNBR=n parameter plus the number of DD DYNAM statements and is the number of dynamically allocated resources which may be held for reuse. This limit is imposed because an existing allocation even when not in use, can prohibit other users from using its associated resources, for example, DASD space or a tape drive, or the data set itself.

The control value may be exceeded in this manner: requests for new allocations will be satisfied, up to the system limit of 1635 concurrent allocations. Some of these allocated resources may be later designated as not in use when they are no longer needed, and the number of resources so designated can exceed the control value.

When the dynamic allocation routines need to make a new allocation while the control value is exceeded, they try to automatically unallocate enough resources to meet the control value limit.

Unallocating Resources Held for Reuse

The only resources eligible for automatic unallocation are those which were allocated dynamically without the permanently allocated attribute. (Resources allocated through JCL, and through the time-sharing ALLOCATE command are not eligible because they have the permanently allocated attribute.)

When many resources are eligible for automatic unallocation, the dynamic allocation routines choose those which have been designated as not in use for the longest time. These are unallocated and the new allocation is processed.

If the control value is still exceeded after all eligible resources have been unallocated, then the request for a new allocation fails. In this case the user must explicitly request unallocation of an existing allocation before the new allocation can be performed.

Differences Between Step Allocation and Dynamic Allocation

A request for a new allocation is processed in a manner similar to the way requests to step allocation are processed. The following are differences in this processing:

- If a ddname is not specified in the request, a ddname unique within the step will be generated. This ddname will consist of the characters 'SYS' followed by five digits.
- If space information is not specified in a request for a new direct access data set, a default of a block length of 1000, 10 primary blocks, 50 secondary blocks, and release of unused space (RLSE) is used. These space defaults are contained in the Allocation Default CSECT, IEFAB445 (a member of load module IEFW21SD), so the installation can conveniently modify them. The contents of the module are (beginning at offset zero):
  - — three bytes for the binary value of the primary quantity (x'00000A')
  - — three bytes for the binary value of the secondary quantity (x'000032')
  - — three bytes for the binary value of the average block length (x'0003E8')
  - — three bytes for the binary value of the number of directory blocks (x'000000')
  - — one byte of flags with the following bit meanings:
    bit 0 - TRK (0)
    bit 1 - CYL (0)
    bit 2 - blocklength (1)
    bit 3 - RLSE (1)
    bit 4 - CONTIG (0)
    bit 5 - MXIG (0)
    bit 6 - ALX (0)
    bit 7 - ROUND (0)
- Passed Data Set Information is not used to retrieve volume information.
- ISAM data sets cannot be created through dynamic allocation.
- For time sharing users allocating new data sets, DSORG is defaulted to partitioned organization if a directory quantity is specified, or to physical sequential otherwise.
- If a unit description is not specified, a unit description is obtained from a time-sharing user's UADs entry. If the user is not a time-sharing user, or if the UADs entry does not contain a unit description, a default of 'SYSALLDA', that is, all direct access devices, is used. This default is contained in the allocation default CSECT IEFAB445, in the 8 bytes beginning at offset 13 (decimal). The unit description supplied is eligible to override the unit type for a cataloged data set; however, the default unit description from the UADS is not eligible for unit override.
- If the request was eligible to be satisfied by an existing allocation, but no existing allocation of the specified dsname could be used to satisfy the request, the volume and unit information associated with an existing allocation of the specified dsname is copied and associated with the request.
- The dynamic allocation routines will not wait for another user to release a data set in order to obtain use of it. If a conflict occurs, the request fails.

The following topics explain other differences between step and dynamic allocation.

Device and Volume Use

Dynamic allocation supports the same devices that are supported by step allocation. Devices and volumes are selected to satisfy dynamic allocation requests in the same manner as they are selected for step allocation requests.

However, dynamic allocation will not wait for devices or volumes to be released by other users to satisfy an allocation request. Rather, if a specified device or volume is currently unavailable, the request will be failed.

Mounting Volumes/Offline Devices

Dynamic allocation can bring devices online and have volumes mounted. Because this is a time consuming operation and requires operator communication, and therefore is not always desirable in an interactive environment, this function is an option for time sharing users. Installations can assign this option to time sharing users via the UADs entries. Non-time sharing users always have volume mounting ability and the ability to have devices brought online. However, any user may indicate in the dynamic allocation request that volumes are not to be mounted and that devices are not to be brought online for a request.

The operator may inform the dynamic allocation routines that a volume is not to be mounted or that a device is not to be brought online. In this case, the request is failed. In order to support this operator communication, dynamic allocation must wait for tape volumes to be mounted. (Step allocation does not wait for tape volumes to be mounted). If the volume is mounted, OPEN will verify that the correct volume has been mounted.

If the option to have volumes mounted and devices brought online is not in effect, then tape and direct access devices which have an outstanding mount request or which are not ready are not eligible for use by dynamic allocation.

Cataloging at Allocation

Direct access data sets that are dynamically allocated with a status of NEW, or MOD are treated as NEW, and a normal disposition of CATLG will be cataloged when allocated rather than when unallocated.

If the data set cannot be allocated, it will not be cataloged. If the data set cannot be cataloged, no allocation will take place. In either case an appropriate return code will be returned to the user.

Specifying the Data Set Password

A dynamic allocation user may specify, as part of his request, the password of a password protected data set. This allows bypassing of prompting when the data set is opened.

Returning Information

A dynamic allocation request can specify that the ddname, data set name, and volume serial numbers that are assigned be returned in the dynamic allocation parameter list. A user can also request the data set organization (DSORG) of the allocated data set. It will be returned as follows:

- If a DSORG is specified with the allocation request, that DSORG is returned.
- If the allocation request is for a terminal as an I/O device or for a SYSOUT data set, 'PS' is returned as a default value.
- If the allocation request is a tape data set, 'PS' is returned as a default value.
- If the allocation request is for a NEW direct access data set, 'PO' will be returned if a directory space quantity was specified; otherwise, 'PS' will be returned.
- If the allocation request is for an existing direct access data set, the data set organization obtained from the Data Set Control Block (DSCB) is returned. If the organization cannot be obtained from the DSCB, the allocation request is failed.
- For other types of allocation requests, zeroes will be returned.

Dynamic Unallocation

The dynamic unallocation routines provide the means of releasing resources when they are no longer needed. There are two functions available through dynamic unallocation:

- releasing a data set, which can involve the following processes:
  — disassociating the ddname from the data set name, which allows the ddname to be used in subsequent dynamic allocations
  — processing the data set disposition
  — releasing the data set for use by other jobs
  — freeing the unit(s) to which the data set was allocated
  — releasing the volume(s) on which the data set was allocated
- removing in-use attribute

Normal processing for the dynamic unallocation routines is to remove the in-use attribute from resources allocated through JCL, through the time-sharing ALLOCATE command, or dynamically with the permanently allocated option; the routines will release data sets allocated dynamically without the permanently allocation option. However, a user may explicitly specify the type of processing to be performed. The explicit specification will be satisfied in all but one case—the in-use attribute will not be removed from non-permanently allocated, non- & dsname data sets with a disposition of DELETE, because such a resource cannot be used to satisfy a subsequent request. Such a resource will be released.

Either dynamic unallocation function can be performed for a dsname or a ddname. The following rules apply to dsname unallocation requests:

- If no ddname is specified and the dsname is associated with more than one ddname, all associations are unallocated. If an error occurs while unallocating one ddname, processing continues for the others and and error code is returned. If errors occur for more than one ddname, the error code applies to the last ddname for which there was an error.
- If a membername is specified with the dsname, only those associations containing both the membername and dsname are unallocated.

The following rules apply to ddname unallocation requests:

- Only the occurrence of the data set associated with the specified ddname is unallocated, even if that data set is associated with other ddnames.
- If a dsname or a dsname and membername are specified in addition to a ddname, they must be associated with that ddname or the request fails.

Unallocating a Data Set

A data set is not unallocated if it is open, is a member of an open concatenated group, or is a private catalog. Also, if a data set, unit, or volume is associated with more than one ddname, it may not be available to other users until all ddnames are unallocated.

The topics below discuss unallocating concatenated data sets, SYSOUT data sets, and disposition processing.

Unallocating Concatenated Data Sets

If the concatenated group does not have the permanently concatenated attribute, the group is deconcatenated and the member associated with specified dsname is unallocated. (The first member is unallocated if a ddname is specified.)

If the group is permanently concatenated and a ddname without a dsname, a VSAM dsname (for data sets spanning device types), or GDG ALL (request for all members of a generation data group) dsname is specified, the entire group is unallocated. The unallocation request fails if any other dsname is specified.

Unallocating SYSOUT Data Sets

When a SYSOUT data set is unallocated it is immediately made available for output unless the user specifies an overriding disposition of DELETE. In this exceptional case, the information in the sysout data set is lost.

The output class, remote work station designation, and Hold/Nohold option may be specified; they will override those specified at allocation.

Disposition Processing

A disposition specified in an unallocation request will override the disposition specified at allocation. Overriding dispositions are ignored for passed data sets, VSAM data sets, and system-named data sets.

An overriding disposition of DELETE for a data set allocated as SHR is invalid; the request fails. A member of a partitioned data set cannot be deleted with a disposition of DELETE; the entire data set is deleted.

Removing the In-Use Attribute

A request to dynamic unallocation may be to remove the in-use attribute. Removing this attribute provides a resource that dynamic allocation can use to satisfy a subsequent allocation request of the same user. Removing the in-use attribute does not actually unallocate the associated resources.

Processing a Request to Remove the In-Use Attribute

If the specified resource is associated with a concatenated group, all members of the group have the in-use attribute removed, and the count of the number of resources held for reuse is increased by the number of members in the group. (An exception is when the group was generated by the system, that is, VSAM data sets spanning device types and GDG ALL groups. In these cases, the group is treated as a single resource.)

Requests to remove the in-use attribute from a private catalog are ignored.

If the resource is associated with an open data set or an open concatenated group, the resource is considered to be in-use until the data set is closed.

- a non-& dsname, non-permanently allocated data set which has a disposition of DELETE is unallocated (instead of having the in-use Attribute removed.)
- a conditional disposition specification which may have been assigned when the data set was allocated is removed, and therefore will not be honored in the event of a subsequent ABEND.

Identifying a Resource by Task-ID

In addition to specifying a ddname or dsname, the user may specify that the in-use attribute be removed based on task-id. The attribute may be removed from:

- all resources associated with a specified task, or all resources except those associated with the current task, its higher-level tasks, and the initiator. This function is used by the time sharing Terminal Monitor Program (TMP) to have the in-use attribute removed from any data sets allocated by a command processor when a command processor completes execution.

Dynamic Concatenation of Data Sets

Dynamic concatenation provides the user with a means of logically connecting allocated data sets into a concatenated group.

The user identifies the data sets to be concatenated by their associated ddnames. These data sets must not be open, or the request for dynamic concatenation fails.

The order of the data sets in the concatenated group will be the order in which the associated ddnames were specified. The name associated with the concatenated group will be the ddname that was specified first. The other ddnames are no longer associated with any data set. If a specified ddname is already associated with a concatenated group, this group will be included in the new concatenation.

After the request for dynamic concatenation is satisfied, all members of the dynamically concatenated group will be assigned the in-use attribute.

The Permanently Concatenated Attribute

The permanently concatenated attribute may be assigned when concatenation is requested. In addition, a concatenated group defined via JCL is automatically assigned the permanently concatenated attribute. Step and dynamic allocation requests which result in a concatenated group defined by the system are also automatically assigned this attribute. A GDG ALL request and a request for a VSAM data set which spans device types are examples of such requests.

A group having the permanently concatenated attribute has the following properties:

- If a dynamic unallocation request specifies the ddname associated with a permanently concatenated group or specifies the dsname of a system defined permanently concatenated group, the entire group is unallocated. If a dynamic unallocation request specifies a dsname associated with a non-system defined permanently concatenated group, that occurrence of the data set is not unallocated.
- The group cannot by dynamically deconcatenated into its member data sets.
- If a permanently concatenated group is dynamically concatenated with other data sets to form a new non-permanently concatenated group, the permanently concatenated group will remain intact if the new group is dynamically deconcatenated.
- If the group is not a system defined permanently concatenated group, the group is automatically assigned the permanently allocated attribute.

Dynamic Deconcatenation of Data Sets

Dynamic deconcatenation provides the user with a means of logically disconnecting the members of a concatenated group. The user identifies the concatenated group to be deconcatenated by specifying the ddnames of the group.

The request for dynamic deconcatenation fails if the concatenated group is open. A permanently concatenated group, or members of a concatenated group which are permanently concatenated, will remain concatenated.

When a concatenated group is dynamically deconcatenated, the ddnames that were associated with the data sets before they were concatenated are restored unless this would result in duplicate ddnames. This situation could arise if a dynamic allocation with the ddname to be restored occurred after a dynamic concatenation. In this case the deconcatenation request fails.

Dynamic deconcatenation has no effect on the in-use attributes associated with the members of the group.

Dynamic Information Retrieval

Dynamic information retrieval provides the user with information about his current allocation environment. The user can request information about ddnames or dsnames. In addition, a user may ask for information about any or all of his currently allocated requests by specifying a relative request number.

For example, information about all requests can be obtained by successively asking for information about the 1st, 2nd, ...Nth entry. A unique return code is provided when information is requested for a nonexisting relative entry.

The following information can be requested:

- data set name
- ddname
- member name
- the data set organization
- status
- normal disposition
- conditional disposition
- whether or not a resource has the permanently allocated attribute, in-use attribute, or permanently concatenated attribute
- whether or not a specified ddname is associated with an allocation of the user's terminal as an I/O device
- whether or not a specified ddname is associated with a DUMMY data set
- the number of resources held in anticipation of reuse which exceed the control value, that is, the number of existing allocations which must be unallocated before a request which requires the creation of a new allocation can be satisfied.
- whether or not the allocation is the last relative entry.

Requesting Dynamic Allocation Functions

To request a dynamic allocation function, code the DYNALLOC macro instruction (it has no operands) and supply the parameter structure shown in Figure 15.

This section describes the parameter structure fields, the dynamic allocation return codes, and the IBM-supplied mapping macros that aid in defining the parameter structure. The section also contains an example of a dynamic allocation request and describes an exit for an installation - written routine for checking allocation requests.

The actual values for verb codes, flags, error codes, information codes, and text unit fields are in the following chapter, Dynamic Allocation Parameter Structure Fields.

Parameter Structure Description

This topic describes the Request Block, Text Pointers, and Text Units fields in Figure 4. The names in parentheses, and those in Figure 4, are those assigned by the macro IEFZB4D0.

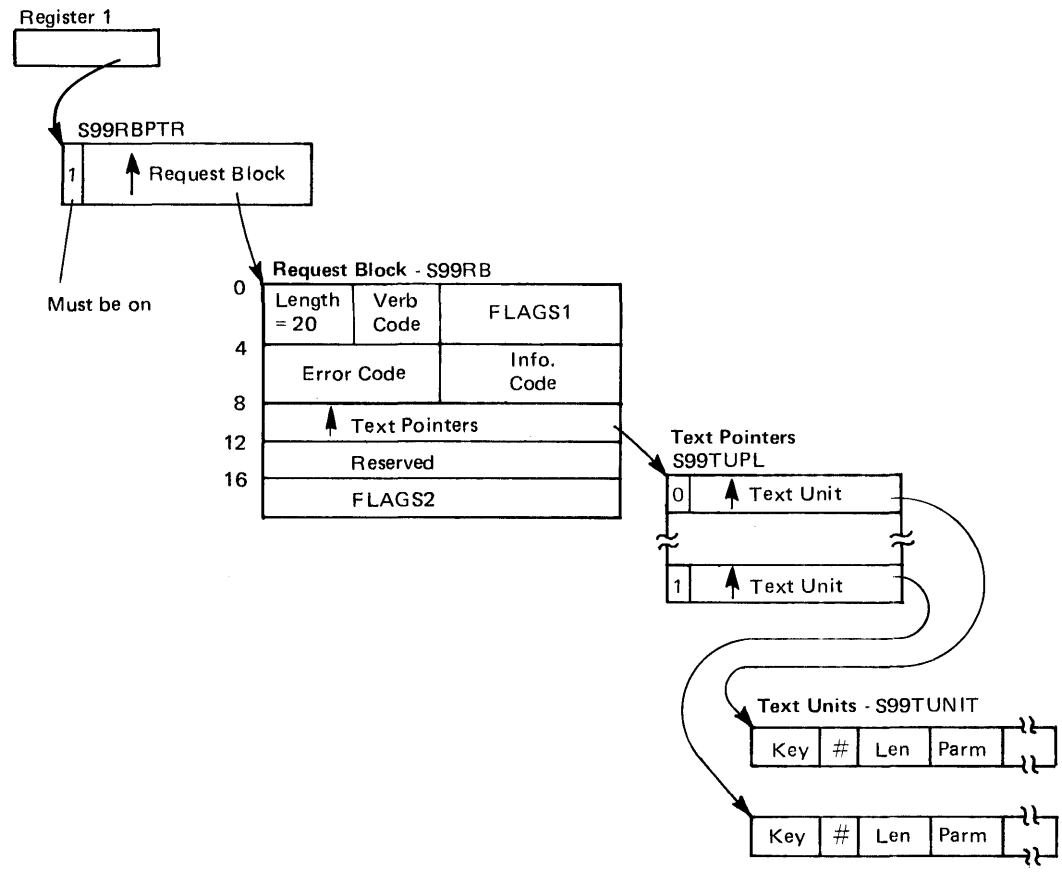

Figure 4. Data Area Structure for Dynamic Allocation Input

Request Block Fields

LENGTH

(S99RBLN) One byte containing the length of the request block.

VERB CODE (S99VERB) One byte which identifies the dynamic allocation to be performed. The functions that can be requested are:

- dsname allocation
- ddname allocation
- dsname or ddname unallocation
- concatenation
- deconcatenation
- removing the in-use attribute based on task id
- information retrieval

FLAGS1 (S99FLAG1) Two bytes of indicators used during dsname allocations to tell dynamic allocation not to satisfy the request with:

- an existing allocation
- an existing allocation that does not have the convertible attribute
- unmounted volumes or offline units.

ERROR CODE (S99ERROR) Two bytes that dynamic allocation uses to return an error reason code.

INFO. CODE (S99INFO) Two bytes that dynamic allocation uses to return an information reason code.

TEXT POINTERS address (S99TXTPP)	A full word containing the address of a list of pointers to the text units.
RESERVED	A fullword of zeros.
FLAGS2 (S99FLAG2)	Four bytes of indicators for use by authorized programs which direct dynamic allocation to: • wait for volumes • wait for the specified dsname to become available. • not reserve data sets • wait for a unit • consider offline devices • set special CATALOG data set indicators. • allocate a private catalog on behalf of the initiator. The indicators are also used to inform dynamic allocation that a TIOT enqueue has been performed and that it may mount volumes.

Text Pointers

TEXT UNIT address (S99TUPL)	A variable-length list of fullword pointers to text units. The end of the list is indicated by setting on the high-order bit of the last pointer. A fullword of zeroes is ignored. (S99TUPTR is the label for each pointer in the list. S99TUPLN is a label for the indicator of the end of the list.)

Text Units

TEXT UNITS (S99TUNIT)	A variable-length field containing the following subfields:
(S99TUKEY)	• KEY — two-bytes that contain a unique binary number that identifies a text unit. Dynamic allocation ignores KEY fields of zero.
(S99TUNUM)	• Number (#) field — two-byte binary number that specified the number of length and parameter combinations in the text unit.
(S99TUENT)	• the label for a length and parameter combination.
(S99TULNG)	• LEN field — two-byte binary number that specifies the length of the following parameter field.
(S99TUPAR)	• PARM field — contains the parameter information such as dsname, disposition, status, and so forth. The following general considerations and rules apply to the structure of the text units: • Special characters — of the type requiring apostrophes in JCL statements — are not permissible in PARM values. • Parameters whose values consist of alphameric and national characters may have trailing blanks. • The text units may be in order. • Each function of dynamic allocation has an associated set of text units, and each set is independent of any other. For example, the functions of both allocation and unallocation may use a KEY value of '0007', but that value does not necessarily have the same meaning for each function.

Dynamic Allocation Return Codes

When the dynamic allocation routines return control to the requesting program, register 15 contains a return code. Depending on the return code, the S99ERROR and S99INFO fields in the input request block (S99RB) may additionally contain error and informational reason codes respectively. The informational reason codes and their explanations and the error reason codes and their explanations are contained in the following chapter. The return codes in register are shown in figure 5.

Code		Meaning
0	—	Successful completion; there will also be an informational reason code if a non-terminating error occurred during request processing.
4	—	An error resulted from the current environment, the unavailability of a system resource, or a system routine failure; there will also be an error reason code.
8	—	The installation validation routine denied this request. (See the topic "Installation Input Validation Routine" for additional information.)
12	—	The error is due to an invalid parameter list; there will also be an error reason code from Class 3 as show in figure 7.

Figure 5. Dynamic Allocation Return Codes

Using the Dynamic Allocation Macros

IBM supplies two macros — IEFZB4D0 and IEFZB4D2 — to aid in constructing the dynamic allocation parameter structure. IEFZB4D0 provides symbolic names (dummy sections) for the positional information in the structure; IEFZB4D2 provides mnemonics for the text unit keyword values.

The following chapter contains a summary of the text unit keys for the dynamic allocation functions, and their mnemonics.

Example of a Dynamic Allocation Request

The assembler language example in figure 6 illustrates a dynamic allocation request for allocating SYS1.LINKLIB with a status of SHARE. It also requests that the dynamic allocation routines return the ddname associated with the data set that gets allocated.

```
LA 0,75                        AMOUNT OF STORAGE THAT THIS REQUEST NEEDS
GETMAIN R,LV=(0)               GET THE STORAGE NECESSARY FOR THE REQUEST
LR 8,1                         SAVE THE ADDRESS OF THE RETURNED STORAGE
USING S99RBP,8                 ESTABLISH ADDRESSABILITY FOR 'RBPTR' DSECT
LA 4,S99RBPTR+4                POINT FOUR BYTES BEYOND START OF 'RBPTR"
USING S99RB,4                  ESTABLISH ADDRESSABILITY FOR 'RB' DSECT
ST 4,S99RBPTR                  MAKE 'RBPTR' POINT TO 'RB'
OI S99RBPTR,S99RBPND           TURN ON THE HIGH ORDER BIT IN 'RBPTR'
XC S99RB(RBLEN),S99RB          ZERO OUT 'RB' ENTIRELY
MVI S99RBLN,RBLEN              PUT THE LENGTH OF 'RB' IN ITS LENGTH FIELD
MVI S99VERB,S99VRBAL           SET VERB CODE FIELD TO ALLOCATION FUNCTION
LA 5,S99RB+RBLEN               POINT TWENTY BYTES BEYOND START OF 'RB'
USING S99TUPL,5                ESTABLISH ADDRESSABILITY FOR TEXT UNIT PTRS
ST 5,S99TXTPP                  INITIALIZE THE TEXT POINTERS ADDRESS IN 'RB'
LA 6,S99TUPL+12                POINT JUST PAST THE THREE TEXT UNIT POINTERS
USING S99TUNIT,6               SET ADDRESSABILITY FOR THE FIRST TEXT UNIT
ST 6,S99TUPTR                  POINT 1ST TEXT UNIT POINTER TO 1ST TEXT UNIT
LA 7,DALDSNAM                  GET THE KEY FOR DSNAME
STH 7,S99TUKEY                 PUT THE KEY IN THE TEXT UNIT KEY FIELD
LA 7,1                         BECAUSE THE DSNAME KEY REQUIRES ONLY ONE
STH 7,S99TUNUM                  PARAMETER, LOAD AND STORE 1 IN NUMBER FIELD
LA 7,L'LINKDSN'                GET THE LENGTH OF THE DSNAME FIELD AND PUT
STH 7,S99TULNG                  IT INTO THE TEXT UNIT'S LENGTH FIELD
MVC S99TUPAR(12),LINKDSN       PUT THE DSNAME INTO TEXT UNIT PARM FIELD
LA 6,S99TUNIT+18               POINT JUST PAST THE FIRST TEXT UNIT
LA 5,S99TUPL+4                 POINT TO THE 2ND TEXT UNIT POINTER IN LIST
ST 6,S99TUPTR                  POINT 2ND TEXT UNIT POINTER TO 2ND TEXT UNIT
LA 7,DALSTATS                  GET THE KEY FOR STATUS SPECIFICA-
STH 7,S99TUKEY                  TION AND PUT THE KEY IN THE TEXT UNIT
LA 7,1                         BECAUSE THE STATUS KEY REQUIRES ONLY ONE
STH 7,S99TUNUM                  PARAMETER, LOAD AND STORE 1 IN THE NUMBER FIELD
STH 7,S99TULNG                 SET THE STATUS PARM LENGTH FIELD ALSO TO 1
MVI S99TUPAR,X'08'             SET THE PARM FIELD TO INDICATE SHARE DISP
LA 6,S99TUNIT+7                POINT JUST PAST THE SECOND TEXT UNIT
LA 5,S99TUPL+4                 POINT TO 3RD TEXT UNIT POINTER IN THE LIST
ST 6,S99TUPTR                  POINT 3RD TEXT UNIT POINTER TO 3RD TEXT UNIT
OI S99TUPTR,S99TUPLN           TURN ON HIGH ORDER BIT TO INDICATE LAST PTR
LA 7,DALRTDDN                  GET THE KEY FOR 'RETURN DDNAME' AND
STH 7,S99TUKEY                  PUT THE KEY IN THE TEXT UNIT KEY FIELD
LA 7,1                         BECAUSE 'RETURN DDNAME' KEY REQUIRES ONLY 1
STH 7,S99TUNUM                  PARAMETER, LOAD AND STORE 1 IN NUMBER FIELD
LA 7,8                         SET LENGTH OF FIELD FOR RETURNING DDNAME TO 8
STH 7,S99TULNG
LR 1,8                         PUT REQ BLK PTR ADDR IN REG 1 FOR DYNALLOC
DYNALLOC                       INVOKE DYNAMIC ALLOCATION TO PROCESS REQUEST
  .
  .
  .
LINKDSN  DC  C'SYS1.LINKLIB'
  .
  .
  .
 IEFZB4D0
 IEFZB4D2
   .
   .
   .
RBLEN    EQU (S99RBEND-S99RB)
```

Figure 6. Example of a Dynamic Allocation Request

Note the following concepts that the example illustrates:

- Storage is obtained via a GETMAIN macro instruction.
  In the example, the requirement is for 75 bytes, derive as follows —

Bytes	Purpose
4	Pointer to the request block.
20	Request block space requirement.
12	Four bytes each for three text unit pointers.
18	Text unit space for the data set name.
7	Text unit space for the data set status.
14	Text unit space for the requested return of the ddname.

- The parameter structure is mapped by the DSECTs that IEFZB4D0 provides.
- Use of the IEFZB4D2 mnemonics in the text unit keys.

Installation Input Validation Routine

An exit from the allocation control routine provides for a user-written routine to validate or alter any request to dynamic allocation. The routine is entered for all system and user dynamic allocation requests. The routine must be coded so as not to interfere with system requests.

The validation routine may test and modify the dynamic allocation input request, and it may indicate through a return code whether processing of the request is to continue. For example, the routine may perform the following functions:

- control the amount of direct access space requested
- check for authorization to use specified units
- check for authorization to use certain data sets
- check for authorization to hold certain resources for reuse.

Programming Considerations

The input validation routine must observe the following programming conventions and receives the following input:

- Its CSECT name must be IEFDB401 and it must reside in load module IEFW21SD.
- It receives control in supervisor state under the scheduler's protection key (key 1). At entry, register 1 pointers to a list of addresses for the following parameters:
  — a copy of the dynamic allocation input request block, text unit pointers, and text units in scheduler key fetch-protected storage.
  — the address of a work area for the use of the routine. This area is contiguous with the text unit pointer list so that it can be used to extend the list and provide additional text units.
  — A fullword that contains the length of the work area (500 bytes).
  — the eight-character job name.
  — the twenty-byte programmer name.
  — an area that contains accounting information from the JOB statement. The first byte of this area contains the number of accounting fields; the accounting fields follow this byte. Each entry for an accounting field contains the length of the field (one byte, binary), followed by the field itself. The entry for a null field contains a length of zero.
  — The eight-character step name.
  — The eight character program name.
  — An area containing accounting information from the EXEC statement. The first byte of this area contains the number of accounting fields (0 for no fields); the accounting fields follow this byte. Each entry for an accounting field contains the length of the field (one byte, binary), followed by the field itself. The entry for a null field contains a length of zero.

- The routine must use register 15 to return to dynamic allocation a code of zero if processing of the request is to continue, or any other code if processing is to terminate.

The IBM-supplied routine that your routine may replace allows all requests to continue processing.

This chapter contains the values that can be specified or returned in the fields of the dynamic allocation parameter structure. (See figure 4 in the section "Allocation Services" for an illustration of the structure.) The fields described here are the Informational Reason Codes, the Error Reason Codes, Flags1, Flags2, Verb Codes, and the Text Unit fields - key, number, length, and parameter. The labels used in this chapter are those assigned by the macros IEFZB4D0 and IEFZB4D2.

Informational Reason Codes

The codes below are returned in the two-byte field S99INFO.

Code	Meaning
0004	reserved
0008	overriding disposition ignored; returned after an unallocation request
000C-001C	reserved
002w	data set was successfully unallocated but completion of the requested CATLG or UNCATLG disposition was unsuccessful. The digit "w" was a code representing the reason for the failure. The meaning of each code is: 1 a control volume was required and a utility program must be used to catalog the data set. 2 the data set to be cataloged had previously been cataloged, or the data set to be uncataloged could not be located, or no change was made to the volume serial list of a data set with a disposition of CATLG. 3 a specified index did not exist. 4 the data set could not be cataloged because space was not available in the catalog data set. 5 not enough storage was available to perform the specified cataloging. 6 the data set to be cataloged in a generation index is improperly name. 7 the data set to be cataloged was not open and no density information was provided (for dual density tape requests only). 9 an uncorrectable I/O error occurred in reading or writing the catalog.
003x	data set was successfully unallocated but completion of requested DELETE disposition was unsuccessful. The one-digit code "x" represents the reason for the failure. Its meanings are: 1 the expiration date had not occurred. 4 no device was available for mounting during deletion. 5 not enough storage was available to perform the specified deletion. 6 either no volumes were mounted or the mounted volumes could not be demounted to permit the remaining volumes to be mounted. 8 the SCRATCH routine returned an error code. If the user's JOB statement requested allocation/termination messages, message IEF283I will appear in the SYSOUT listing. This message will list the volume serial numbers; following each number will be a code that explains why the number was not deleted.

Error Reason Codes

Error reason codes are divided into these classes:

Class	Description
1	reserved
2	unavailable system resource
3	invalid parameter list
4	environment error
5	reserved
6	reserved
7	system routine error

The error reason code contains the codes shown in figure 7. The second hexadecimal digit will be one of the class designations above. The field is labeled S99ERROR.

Note: The explanations of the codes in figure 7 are followed by an indication of the kind of request associated with the code.

CLASS 2 CODES

Code	Meaning
0204	Real storage unavailable; dsname allocation.
0208	Reserved.
020C	Request for exclusive use of a shared data set cannot be honored; dsname allocation.
0210	Requested data set unavailable. The data set is allocated to another job and its usage attribute conflicts with this request; dsname allocation.
0214	Unit(s) not available; dsname allocation.
0218	Specified volume or an acceptable volume is not mounted, and user does not have volume mounting authorization; dsname allocation.
021C	Unit name specified is undefined; dsname allocation.
0220	Requested volume not available; dsname allocation.
0224	Eligible device types do not contain enough units; dsname allocation.
0228	Specified volume or unit in use by system; dsname allocation.
022C	Volume mounted on ineligible permanently resident or reserved unit; dsname allocation.
0230	Permanently resident or reserved volume on required unit; dsname allocation.
0234	More than one device required for a request specifying a specific unit; dsname allocation.
0238	Space unavailable in Task Input Output Table (TIOT); dsname allocation, concatenation.
023C	Required catalog not mounted, and user does not have volume mounting authorization; dsname allocation.
0240	Requested device is a console; dsname allocation.
0244	Telecommunication device not accessible; dsname allocation.
0248	MSS volume unable to be mounted; dsname allocation.

CLASS 3 CODES

Code	Meaning
0304-0338	Assigned by DAIR. (See *OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor* GC28-0648.)
033C-0354	Reserved.
0358	Overriding disposition of DELETE invalid for data set allocated as SHR; unallocation.[1]
035C	Invalid PARM specified in text unit; all functions.[2]
0360	Invalid KEY specified in text unit; all functions.[2]
0364	JOBLIB/STEPLIB/JOBCAT/STEPCAT specified as ddname, or associated with specified dsname; dsname allocation, ddname allocation, unallocation, concatenation, deconcatenation.[1]
0368	Authorized function requested by unauthorized user; all functions.
036C	Invalid parameter list format; all functions.
0370	Reserved.
0374	Invalid # specified in text unit; all functions.[2]
0378	Duplicate KEY specified in text unit; all functions.[2]
037C	Invalid LEN specified in text unit; all functions.[2]
0380	Mututally exclusive KEY specified in text unit; dsname allocation, unallocation, information retrieval, remove In-use.[2]
0384	Mutually inclusive KEY not specified; unallocation, dsname allocation.[2]
0388	Required key not specified; ddname allocation, information retrieval, concatenation, deconcatenation, remove In-Use, unallocation.
038C	Duplicate ddnames specified for concatenation.
0390	GDG group name specified with relative generation number exceeds 35 characters; dsname allocation.
0394	Status and relative generation number are incompatible; dsname allocation.
0398	Volume sequence number exceeds the number of volumes; dsname allocation.
039C	Device type and volume are incompatible; dsname allocation.

1 The informational reason code field contains 0004 if the specified data set has been unallocated, although an error was encountered processing another occurrence of the data set as the error reason code indicates.

2 The informational reason code contains the value of the key that caused the error.

Figure 7. Error Reason Codes (Part 1 of 2)

CLASS 4 CODES

Code	Meaning
0404-040C	Reserved.
0410	Specified ddname unavailable; dsname allocation, ddname allocation.
0414-041C	Reserved.
0420	Specified ddname associated with an OPEN data set; ddname allocation, concatenation, deconcatenation, unallocation, dsname allocation.[1]
0424	Deconcatenation would result in duplicate ddnames.[1]
0428-0430	Reserved.
0434	Ddname specified in ddname allocation request is associated with a convertible or non-permanently allocated resource.
0438	Specified ddname not found; information retrieval, ddname allocation, concatenation, deconcatenation, unallocation.
043C	Resources could not be unallocated to decrease the number of resources held in anticipation or reuse to meet the limit of the control value; dsname allocation.
0440	Specified dsname not found; information retrieval, unallocation.
0444	Relative entry number specified in information retrieval request not found.
0448	Data set requested NEW found allocated; dsname allocation.
044C	Existing data set requested found allocated as eligible for deletion; dsname allocation.
0450	Request would cause the limit of 1635 concurrent allocations to be exceeded; dsname allocation.
0454	Ddname in DCB reference not found; dsname allocation.
0458	Dsname in DCB reference or volume reference is a GDG group name; dsname allocation.
045C	Specified dsname to be unallocated is a member of Permanently Concatenated group.[1]
0460	Specified dsname or member to be unallocated is not associated with specified ddname.
0464	Specified dsname to be unallocated is a private catalog.[1]
0468	Error while allocating or opening a private catalog.
046C	Remote work station not defined to Job Entry Subsystem; dsname allocation, unallocation.
0470	User unauthorized for Job Entry Subsystem request; dsname allocation.
0474	Error while attempting to select optimum device; dsname allocation.
0478	Unable to process Job Entry Subsystem request; dsname allocation, unallocation.
047C	Unable to establish ESTAE environment; all functions.
0480	The number of units to satisfy the request exceeds the limit; dsname allocation.
0484	Request denied by operator; dsname allocation.
0488	GDG pattern DSCB not mounted; dsname allocation.
048C	GDG pattern DSCB not found; dsname allocation.
0490	Error changing allocation assignments; dsname allocation.
0494	Error processing OS CVOL.
0498	MSS volume not accessible; dsname allocation.
049C	MSS volume not defined; dsname allocation.
04A0	Specified MSVGP name not defined; dsname allocation.

CLASS 7 CODES

Code	Meaning
17zz	LOCATE error; dsname allocation. (Note: Hexadecimal '08', '18', and '2C' are the only expected LOCATE return codes. 'FF' is returned as the value of zz if an unexpected return code is returned by LOCATE.)
27zz	Reserved.
37zz	Reserved.
47zz	DADSM error; dsname allocation.
57zz	CATALOG error; dsname allocation.
67zz	OBTAIN error; dsname allocation, information retrieval.

Note: The failing system routine returns the code represented by "zz".

1 The informational reason code field contains 0004 if the specified data set has been unallocated, although an error was encountered processing another occurrence of the data set as the error reason code indicates.

Figure 7. Error Reason Codes (Part 2 of 2)

FLAGS1

The FLAGS1 field is a two-byte labeled S99FLAG that is used during dsname allocation requests. The meaning of the bits in the field is as follows:

Bit	Meaning When On
0 (S99ONCVN)	Do not use an existing allocation that does not have the convertible attribute to satisfy the request.
1 (S99NOCVN)	Do not use an existing allocation to satisfy the request.
2 (S99NOMNT)	Do not mount volumes or consider offline units. (This bit overrides S99MOUNT and S99OFFLN in FLAGS2.)
3 (S99JBSYS)	Treat the data set as part of the job's normal output. The data set is not expected to be dynamically unallocated (spun off). (This flag is used for SYSOUT data sets.) If the data set is dynamically unallocated the data set will be printed immediately but paging space will not be released until the job ends.
4-15	Reserved. Must be zero.

FLAGS2

The FLAGS2 field is a four-byte field of indicators labeled S99FLAG2. These indicators may be set only by authorized programs. To be authorized the requesting program must meet at lease one of these criteria:

- it must have a system storage protection key (0-7)
- it must be in supervisor state
- it must be APF authorized
- it must be the installation-writer input validation routine.

If a FLAGS2 indicator is set by an unauthorized program, the dynamic allocation routines fail the request.

The FLAGS2 indicators are used only for dsname allocation requests except for bit 5; bit 5 may be used for all requests. The meaning of the bits is:

Bit	Meaning When On
0 (S99WTVOL)	Wait for volumes
1 (S99WTDSN)	Wait for dsname
2 (S99NORES)	Do not reserve data sets
3 (S99WTUNT)	Wait for units
4 (S99OFFLN)	Consider offline devices
5 (S99TIONQ)	TIOT ENQ already performed
6 (S99CATLG)	Set special catalog data set indicators
7 (S99MOUNT)	Volumes may be mounted
8 (S99UDEVT)	Unitname parameter is a device type
9 (S99PCINT)	Allocate a private catalog on behalf of the initiator
10-31	Reserved. Must be zero.

When the user requests the wait functions (bits 0, 1, and 3), dynamic allocation assumes that no conflicts can occur because of resources the requestor already owns. For example, if the resource requested by user A is held by user B, an interlock will occur if user B in turn unconditionally requests a resource held by A. With the dsname and volume wait functions dynamic allocation attempts to gain control of the resource with an unconditional ENQ. If the resource requested is held by the requestor, the ENQ will result in an ABEND.

Dynamic allocation informs the operator whenever it must wait for a data set, volume, or unit. The operator may then optionally indicate that dynamic allocation should not wait. If he does, the request fails.

Verb Codes

The verb code is a one-byte field labeled S99VERB that identifies the dynamic allocation function to be performed. The following codes may be specified:

Verb Code	Meaning
01 (S99VRBAL)	request for dsname allocation
02 (S99VRBUN)	request for unallocation (based on dsname or ddname)
03 (S99VRBDC)	request for concatenation
04 (S99VRBDC)	request for deconcatenation
05 (S99VRBRI)	request for removing the in-use attribute based on task-id
06 (S99VRBDN)	request for ddname allocation
07 (S99VRBIN)	request for information retrieval

Text Unit Fields

The TEXT UNIT field is a variable-length field labeled S99TUNIT. It contains an identifying key, one or more parameter length/parameter combinations, and the number of length/parameter combinations. The fields and their labels are:

KEY field (S99TUKEY)	two bytes that contain a unique binary identification number for a specific key. There is a set of keys for each dynamic allocation function. The key values, their labels, and their meanings are summarized in figures 8-16. The topics that follow these figures explain the keys in detail.
Number field (S99TUNUM)	two-byte binary number that specifies the number of length/parameter combinations in the text unit.
LEN field (S99TULNG)	two-byte binary number that specifies the length of the parameter field that follows it.
PARM field (S99TUPAR)	contains parameter information. The parameters that can be specified with each key are explained together with the explanation of the associated key in the topics that follow.

S99TUENT is the label for a length/parameter combination. Additionally, IEFZB4D0 provides the following DSECT for use when specifying multiple parameters in a single text unit. This DSECT places the length field at zero displacement for the second and subsequent combinations:

S99TUFLD	label for the DSECT
S99TULEN	label for the length field
S99TUPRM	label for the parameter

Hex Text Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
0001	DALDDNAM	Associates a ddname with an allocation request.
0002	DALDSNAM	Names the data set to be allocated.
0003	DALMEMBR	Allocates only a particular data set member.
0004	DALSTATS	Specifies the data set status.
0005	DALNDISP	Specifies the data set's normal disposition.
0006	DALCDISP	Specifies the data set's conditional disposition.
0007	DALTRK	Specifies the space allocation in tracks.
0008	DALCYL	Specifies the space allocation in cylinders.
0009	DALBLKLN	Specifies the average data block length.
000A	DALPRIME	Specifies a primary space quantity.
000B	DALSECND	Specifies a secondary space quantity.
000C	DALDIR	Specifies the number of PDS directory blocks.
000D	DALRLSE	Deletes unused space at data set closure.
000E	DALSPFRM	Ensures a specific allocated space format.
000F	DALROUND	Specifies space allocation in whole cylinders.
0010	DALVLSER	Specifies volume serial numbers.
0011	DALPRIVT	Specifies the private volume use attribute.
0012	DALVLSEQ	Specifies the volume sequence number processing.
0013	DALVLCNT	Specifies the data set's volume count.
0014	DALVLRDS	Specifies volume reference to a cataloged data set.
0015	DALUNIT	Describes the unit specification.
0016	DALUNCNT	Specifies the number of devices to be allocated.
0017	DALPARAL	Specifies parallel mounting for a data set's volumes.
0018	DALSYSOU	Specifies the SYSOUT data set and defines its class.
0019	DALSPGNM	Specifies the SYSOUT program name.
001A	DALSFMNO	Specifies the SYSOUT form number.
001B	DALOUTLM	Limits the SYSOUT data set's logical record count.
001C	DALCLOSE	Frees a data set at closure.
001D	DALCOPYS	Specifies the SYSOUT listing copies count.
001E	DALLABEL	Specifies the type of volume label.
001F	DALDSSEQ	Specifies a tape data set's relative position.
0020	DALPASPR	Password protects the created data set.
0021	DALINOUT	Specifies "input only" or "output only" data set processing.
0022	DALEXPDT	Specifies the data set's expiration date.
0023	DALPRETPD	Specifies the data set's retention period.
0024	DALDUMMY	Allocates a dummy data set.
0025	DALFCBIM	Identifies the forms control buffer image.
0026	DALFCBAV	Requests operator verification of the image display or forms alignment.
0027	DALQNAME	Names a TPROCESS macro.
0028	DALTERM	Specifies a time sharing terminal as an I/O device.
0029	DALUCS	Specifies a universal character set.
002A	DALUFOLD	Specifies "fold mode" for loading the requested print chain or train.
002B	DALUVRFY	Requests operator verification of the correct print chain or train mounting.
002C	DALDCBDS	Specifies the retrieval of DCB information from a cataloged data set's label.
002D	DALDCBDD	Specifies the retrieval of DCB information from a ddname-related, currently allocated data set.
0058	DALSUSER	Specifies remote workstation routing for the SYSOUT data set.
0059	DALSHOLD	Specifies hold queue routing for the SYSOUT data set.
005E	DALMSVGP	Specifies a group of virtual volumes.

Figure 8. Data Set Name Allocation (Verb Code 01) — Text Unit Keys, Mnemonics, and Functions

Data Set Name Allocation Text Units

Most of the information that can be specified on a JCL DD card may be specified in text units for the allocation function of dynamic allocation (VERB code X'01'). These text units are listed below. The meaning of the parameters is the same as when specified on a DD statement as described in *OS/VS2 JCL*, GC28-0692.

Ddname specification - Key = X'0001'

This key is used to specify a ddname to be associated with an allocation request. When this key is specified, # must be one, LEN is the length of the ddname field, and PARM contains the ddname.

Example: to specify the ddname DD1, code

```
KEY     #       LEN     PARM
0001    0001    0003    C4 C4 F1
```

Dsname specification - Key = X'0002'

This key is used to specify the name of the data set to be allocated. The user cannot refer to a previously defined dsname. The QNAME and IPLTXTID keys are mutually exclusive with this key. When this key is specified, # must be one, LEN is the length of the dsname, and PARM contains the dsname.

Example: to specify the dsname MYDATA, code

```
KEY     #       LEN     PARM
0002    0001    0006    D4 E8 C4 C1 E3 C1
```

Example: to specify the temporary dsname &LOAD, code

```
KEY     #       LEN     PARM
0002    0001    0005    50 D3 D6 C1 C4
```

Example: to specify the dsname A.B, code

```
KEY     #       LEN     PARM
0002    0001    0003    C1 4B C2
```

Member name specification - Key = X'0003'

This key is used to specify that a particular member of a data set is to be allocated, rather than the entire data set. A relative generation group number may be specified as the member name. The QNAME and IPLTXTID keys are mutually exclusive with this key. When this key is specified, # must be one, LEN is the actual length of the member name, and PARM contains the member name.

Example: to specify the membername MEM1, code

```
KEY     #       LEN     PARM
0003    0001    0004    D4 C5 D4 F1
```

Example: to specify the relative generation number *1, code

```
KEY     #       LEN     PARM
0003    0001    0002    4E F1
```

Data Set Status specification - Key = X'0004'

This key specifies the data set status. It is mutually exclusive with a Sysout Specification. When this key is specified, # and LEN must be one, and PARM contains the value:

X'01' if OLD is desired
X'02' if MOD is desired
X'04' if NEW is desired
X'08' if SHR is desired

Example: to specify a status of NEW, code

```
Key     #       LEN     PARM
0004    0001    0001    04
```

Data Set Normal Disposition specification - Key = X'0005'

This key specifies the data set normal disposition. This key is mutually exclusive with a Sysout specification. When this key is specified, # and LEN must be one, and PARM contains the value:

X'01' if UNCATLG is desired
X'02' if CATLG is desired
X'04' if DELETE is desired
X'08' if KEEP is desired

Example: to specify a normal disposition of DELETE, code

```
KEY     #       LEN     PARM
0005    0001    0001    04
```

Data Set Conditional Disposition specification - Key = X'0006'

This key specifies the conditional data set disposition. The values for #, LEN, and PARM are the same as for normal disposition. This key is mutually exlusive with a Sysout specification.

Example: to specify a conditional disposition of DELETE code

```
KEY     #       LEN     PARM
0006    0001    0001    04
```

Track Space Type (TRK) specification - Key = X'0007'

This key is used to specify that space is to be allocated in tracks. The primary quantity space key or the secondary quantity space key (see below) must also be specified when this key is specified. The Cylinder and Block Space Type specifications are mutually exclusive with this key. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify Track space type, code

```
KEY     #       LEN     PARM
0007    0000    -       -
```

Cylinder Space Type (CYL) specification - Key = X'0008'

This key is used to specify that space is to be allocated in cylinders The primary quantity space key or secondary quantity space key must also be specified when this key is specified. The Track and Block Space Type specifications are mutually exclusive with this key. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify Cylinder space type, code

```
KEY     #       LEN     PARM
0008    0000    -       -
```

Block Space Type specification - Key = X'0009'

This key is used to specify the average data block length to be used by the system in computing the amount of space to allocate. The primary quantity space key or the secondary quantity space key must also be specified when this key is specified. The Track and Cylinder

Space type keys are mutually exclusive with this key. When this key is specified, # must be one, LEN must be three, and PARM contains the average data block length.

Example: to specify an average data block length of 80, code

```
KEY      #        LEN      PARM
0009     0001     0003     00 00 50
```

Primary Space Quantity specification - Key = '000A'

This key is used to specify a primary space quantity. A space type key must also be specified when this key is specified. When this key is specified, # must be one, LEN must be three, and PARM contains the primary quantity value.

Example: to specify a primary quantity of 20, code

```
KEY      #        LEN      PARM
000A     0001     0003     00 00 14
```

Secondary Space Quantity specification - Key = X'000B'

This key is used to specify a secondary space quantity. When this key is specified, # must be one, LEN must be three, and PARM contains the primary quantity value.

Example: to specify a secondary space quantity of 10, code

```
KEY      #        LEN      PARM
000B     0001     0003     00 00 0A
```

Directory Block specification - Key = X'000C'

This key is used to specify the number of blocks to be contained in the directory of a partitioned data set. A space type key and the primary space quantity key must also be specified when this key is specified. When this key is specified, # must be one, LEN must be three, and PARM contains the number of directory blocks.

Example: to specify two directory blocks, code

```
KEY      #        LEN      PARM
000C     0001     0003     00 00 02
```

Unused Space Release (RLSE) specification - Key = X'000D'

This key is used to specify that unused space is to be deleted when the data set is closed. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify the release of unused space, code

```
KEY      #        LEN      PARM
000D     0000     -        -
```

Format of Allocated Space specification - Key = X'000E'

This key is used to ensure a particular format of allocated space. When this key is specified, # and LEN must be one, and PARM contains:

X'02' if different areas of contiguous space are to be allocated (ALX)
X'04' if maximum contiguous space is required (MXIG)
X'08' if space must be contiguous (CONTIG)

Example: to specify contiguous space format, code

```
KEY     #       LEN     PARM
000E    0001    0001    08
```

Whole Cylinder Allocation (ROUND) specification - Key = X'000F'

This key is used to request that allocated space be equal to one or more whole cylinders when space is requested in units of blocks. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify allocation of whole cylinders, code

```
KEY     #       LEN     PARM
000F    0000    -       -
```

Volume Serial specification - Key = X'0010'

This key is used to specify volume serial numbers. It is mutually exclusive with a Sysout specification and volume reference specification (see below). When this key is specified, # contains the number of volume serials being specified, LEN contains the length of the immediately following volume serial, and PARM contains the volume serial.

Example: to specify the volume serials 231400 and 231401, code

```
KEY     #       LEN     PARM            LEN     PARM
0010    0002    0006    F2F3F1F4F0F0    0006    F2F3F1F4F0F1
```

Private Volume specification - Key = X'0011'

This key is used to specify that the volume(s) allocated be assigned the PRIVATE volume use attribute. This key is mutually exclusive with a Sysout specification. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify the PRIVATE volume attribute, code

```
KEY     #       LEN     PARM
0011    0000    -       -
```

Volume Sequence Number specification - Key = X'0012'

This key is used to specify which volume of a multi-volume data set processing is to begin with. This key is mutually exclusive with a Sysout specification. When this key is specified, # must be one, LEN must be two, and PARM contains the volume sequence number.

Example: to specify a volume sequence number of two, code

```
KEY     #       LEN     PARM
0012    0001    0002    0002
```

Volume Count specification - Key = X'0013'

This key is used to specify the maximum number of volumes an output data set may require. This key is mutually exclusive with a Sysout specification. When this key is specified, # and LEN must be one, and PARM contains the volume count.

Example: to specify a volume count of 10, code

```
KEY     #       LEN     PARM
0013    0001    0001    0A
```

Volume Reference to a dsname specification - Key = X'0014'

This key is used to specify that the system is to obtain volume serial information from the specified cataloged data set. This key is mutually exclusive with a Sysout specification and volume serial specification. (Volume reference to a ddname can not be done through dynamic allocation.) When this key is specified, # must be one, LEN is the actual length of the dsname, and PARM contains the dsname.

Example: to specify volume reference to the data set, DSN1, code

```
KEY      #        LEN      PARM
0014     0001     0004     C4 E2 D5 F1
```

Unit Description specification - Key = X'0015'

This key is used to specify a unit group (esoteric) name, device type, or specific unit address (in EBCDIC). When this key is specified, # must be one, LEN is the actual length of the unit description, and PARM contains the unit description.

Example: to specify the unit group name SYSDA, code

```
KEY      #        LEN      PARM
0015     0001     0005     E2 E8 E2 C4 C1
```

Example: to specify the device type 3330, code

```
KEY      #        LEN      PARM
0015     0001     0004     F3 F3 F3 F0
```

Example: to specify the unit address 230, code

```
KEY      #        LEN      PARM
0015     0001     0003     F2 F3 F0
```

Unit Count specification - Key = X'0016'

This key is used to specify the number of devices to be allocated. It is mutually exclusive with a parallel mount specification (see below). When this key is specified, # and LEN must be one, and PARM contains the unit count.

Example: to specify a unit count of ten, code

```
KEY      #        LEN      PARM
0016     0001     0001     0A
```

Parallel Mount specification - Key = X'0017'

This key is used to specify that each volume of a data set is to be assigned a device. It is mutually exclusive with the unit count key. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify parallel mount, code

```
KEY      #        LEN      PARM
0017     0000     -        -
```

Sysout specification - Key = X'0018'

This key is used to indicate that a system output data set is to be allocated and to define the output class of the data set. When this key is specified and a class other than the default of the message class is desired, # and LEN must be one, and PARM contains the output class. To obtain the default of the message class, # must be zero, LEN and PARM are not specified. Volume, QNAME, Status, and Disposition specifications are mutually exclusive with the Sysout specification.

Example: to specify a Sysout data set in class A, code

```
KEY     #       LEN     PARM
0018    0001    0001    C1
```

Example: to specify a Sysout data set and to default the class, code

```
KEY     #       LEN     PARM
0018    0000    -       -
```

Sysout Program Name specification - Key = '0019'

This key specifies the sysout program name. The Sysout key must also be specified when this key is specified. When this key is specified, # must be one, LEN is the actual length of the name, and PARM contains the program name.

Example: to specify the program name MYWRITER, code

```
KEY     #       LEN     PARM
0019    0001    0008    D4 E8 E6 D9 C9 E3 C5 D9
```

Sysout Form Number specification - Key = '001A'

This key specifies the sysout form number. The Sysout key must also be specified when this key is specified. When this key is specified, # must be one, LEN is the actual length of the form number, and PARM contains the form number.

Example: to specify the form number 1234, code

```
KEY     #       LEN     PARM
001A    0001    0004    F1 F2 F3 F4
```

Sysout Output Limit specification - Key = X'001B'

This key is used to limit the number of logical records in a sysout data set. The Sysout specification must also be specified when this key is specified. When this key is specified, # must be one, LEN must be three, and PARM contains the output limit.

Example: to specify an Output Limit of 1000, code

```
KEY     #       LEN     PARM
001B    0001    0003    0003E8
```

Unallocation At CLOSE specification - Key X'001C'

This key is used to specify that unallocation is to occur when a DCB is CLOSEd rather than at Step Unallocation. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify Unallocation at CLOSE, code

```
KEY     #       LEN     PARM
001C    0001    -       -
```

Sysout Copies specification - Key = X'001D'

This key is used to request up to 255 hardcopy listings of a particular SYSOUT data set to be printed. The SYSOUT key must also be specified when this key is specified. When this key is specified, # and LEN must be one, and PARM contains the number of copies being requested.

Example: to specify a request for 25 copies, code

```
KEY     #       LEN     PARM
001D    0001    0001    19
```

Label Type specification - Key = X'001E'

This key is used to specify the type of label associated with a volume. This key is mutually exclusive with a Sysout specification. When this key is specified, # and LEN must be one, and PARM contains:

X'01' if the volume has no label (NL)
X'02' if the volume has an IBM standard label (SL)
X'04' if the volume has a non standard label (NSL)
X'08' if the volume has both an IBM standard label and a user label (SUL)
X'10' if label processing is to be bypassed (BLP)
X'21' if the system is to check for and bypass a leading tape mark on DOS unlabeled tape (LTM).
X'40' if the volume has an American National Standard label (AL)
X'48' if the volume has an American National Standard label and an American National Standard user label (AUL)

Example: to specify no labels, code

```
KEY     #       LEN     PARM
001E    0001    0001    01
```

Data Set Sequence Number specification - Key = X'001F'

This key is used to specify the relative position of a data set on a tape volume (data set sequence number). This key is mutually exclusive with a Sysout specification. When this key is specified, # must be one, LEN must be two, and PARM contains the sequence number.

Example: to specify a data set sequence number of 2, code

```
KEY     #       LEN     PARM
001F    0001    0002    0002
```

Password Protection specification - Key = X'0020'

This key is used to specify that the data set being created is to be password protected. This key is mutually exclusive with a Sysout specification. When this key is specified, # and LEN must be one, and PARM contains:

X'10' if the data set should not be read, changed, extended, or deleted without the password.
X'30' if the data set should not be changed, extended, or deleted without the password. Reading is permitted.

Example: to specify complete password protection, code

```
KEY     #       LEN     PARM
0020    0001    0001    10
```

Input Only or Output Only specifications - Key = X'0021'

This key is used to specify that the data set is to be processed for input only or output only. This key is mutually exclusive with a Sysout specification. When this key is specified, # and LEN must be one, and PARM contains:

X'40' if output only is to be specified
X'80' if input only is to be specified

Example: to specify processing for input only, code

```
KEY     #       LEN     PARM
0021    0001    0001    80
```

Expiration Date specification - Key = X'0022'

This key is used to specify the date when the data set can be deleted or overwritten by another data set. This key is mutually exclusive with the Retention Period and Sysout specifications. When this key is specified, # must be one, LEN must be five, and PARM contains five digits, a two digit year number and a three digit day number (YYDDD).

Example: to specify an expiration date of January 1, 1975 (75001), code

```
KEY     #       LEN     PARM
0022    0001    0005    F7 F5 F0 F0 F1
```

Retention Period specification - Key = X'0023'

This key is used to specify the number of days that must pass before the data set can be deleted or overwritten by another data set. This key is mutually exclusive with the Expiration Date and Sysout specifications. When this key is specified, # must be one, LEN must be two, and PARM contains the retention period. (Note: maximum retention period is 9999 days.)

Example: to specify a retention period of 10 days, code

```
KEY     #       LEN     PARM
0023    0001    0002    000A
```

Dummy Data Set specification - Key = X'0024'

This key is used to specify that a Dummy data set is to be allocated. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify a DUMMY data set is to be allocated, code

```
KEY     #       LEN     PARM
0024    0000    -       -
```

Forms Control Buffer (FCB) Image Identification specification - Key = X'0025'

This key is used to specify the code that identifies the image to be loaded into the FCB. This key is mutually exclusive with the DCB INTVL and FRID specifications (see below). When this key is specified, # must be one, LEN contains the length of the image-id (maximum of 4), and PARM contains the image-id.

Example: to specify the image-id STD1, code

```
KEY     #       LEN     PARM
0025    0001    0004    E2 E3 C4 F1
```

Form Alignment and Image Verification specification - Key = X'0026'

This key is used to request that the operator check the alignment of the printer forms before the data set is printed or that he visually verify the image displayed on the printer as the desired one. The FCB image-id specification must also be coded when this key is specified. When this key is specified, # and LEN must be one, and PARM contains:

X'04' if verification is to be requested (VERIFY).
X'08' if alignment is to be requested (ALIGN)

Example: to specify verification, code

```
KEY     #       LEN     PARM
0026    0001    0001    04
```

QNAME specification - Key = X'0027'

This key is used to specify the name of a TPROCESS macro. The Dsname, member name, IPLTXTID, and Sysout specifications are mutually exclusive with this key. The DCB BLKSIZE, BUFL, LRECL, OPTCD and RECFM specifications (see below) are meaningful with this key. When this key is specified, # must be one, LEN is the length of the process name, and PARM contains the process name.

Example: to specify the process name, TP1, code

```
KEY     #       LEN     PARM
0027    0001    0003    E3 D7 F1
```

Terminal specification - Key = X'0028'

This key is used to specify that a time sharing terminal is to be used as an I/O device. In a batch environment, the specification is not used, but is checked for syntax. In a time sharing environment, all other specifications except DCB specifications are ignored. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify a terminal allocation, code

```
KEY     #       LEN     PARM
0028    0000    -       -
```

Universal Character Set (UCS) specification - Key = X'0029'

This key is used to identify a special character set to be used for printing a data set. The DCB INTVL and RESERVE specifications (see below) are mutually exclusive with this key. When this key is specified, # must be one, LEN is the length of the character set code name (maximum is four) and PARM contains the character set code.

Example: to specify the character set code AN, code

```
KEY     #       LEN     PARM
0029    0001    0002    C1 D5
```

Fold Mode specification - Key = X'002A'

This key is used to specify that the chain or train corresponding to the desired character set be loaded in the fold mode. The Universal Character Set specification must also be specified when this key is coded. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify Fold Mode, code

```
KEY     #       LEN     PARM
002A    0000    -       -
```

Character Set Image Verification specification - Key = X'002B'

This key is used to specify that the operator is to verify that the correct chain or train is mounted before the data set is printed. The Universal Character Set specification must also be specified when this key is coded. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify Character Set Image Verification, code

```
KEY     #       LEN     PARM
002B    0000    -       -
```

DCB Reference to a Dsname specification Key = X'002C'

This key is used to specify that DCB information is to be retrieved from the data set label of a cataloged data set. This data set must reside on a direct access volume and the volume must currently be mounted. The DSORG, RECFM, OPTCD, BLKSIZE, LRECL, RKP, and KEYLEN DCB attributes, and the volume sequence number and expiration date are copied from the data set label. If text units for these parameters are specified in addition to this key, that specification overrides the corresponding parameter that was copied. This key is mutually exclusive with a DCB reference to a ddname (see below). When this key is specified, # must be one, LEN is the length of the dsname, and PARM contains the dsname.

Example: to specify DCB reference to the dsname ABC, code

```
KEY     #       LEN     PARM
002C    0001    0003    C1 C2 C3
```

DCB Reference to a Ddname specification - Key = X'002D'

This key is used to specify that DCB information is to be retrieved from the currently allocated data set associated with the specified ddname. For time sharing users, the Expiration Date and INPUT/OUTPUT ONLY specifications are also retrieved. This key is mutually exclusive with a DCB reference to a dsname specification. Any DCB attributes, Expiration Date, and INPUT/OUTPUT ONLY options specified in addition to this key override the corresponding specifications associated with the ddname. When this key is specified, # must be one, LEN is the length of the ddname, and PARM contains the ddname.

Example: to specify DCB reference to the ddname DD1, code

```
KEY     #       LEN     PARM
002D    0001    0003    C4 C4 F1
```

Sysout Remote Work Station specification - Key = X'0058'

This key is used to specify that the sysout data set being allocated is to be routed to a remote work station when it is unallocated. The Sysout key must also be specified when this key is specified. When this key is coded, # must be one, LEN is the length of the remote work station name (maximum of 7), and PARM contains the remote user name.

Example: to specify the remote work station USER01, code

```
KEY     #       LEN     PARM
0058    0001    0006    E4 E2 C5 D9 F0 F1
```

Sysout Hold Queue specification - Key = X'0059'

This key is used to specify that the sysout data set being allocated is to be placed on the Hold Queue when it is unallocated. The Sysout key must also be specified when this key is specified. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify Hold, code

```
KEY      #        LEN      PARM
0059     0000     -        -
```

MSVGP specification - Key = X'005E'

This key is used to specify a group of MSS volumes. This key is mutually exclusive with SYSOUT, QNAME, and volume serial specifications. When this key is specified, # must be one, LEN is the actual length of the MSVGP name, and PARM contains the group name.

Example: to specify a MSS volume group of SYSGROUP, code

```
KEY      #        LEN      PARM
005E     0001     0008     E2 E8 E2 C7 D9 D6 E4 D7
```

Hex Text Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
002E	DALBFALN	Specifies buffer alignment.
002F	DALBFTEK	Specifies the buffering technique.
0030	DALBLKSZ	Specifies blocksize.
0031	DALBUFIN	Specifies the receiving buffering count.
0032	DALBUFL	Specifies the buffer length.
0033	DALBUFMX	Specifies the buffer count per line.
0034	DALBUFNO	Specifies the buffer count per DCB.
0035	DALBUFOF	Specifies the buffer offset.
0036	DALBUFOU	Specifies the sending buffer count.
0037	DALBUFRQ	Specifies the buffer count per GET macro instruction.
0038	DALBUFSZ	Specifies the line group buffer size.
0039	DALCODE	Specifies the data's paper tape code.
003A	DALCPRI	Specifies the relative sending and receiving priority.
003B	DALDEN	Specifies the magnetic tape density.
003C	DALDSORG	Specifies the data set organization.
003D	DALEROPT	Specifies reading and writing error options.
003E	DALGNCP	Specifies the GAM-I/O count per WAIT macro instruction.
003F	DALINTVL	Specifies the line polling interval per group.
0040	DALKYLEN	Specifies the data set key lengths.
0041	DALLIMCT	Specifies the search limit.
0042	DALLRECL	Specifies the logical record length.
0043	DALMODE	Specifies card punch/reader operational mode.
0044	DALNCP	Specifies the READ/WRITE count per CHECK.
0045	DALOPTCD	Specifies the control program's operational services.
0046	DALPCIR	Specifies the relationship of the receiving PCI to the allocation and freeing of buffers.
0047	DALPCIS	Specifies the relationship of the sending PCI to the allocation and freeing of buffers.
0048	DALPRTSP	Specifies printer line spacing.
0049	DALRECFM	Specifies the record format.
004A	DALRSRVF	Specifies the first insertion buffer's reserve byte count.
004B	DALRSRVS	Specifies the secondary insertion buffer's reserve byte count.
004C	DALSOWA	Specifies the user's telecommunications input work areas size.
004D	DALSTACK	Specifies the card punch's stacker bin.
004E	DALTHRSH	Specifies the use percentage of nonreusable direct access message queue records per flush closedown.
004F	DALTRTCH	Specifies the 7-track tape recording technique.
0051	DALIPLTX	Specifies a TCAM network control program name.
0054	DALDIAGN	Requests OPEN/CLOSE/EOV diagnostic trace option.
005A	DALFUNC	Specifies the type of data set to be opened for the 3525 Card-Read-Punch-Print.
005B	DALFRID	Specifies a SYS1.IMAGELIB member for 3886 input.

Figure 9. DCB Attributes (Used with Verb Code 01) — Text Unit Keys, Mnemonics, and Functions

DCB Attribute Text Units

The following keys are used to specify DCB attributes. These attributes are described in *OS/VS2 JCL*, GC28-0692 under the DCB keyword subparameter specification and in the *OS/VS2 Data Management Macro Instructions, GC26-3793.*

BFALN specification - Key = X'002E'

This key is used to specify buffer alignment. The GNCP specification is mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains:

X'01' for fullword not a doubleword boundary (F)
X'02' for doubleword boundary (D)

Example: to specify doubleword boundary, code

```
KEY     #       LEN     PARM
002E    0001    0001    02
```

BFTEK specification - Key = X'002F'

This key is used to specify the buffering technique. The GNCP specification is mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains:

X'08' for dynamic buffering (D)
X'10' for exchange buffering (E)
X'20' for record buffering (R)
X'40' for simple buffering (S)
X'60' for record area buffering (A)

Example: to specify exchange buffering, code

```
KEY     #       LEN     PARM
002F    0001    0001    10
```

BLKSIZE specification - Key = X'0030'

This key is used to specify the block size. The BUFSIZE specification is mutually exclusive with this key. When this key is specified, # must be one, LEN must be two, and PARM contains the block size.

Example: to specify a block size of 80, code

```
KEY     #       LEN     PARM
0030    0001    0002    0050
```

BUFIN specification - Key = X'0031'

This key is used to specify the number of buffers to be initially assigned for receiving operations for each line in the line group. The BUFNO and BUFRQ specifications are mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains the number of buffers.

Example: to specify 2 buffers, code

```
KEY     #       LEN     PARM
0031    0001    0001    02
```

BUFL specification - Key = X'0032'

This key is used to specify the buffer length. When this key is specified, # must be one, LEN must be two, and PARM contains the buffer length.

Example: to specify a buffer length of 80, code

```
KEY     #       LEN     PARM
0032    0001    0002    0050
```

BUFMAX specification - Key = X'0033'

This key is used to specify the maximum number of buffers to be allocated to a line at one time. The NCP specification is mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains the number of buffers.

Example: to specify 4 buffers, code

```
KEY     #       LEN     PARM
0033    0001    0001    04
```

BUFNO specification - Key = X'0034'

This key is used to specify the number of buffers to be assigned to the data control block. The BUFIN, BUFOUT, and BUFRQ specifications are mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains the number of buffers.

Example: to specify 2 buffers, code

```
KEY     #       LEN     PARM
0034    0001    0001    02
```

BUFFOFF specification - Key = X'0035'

This key is used to specify the buffer offset. When this key is specified, # and LEN must be one, PARM contains:

X'80' if the block prefix is four bytes long and contains the block length (L)

or, the length of the block prefix (maximum of X'63).

Example: to specify offset of 16,.code

```
KEY     #       LEN     PARM
0035    0001    0001    10
```

BUFOUT specification - Key = X'0036'

This key is used to specify the number of buffers to be assigned initially for sending operations for each line in the line group. The BUFNO and BUFRQ specifications are mutually exclusive with this key. When this key is specified, # and LEN must be one, PARM contains number of buffers.

Example: to specify 4 buffers, code

```
KEY     #       LEN     PARM
0036    0001    0001    04
```

BUFRQ specification - Key = X'0037'

This key is used to specify the number of buffers to be requested in advance for the GET macro instruction. The BUFNO, BUFIN, and BUFOUT specifications are mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains the number of buffers.

Example: to specify 4 buffers, code

```
KEY     #       LEN     PARM
0037    0001    0001    04
```

BUFSZ specification - Key = X'0038'

This key is used to specify the length in bytes of each of the buffers to be used for all lines in a particular line group. The BLKSIZE specification is mutually exclusive with this key. When this key is specified, # must be one, LEN must be two, and PARM contains the buffer length.

Example: to specify a buffer length of 80, code

```
KEY     #       LEN     PARM
0038    0001    0002    0050
```

CODE specification - Key = X'0039'

This key is used to specify the paper tape code in which the data is punched. The KEYLEN, MODE, PRTSP, STACK, and TRTCH specifications are mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains:

X'02' for Teletype 5-track (T)
X'04' for USASCII 8-track (A)
X'08' for National Cash Register 8-track (C)
X'10' for Burroughs 7-track (B)
X'20' for Friden 8-track (F)
X'40' for IBM BCD 8-track (I)
X'80' for no conversion (N)

Example: to specify USASCII, code

```
KEY     #       LEN     PARM
0039    0001    0001    04
```

CPRI specification - Key = X'003A'

This key is used to specify the relative priority to be given to sending and receiving operations. The THRESH specification is mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains:

X'01' for send priority (S)
X'02' for equal priority (E)
X'04' for receiving priority (R)

Example: to specify equal priority, code

```
KEY     #       LEN     PARM
003A    0001    0001    02
```

DEN specification - Key = X'003B'

This key is used to specify the magnetic tape density. When this key is specified, # and LEN must be one, and PARM contains:

X'03' for 200 bpi 7 - track (0)
X'43' for 556 bpi 7 - track (1)
X'83' for 800 bpi 7 - track, 800 bpi 9 - track (2)
X'C3' for 1600 bpi 9 - track (3)
X'D3' for 6250 bpi 9 - track (4)

Example: to specify 1600 bpi 9 - track, code

```
KEY     ##      LEN     PARM
003B    0001    0001    C3
```

DSORG specifications - Key = X'003C'

This key is used to specify the data set organization. When this key is specified, # must be one, LEN must be two, and PARM contains:

X'0004' for TCAM 3705
X'0008' for VSAM
X'0020' for TCAM message queue (TQ)
X'0040' for TCAM line group (TX)
X'0080' for Graphics (GS)

X'0200' for Partitioned (PO)
X'0300' for Partitioned Unmovable (POU)
X'0400' for government of message transfer to or from a telecommunications message processing queue (MQ)
X'0800' for Direct access message queue (CQ)
X'1000' for Communication line group (CX)
X'2000' for Direct access (DA)
X'2100' for Direct access unmovable (DAU)
X'4000' for Physical Sequential (PS)
X'4100' for Physical Sequential Unmovable (PSU)

Example: to specify Partitioned Organization, code

```
KEY     #       LEN     PARM
003C    0001    0002    0200
```

EROPT specification - Key = X'003D'

This key is used to specify the option to be executed if an error occurs in writing or reading a record. When this key is specified, # and LEN must be one, and PARM contains:

X'10' for online BSAM testing (T)
X'20' to cause abnormal end of task (ABE)
X'40' to skip the block causing the error (SKP)
X'80' to accept the block causing the error (ACC)

Example: to specify the SKP error option, code

```
KEY     #       LEN     PARM
003D    0001    0001    40
```

GNCP specification - Key = X'003E'

This key is used to specify the maximum number of GAM input/output macro instructions that will be issued before a WAIT macro instruction. This key is mutually exclusive with the BFTEK and BFALN specifications. When this key is specified, # and LEN must be one, and PARM contains the GNCP value.

Example: to specify a GNCP value of four, code

```
KEY     #       LEN     PARM
003E    0001    0001    04
```

INTVL specification - Key = X'003F'

This key is used to specify the polling interval for the lines in the line group. This key is mutually exclusive with UCS and FCB specifications. When this key is specified, # and LEN must be one, and PARM contains the INTVL value.

Example: to specify an INTVL value of 10, code

```
KEY     #       LEN     PARM
003F    0001    0001    0A
```

KEYLEN specification - Key = X'0040'

This key is used to specify the length, in bytes, of the keys used in the data set. The CODE, MODE, PRTSP, STACK, and TRTCH specifications are mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains the key length.

Example: to specify a key length of eight, code

```
KEY     #       LEN     PARM
0040    0001    0001    08
```

LIMCT specification - Key = X'0041'

This key is used to specify the search limit. When this key is specified, # must be one, LEN must be three, and PARM contain the search limit value.

Example: to specify a search limit of 1000, code

```
KEY     #       LEN     PARM
0041    0001    0003    0003E8
```

LRECL specification - Key = X'0042'

This key is used to specify the actual or maximum length, in bytes, of a logical record. When this key is specified, # must be one, LEN must be two, and PARM contains:

X'8000' if, for variable length spanned records processed under QSAM and BSAM, the logical records exceed 32,756 bytes (X)

or, the logical record length.

Example: to specify a logical record length of 80, code

```
KEY     #       LEN     PARM
0042    0001    0002    0050
```

MODE specification - Key = X'0043'

This key is used to specify the mode of operations for a card reader or punch. This key is mutually exclusive with the CODE, KEYLEN, PRTSP and TRTCH specification. When this key is specified, # and LEN must be one, and PARM contains:

X'40 for EBCDIC mode (E)
X'50' for EBCDIC, read column eliminate mode (ER)
X'60' for EBCDIC, optical mark read mode (EO)
X'80' for card image mode (C)
X'90' for card image, read column eliminate mode (CR)
X'A0' for card image, optical mark read mode (CO)

Example: to specify EBCDIC mode, code

```
KEY     #       LEN     PARM
0043    0001    0001    40
```

NCP specification - Key = X'0044'

This key is used to specify the maximum number of READ or WRITE macro instructions issued before a CHECK macor instruction is issued. This key is mutually exclusive with BUFMAX specification. When this key is specified, # and LEN must be one, and PARM contains the NCP value.

Example: to specify a NCP value of two, code

```
KEY     #       LEN     PARM
0044    0001    0001    02
```

OPTCD specification - Key = X'0045'

This key is used to specify optional services to be performed by the control program. When this key is specified, # and LEN must be one, and PARM contains:

X'01' for Relative block addressing (R)
X'02' for user totaling facility (T)
X'04' for reduced tape error recovery, or direct access search direct (Z)
X'08' for direct addressing (A), or for translation of ASCII to or from EBCDIC (Q)

X'10' for feedback (F), or for hopper empty exit (H), or for online correction for Optical Readers (O)
X'20' for chained scheduling, or TCAM segment identification (C), or for extended search (E)
X'40' for end-of-file recognition to be disregarded for tapes (B), or for allowance of data checks caused by an invalid character, or TCAM work unit is to be handled as a message (U)
X'80' for write validity check (W)

Notes: When more than one OPTCD value is to be specified, PARM contains the sum of the values.

Examples: to specify OPTCD value U, code

```
KEY     #       LEN     PARM
0045    0001    0001    40
```

Example: to specify OPTCD values U and C, code

```
KEY     #       LEN     PARM
0045    0001    0001    60
```

Receiving PCI specification - Key = X'0046'

This key is used to specify the relationship of program-controlled interrupts (PCI) during receiving operations to the allocation and freeing of buffers. When this key is specified, # and LEN must be one, and PARM contains:

X'02' for a PCI and no new buffer allocated (R)
X'08' for no PCI's (N)
X'20' for a PCI and new buffer allocated (A)
X'80' for a PCI, new buffer allocated, and the first buffer remains allocated (X)

Example: to specify no PCI's during receiving operations, code

```
KEY     #       LEN     PARM
0046    0001    0001    08
```

Sending PCI specification - Key = X'0047'

This key is used to specify the relationship of PCI's during sending operations to the allocation and freeing of buffers. When this key is specified, # and LEN contain one, and PARM contains:

X'01' for a PCI and no new buffer allocated (R)
X'04' for no PCI's (N)
X'10' for a PCI and a new buffer allocated (A)
X'40' for a PCI, new buffer allocated, and first buffer remains allocated (X)

Example: to specify no PCI's during sending operations, code

```
KEY     #       LEN     PARM
0047    0001    0001    04
```

PRTSP specification - Key = X'0048'

This key is used to specify printer line spacing. The CODE, KEYLEN, MODE, STACK, and TRTCH specifications are mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains:

X'01' for no spacing (0)
X'09' for one line spacing (1)
X'11' for two line spacing (2)
X'19' for three line spacing (3)

Example: to specify no spacing, code

```
KEY     #       LEN     PARM
0048    0001    0001    01
```

RECFM specification - Key = X'0049'

This key is used to specify the record format. When this key is specified, # and LEN must be one, and PARM contains:

X'02' for machine code printer control characters in record (M), or for complete QTAM record (R)
X'04' for ASA printer control characters in record (A), or for complete QTAM message (G)
X'08' for standard fixed records, spanned variable records, or segment of QTAM message (S)
X'10' for blocked records (B)
X'20' for variable ASCII records (D), or for track overflow (T)
X'40' for variable records (V)
X'80' for fixed records (F)
X'C0' for undefined records (U)

Notes: When more than one RECFM value is to be specified in combination, PARM contains the sum of the values.

Example: to specify fixed records, code

```
KEY     #       LEN     PARM
0049    0001    0001    80
```

First Buffer RESERVE specification - Key = X'004A'

This key is used to specify the number of bytes to be reserved in other than the first buffer for insertion of data by the DATETIME AND SEQUENCE macros. The UCS specification is mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains the number of bytes to reserve.

Example: to reserve 8 bytes in secondary buffers, code

```
KEY     #       LEN     PARM
004B    0001    0001    08
```

Secondary Buffer RESERVE specification - Key = X'004B'

This key is used to specify the number of bytes to be reserved in other than the first buffer for insertion of data by the DATETIME and SEQUENCE macro instructions. The UCS specification is mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains the number of bytes to reserve.

Example: to reserve 8 bytes in secondary buffers, code

```
KEY     #       LEN     PARM
004B    0001    0001    08
```

SOWA specification - Key = X'004C'

This key is used to specify the size, in bytes, of the user-provided input work areas for telecommunication jobs. When this key is specified, # must be one, LEN must be two, and PARM contains the SOWA value.

Example: to specify a 256 byte work area, code

```
KEY     #       LEN     PARM
004C    0001    0002    0100
```

STACK specification - Key = '004D'

This key is used to specify the stacker bin to receive cards. The CODE, KEYLEN, PRTSP, and TRTCH specification are mutually exclusive with this key. When this key is specified, # and LEN are one, and PARM contains:

X'01' for bin 1 (1)
X'02' for bin 2 (2)

Example: to specify stacker 2, code

```
KEY     #       LEN     PARM
004D    0001    0001    02
```

THRESH specification - Key = X'004E'

This key is used to specify the percentage of nonreusable disk message queue records to be used before a flush closedown occurs. This key is mutually exclusive with the CPRI specification. When this key is specified, # and LEN must be one, and PARM contains the percentage.

Example: to specify a THRESH percentage of 99, code

```
KEY     #       LEN     PARM
004E    0001    0001    63
```

TRTCH specification - Key = X'004F'

This key is used to specify the recording technique for 7 - track tape. The KEYLEN, MODE, CODE, STACK, and PRTSP specifications are mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains:

X'13' for data conversion (C)
X'23' for even parity (E)
X'2B' for even parity and BCD/EBCDIC translation, (ET)
X'3B' for BCD/EBCDIC translation (T)

Example: to specify even parity, code

```
KEY     #       LEN     PARM
004F    0001    0001    23
```

IPLTXTID specification - Key = X'0051'

This key is used to specify the name of a TCAM network control program. This key is mutually exclusive with the DSNAME MEMBER NAME, and QNAME specifications. When this key is specified, # must be one, LEN is the length of the name (maximum of 7), and PARM contains the name.

Example: to specify an IPLTXTID value of PGM, code

```
KEY     #       LEN     PARM
0051    0001    0003    D7 D7 D4
```

Diagnostic Trace specification (DIAGNS = TRACE) - Key = X'0054'

This key requests the OPEN/CLOSE/EOV trace option which gives a module by module trace of OPEN/CLOSE/EOV's workarea and the user's DCB. When this key is specified, # must be zero. LEN and PARM must not be specified. (GTF must be active in the system while the job that requested the trace is running).

Example: to specify the Diagnostic Trace specification, code

```
KEY     #       LEN     PARM
0054    0000    -       -
```

FUNC = function specification - Key = X'005A'

This key can be used with BSAM and QSAM and specifies the type of data set to be opened for the 3525 Card Read-Punch-Print. When this key is specified, # and LEN must be one, and PARM must be one of the following values:

X'10' for W
X'12' for WT
X'14' for WX
X'16' for WXT
X'20' for P
X'30' for PW
X'34' for PWX
X'36' for PWXT
X'40' for R
X'50' for RW
X'52' for RWT
X'54' for PWX
X'56' for PWXT
X'60' for RP
X'68' for RPD
X'70' for RPW
X'74' for RPWX
X'76' for RPWXT
X'78' for RPWD
X'80' for I

Where:

- D is data protection for a punch data set
- I is interpret punch data set
- P is punch
- R is read
- T is two line printer
- W is print
- X is printer

Note: If this information is not supplied by any source, the system assumes P.

- D, X, and T cannot be coded alone
- If D is specified as part of a value, the FCB image-id key must also be specified giving the image identifier for the data protection image.

Example: to specify FUNC=RPWD

```
KEY     #       LEN     PARM
005A    0001    0001    78
```

FRID = member specification - Key = X'005B'

This key is used to specify the last four characters of a SYS1.IMAGELIB member to be used in the interpretation of documents for input to SHARK. The characters must be alphanumeric or national, and if the member has a name with a length less than four, the entire name must be specified. This key is mutually exclusive with the FCB specification. When this key is specified # must be one, LEN is the number of characters specified, and PARM contains the characters of the member name.

Example: to specify the last four characters of member name SHARK1, code

```
KEY     #       LEN     PARM
005B    0001    0004    C1D9D2F1
```

Hex Test Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
0050	DALPASSW	Specifies the password for a protected data set.
0052	DALPERMA	Specifies the permanently allocated attribute.
0053	DALCNVRT	Specifies the convertible attribute.
0055	DALRTDDN	Requests the return of the associated ddname.
0056	DALRTDSN	Requests the return of the allocated data set's name.
0057	DALRTORG	Requests the return of data set organization.
005D	DALRTVOL	Requests the return of the volume serial number.

Figure 10. Non-JCL Dynamic Allocation Functions (Used with Verb Code 01) — Text Unit Keys, Mnemonics, and Functions

Non-JCL Dynamic Allocation Functions

The following keys do not have JCL equivalents, rather they only have meaning to dynamic allocation.

Password specification - Key = X'0050'

This key is used to specify the password of a password protected data set. The dsname key must also be specified when this key is specified. When this key is specified, # must be one, LEN contains the length of password, and PARM contains the password.

Example: to specify the password, MYKEY, code

```
KEY      #        LEN      PARM
0050     0001     0005     D4 E8 D2 C5 E8
```

Permanently Allocated Attribute specification - Key = X'0052'

This key is used to specify that the Permanently Allocated attribute is to be assigned to this allocation. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify assignment of the Permanently Allocated attribute, code

```
KEY      #        LEN      PARM
0052     0000     -        -
```

Convertible Attribute specification - Key = X'0053'

This key is used to specify that the Convertible attribute is to be assigned to this allocation. (Note: this specification is defaulted if the Permanently Allocated attribute text unit is not specified.) When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify assignment of the Convertible attribute, code

```
KEY      #        LEN      PARM
0053     0000     -        -
```

Ddname Return specification - Key = X'0055'

This key is used to specify that the ddname that is associated with the allocation be returned to the dynamic allocation caller. When this key is specified, # must be one, LEN must be eight, and PARM is an eight byte field. Dynamic allocation will place the allocated ddname in PARM and update LEN to the length of this ddname.

Example: to specify that the allocated ddname be returned, code

```
KEY      #        LEN      PARM
0055     0001     0008     --------
```

This specification would be updated for the allocation of the ddname, DD1, as follows:

```
KEY      #        LEN      PARM
0055     0001     0003     C4 C4 F1-----
```

Dsname Return specification - Key = X'0056'

This key is used to specify that the dsname that is allocated be returned to the dynamic allocation caller. When this key is specified, # must be one, LEN must be forty-four, and PARM is a forty-four byte field. Dynamic allocation will place the allocated dsname in PARM, and update LEN to the length of this dsname.

Example: to specify that the allocated dsname be returned, code

```
KEY      #        LEN      PARM
0056     0001     002C     --------...--
```

This specification would be updated for the allocation of the dsname, ABC, as follows.

```
KEY      #        LEN      PARM
0056     0001     0003     C1 C2 C3-----...--
```

DSORG Return specification - Key = X'0057'

This key is used to specify that the data set organization be returned to the dynamic allocation caller. When this key is specified, # must be one, LEN must be two, and PARM is a two byte field. Dynamic allocation will set PARM as follows:

X'0000' if DSORG cannot be determined by dynamic allocation.
X'0004' if TR
X'0008' if VSAM
X'0020' if TQ
X'0040' if TX
X'0080' if GS
X'0200' if PO
X'0300' if POU
X'0400' if MQ
X'0800' if CQ
X'1000' if CX
X'2000' if DA
X'2100' if DAU
X'4000' if PS
X'4100' if PSU
X'8000' if IS
X'8100' if ISU

Example: to specify that the DSORG be returned, code

```
KEY      #        LEN      PARM
0057     0001     0002     -
```

This specification would be updated for a DSORG of PS as follows:

```
KEY      #        LEN      PARM
0057     0001     0002     4000
```

Volume Serial Return specification - Key = X'005D'

This key is used to specify that the volume serial associated with the data set being allocated be returned. Only the first volume serial of the multiple volume data set is returned, and volume sequence number, if any, is ignored. When this key is specified, # must be one, LEN must be six, and PARM is a six byte field. Dynamic allocation will place the allocated volume serial in PARM. If no volume serial is associated with the data set, for example, a VIO or Job Entry Subsystem data set, # will be set to zero.

Example: to specify that the allocated volume serial be returned, code

```
KEY      #        LEN      PARM
005D     0001     0006     ------
```

Hex Text Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
0001	DUNDDNAM	Specifies the ddname of the resource to be unallocated.
0002	DUNDSNAM	Specifies the data set to be unallocated.
0003	DUNMEMBR	Specifies the PDS member to be unallocated.
0005	DUNOVDSP	Specifies an overriding disposition.
0007	DUNUNALC	Specifies unallocation even if a permanently allocated resource.
0008	DUNREMOV	Specifies the removal of the "in-use" attribute, even if not permanently allocated.
000A	DUMOVSNH	Specifies "nohold" status for an unallocated SYSOUT data set.
0018	DUNOVCLS	Specifies an overriding SYSOUT class.
0058	DUNOVSUS	Specifies an overriding remote work station.
0059	DUMOVSHQ	Puts the SYSOUT data set on the hold queue and overrides previous "nohold" specifications.

Figure 11. Dynamic Unallocation (Verb Code 02) — Text Unit Keys, Mnemonics, and Functions

Dynamic Unallocation Text Units

The following text units are used with the unallocation function of Dynamic Allocation (VERB code X'02').

Ddname specification - Key = X'0001'

This key is used to specify the ddname of the resource to be unallocated. When this key is specified, # must be one, LEN is the length of the ddname field, and PARM contains the ddname.

Example: to specify the ddname, DD1, code

```
KEY     #       LEN     PARM
0001    0001    0003    C4 C4 F1
```

Dsname specification - Key = X'0002'

This key is used to specify the data set name to be unallocated. When this key is specified, # must be one, LEN contains the length of the dsname, and PARM contains the dsname.

Example: to specify the dsname, MYDATA, code

```
KEY     #       LEN     PARM
0002    0001    0006    D4 E8 C4 C1 E3 C1
```

Membername specification - Key = X'0003'

This key is used to specify that a particular member of the data set is to be unallocated. Dsname must also be specified when this key is specified. When this key is specified, # must be one, LEN is the length of the member name, and PARM contains the membername.

Example: to specify the membername, MEM1, code

```
KEY     #       LEN     PARM
0003    0001    0004    D4 C5 D4 F1
```

Overriding Disposition specification - Key = X'0005'

This key is used to specify a disposition which overrides the disposition assigned to a data set when it was allocated. When this key is specified, # and LEN must be one, and PARM contains:

X'01' for an overriding disposition of UNCATLG
X'02' for an overriding disposition of CATLG
X'04 for an overriding disposition of DELETE
X'08' for an overriding disposition of KEEP

Example: to specify an overriding disposition of CATLG, code

```
KEY     #       LEN     PARM
0005    0001    0001    02
```

Unalloc Option specification - Key = X'0007'

This key is used to specify that unallocation is to occur even if the resource has the Permanently Allocated attribute. The Remove option specification (see below) is mutually exclusive with this key. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify the Unalloc option, code

```
KEY     #       LEN     PARM
0007    0000    -       -
```

Remove Option specification - Key = X'0008'

This key is used to specify that the In-Use attribute is to be removed even if the resource does not have the Permanently Allocated attribute. The Unalloc Option specification is mutually exclusive with this key. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify the Remove Option, code

```
KEY     #       LEN     PARM
0008    0000    -       -
```

Overriding Sysout Nohold specification - Key = X'000A'

This key is used to specify that the Sysout data set being unallocated is not to be placed on the Hold Queue. This specification overrides the Hold/Nohold specification assigned when the data set was allocated. This specification is ignored if the data set is not a Sysout data set. The Overriding Hold specification is mutually exclusive with this key. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify Nohold, code

```
KEY     #       LEN     PARM
000A    0000    -       -
```

Overriding Sysout Class specification - Key = X'0018'

This key is used to specify a sysout class which overrides the class assigned when the sysout data set was allocated. This key is ignored for non-Sysout data sets. When specified, # and LEN must be one, and PARM contains the overriding class.

Example: to specify an overriding class of C, code

```
KEY      #        LEN      PARM
0018     0001     0001     C3
```

Overriding Sysout Remote Work Station specification - Key = X'0058'

This key is used to specify that the Sysout data set being unallocated is to be routed to a remote user. This specification overrides the remote work station specification assigned when the data set was allocated. This specification is ignored if the data set is not a Sysout data set. When this key is specified, # must be one, LEN is the length of the remote work station name (maximum of 7), and PARM contains the remote user name.

Example: to specify the remote work station, USER01, code

```
KEY      #        LEN      PARM
0058     0001     0006     E4 E2 C5 D9 F0 F1
```

Overriding Sysout Hold Queue specification - Key = X'0059'

This key is used to specify that the Sysout data set being unallocated is to be placed on the Hold Queue. This specification overrides the Hold/Nohold specification assigned when the data set was allocated. This specification is ignored if the data set is not a Sysout data set. The Overriding Nohold specification is mutually exclusive with this key. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify Hold, code

```
KEY      #        LEN      PARM
0059     0000     -        -
```

Hex Text Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
0001	DCCDDNAM	Specifies the ddnames to be concatenated.
0004	DCCPERMC	Specifies the permanently concatenated attribute.

Figure 12. Dynamic Concatenation (Verb Code 03) — Text Unit Keys, Mnemonics, and Functions

Dynamic Concatenation Text Units

The text units for the concatenation function of Dynamic Allocation (VERB code X'03') are as follows:

Ddname specification - Key = X'0001'

This key is used to specify the ddnames that are associated with the data sets to be concatenated. When this key is specified, # is the number of ddnames being specified (a minimum of two), LEN is length of the immediately following ddname field and PARM contains a ddname.

Example: to specify concatenation of SYSLIB to MYLIB, code

```
KEY     #       LEN     PARM          LEN     PARM
0001    0002    0005    D4E8D3C9C2    0006    E2E8E2D3C9C2
```

Permanently Concatenated Attribute specification - Key = X'0004'

This key is used to specify that the concatenated group be assigned the Permanently Concatenated attribute. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify assignment of the Permanently Concatenated Attribute, code

```
KEY     #       LEN     PARM
0004    0000    -       -
```

Hex Text Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
0001	DDCDDNAM	Specifies the ddname of the group to be deconcatenated.

Figure 13. Dynamic Deconcatenation (Verb Code 04) — Text Unit Key, Mnemonic, and Function

Dynamic Deconcatenation Text Units

The text unit for the deconcatenation function of Dynamic Allocation (VERB code X'04') is:

Ddname specification - Key = X'0001'

This key is used to specify the ddname of the concatenated group that is to be deconcatenated. This key must be specified. In specifying this text unit, # must be one, LEN is the length of the ddname field and PARM contains the ddname.

Example: to specify the ddname, DD1, code

```
KEY      #        LEN      PARM
0001     0001     0003     C4 C4 F1
```

Hex Text Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
0001	DRITCBAD	Removes the "in-use" attribute from all resources associated with the specified TCB address.
0002	DRICURNT	Removes the "in-use" attribute from all resources but those of the current task and its higher-level tasks.

Figure 14. In-Use Attribute Removal (Verb Code 05) — Text Unit Keys, Mnemonics, and Functions

Text Units for Removing the In-Use Attribute Based on Task-ID

The following text units are for the 'Removal of the In-Use attribute based on task-ID function' of Dynamic Allocation (VERB code X'05').

TCB Address specification - Key = X'0001'

This key is used to specify that the In-Use attribute is to be removed from all resources associated with the specified TCB address. The Current Task Option specification (see below) is mutually exclusive with this key. When this key is specified, # must be one, LEN must be four, and PARM contains the TCB address.

Example: to specify the TCB address 22AC0, code

```
KEY      #        LEN      PARM
0001     0001     0004     00022AC0
```

Current Task Option specification - Key = X'0002'

This key is used to specify that the In-Use attribute is to be removed from all resources except those associated with the current task, its direct ancestors, and the initiator. This key is mutually exclusive with the TCB Address specification. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify the Current Task Option, code

```
KEY      #        LEN      PARM
0002     0000     -        -
```

Hex Text Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
0001	DDNDDNAM	Specifies the ddname to be allocated.
0002	DDNRTDUM	Requests a dummy data set indication.

Figure 15. Ddname Allocation (Verb Code 06) — Text Unit Keys, Mnemonics, and Functions

Ddname Allocation Text Units

The following text units are used with the ddname allocation function (VERB code X'06').

Ddname specification - Key = X'0001'

This key is used to specify the ddname to be allocated. This text unit must be specified. When this key is specified, # must be one, LEN contains the length of the ddname field, and PARM contains the ddname.

Example: to specify the ddname, SYSLIB, code

```
KEY      #        LEN      PARM
0001     0001     0006     E2 E8 E2 D3 C9 C2
```

Return DUMMY Indication specification - Key = X'0002'

This key is used to request an indication of a DUMMY data set being associated with the specified ddname. When this key is specified, # and LEN must be one, and PARM is a one byte field. Dynamic allocation sets PARM as follows:

X'80' if DUMMY data set associated with the ddname
X'00' otherwise

Hex Text Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
0001	DINDDNAM	Specifies the ddname identifier of the requested information.
0002	DINDSNAM	Specifies the data set for which the information is requested.
0004	DINRTDDN	Requests the return of the associated ddname.
0005	DINRTDSN	Requests the return of the data set name.
0006	DINRTMEM	Requests the return of the PDS membername.
0007	DINRTSTA	Requests the return of the data set's status.
0008	DINRTNDP	Requests the return of the data set's normal disposition.
0009	DINRTCDP	Requests the return of the data set's conditional disposition.
000A	DINRTORG	Requests the return of the data set's organization.
000B	DINRTLIM	Requests the number of resources that must be unallocated before making a new allocation.
000C	DINRTATT	Requests the return of special attribute indications.
000D	DINRTLST	Requests the return of a last relative entry indication.
000E	DINRTTYP	Requests the return of the data set's type (terminal or dummy).
000F	DINRELNO	Specifies the desired allocation information retrieval by relative request number.

Figure 16. Dynamic Information Retrieval (Verb Code 07) — Text Unit Keys, Mnemonics, and Functions

Dynamic Information Retrieval Text Units

The text units for the information retrieval function of Dynamic Allocation (VERB code '07') are as follows:

Ddname specification - Key = X'0001'

This key is used to specify a ddname that identifies the allocation about which information is to be returned. It is mutually exclusive with the dsname and relative entry specifications. When this key is specified, # must be one, LEN is the length of the ddname field and PARM contains the ddname.

Example: to specify the ddname, DD1, code

```
KEY     #       LEN     PARM
0001    0001    0003    C4 C4 F1
```

Dsname specification - Key = X'0002'

This key is used to specify a dsname that identifies the allocation about which information is to be returned. It is mutually exclusive with the ddname and relative entry specifications. When specified, # must be one, LEN is the length of the dsname, and PARM contains the dsname.

Example: to specify the dsname, MYDATA, code

```
KEY     #       LEN     PARM
0002    0001    0006    D4 E8 C4 C1 E3 C1
```

Return Ddname specification - Key = X'0004'

This key is used to specify that the ddname which is associated with the specified allocation be returned. When this key is specified, # must be one, LEN must be eight, and PARM is an eight byte field. Upon return to the caller, PARM is an eight byte field. Upon return to the caller, PARM will contain the ddname and LEN is set to its length.

Example: to specify that the ddname be returned, code

```
KEY     #       LEN     PARM
0004    0001    0008    --------
```

Return Dsname specification - Key = X'0005'

This key is used to specify that the dsname which is associated with the specified allocation be returned. When this key is specified, # must be one, LEN must be forty-four, and PARM is a forty-four byte field. Upon return to the caller, PARM will contain the dsname, and LEN will be set to its length.

Example: to specify that the dsname be returned, code

```
KEY      #        LEN      PARM
0005     0001     002C     --------
```

Return Membername specification - Key = X'0006'

This key is used to specify that the membername which is associated with the specified allocation be returned. When this key is specified, # must be one, LEN must be eight, and PARM is an eight byte field. Upon return to the caller, PARM will contain the membername, and LEN will be set to its length (zero if none).

Example: to specify that the membername be returned, code

```
KEY      #        LEN      PARM
0006     0001     0008     --------
```

Return Status specification - Key = X'0007'

This key is used to specify that the data set status of the specified allocation be returned. When this key is specified, # and LEN must be one, and PARM is a one byte field which is set as follows upon return to the caller:

X'01' for OLD
X'02' for MOD
X'04' for NEW
X'08' for SHR

Example: to specify that the status be returned, code

```
KEY      #        LEN      PARM
0007     0001     0001     -
```

Return Normal Disposition specification - Key = X'0008'

This key is used to specify that the data set normal disposition of the specified allocation be returned. When this key is specified, # and LEN must be one, and PARM is a one byte field which is set as follows upon return to the caller:

X'01' for UNCATLG
X'02' for CATLG
X'04' for DELETE
X'08' for KEEP
X'10' for PASS

Example: to specify that the normal disposition be returned, code

```
KEY      #        LEN      PARM
0008     0001     0001     -
```

Return Conditional Disposition specification - Key = X'0009'

The key is used to specify that the data set conditional disposition of the specified allocation be returned. The values for #, LEN and PARM are the same as for Return Normal Disposition.

Example: to specify that the conditional disposition be returned, code

```
KEY     #       LEN     PARM
0009    0001    0001    -
```

Return Data Set Organization specification Key = X'000A'

This key is used to specify that the data set organization of the specified allocation be returned. When this key is specified, # must be one, LEN must be two, and PARM is a two byte field which is set as follows upon return to the caller:

X'0000' if undetermined
X'0004' if TR
X'0008' for VSAM
X'0020' if TQ
X'0040' if TX
X'0080' for GS
X'0200' for PO
X'0300' for POU
X'0400' for MQ
X'0800' for CQ
X'1000' for CX
X'2000' for DA
X'2100' for DAU
X'4000' for PS
X'4100' for PSU
X'8000' for IS
X'8100' for ISU

Example: to specify that the data set organization be returned, code

```
KEY     #       LEN     PARM
000A    0001    0002    --
```

Return Limit specification - Key = X'000B'

This key is used to specify that the number of resources which must be unallocated before a request can be made which requires the creation of a new allocation be returned. When this key is specified, # must be one, LEN must be two, and PARM is a two byte field. Upon return to the caller, PARM is set to the number.

Example: to specify that the number be returned, code

```
KEY     #       LEN     PARM
000B    0001    0002    --
```

If three resources must be unallocated, this text unit is returned as follows:

```
KEY     #       LEN     PARM
000B    0001    0002    0003
```

Return Dynamic Allocation Attribute specification - Key = X'000C'

This key is used to specify that an indication of Permanently Allocated, Convertible, In-Use, and Permanently Concatenated Attributes be returned. When this key is specified, # and LEN must be one, and PARM is a one byte field. Upon return to the caller, PARM is set as follows:

Bit 0, on if Permanently Concatenated Attribute
Bit 1, on if In-Use Attribute
Bit 2, on if Permanently Allocated Attribute
Bit 3, on if Convertible Attribute
Bit 4-7 reserved

Example: to specify that the Data Set Attributes be returned, code

```
KEY     #       LEN     PARM
000C    0001    0001    -
```

If the allocation has the In-Use and Permanently Allocated attributes, this field is returned as follows:

```
KEY     #       LEN     PARM
000C    0001    0001    60
```

Return Last Entry specification - Key = X'000D'

This key is used to specify that an indication be returned that the relative entry number specified corresponds to the last relative entry. When this key is specified, # and LEN must be one, and PARM is a one byte field. Upon return to the caller, PARM will be set as follows:

X'80' if last relative entry
X'00' otherwise

Example: to specify that the last entry indication be returned, code

```
KEY     #       LEN     PARM
000D    0001    0001    -
```

Return Data Set Type specification - Key = X'000E'

This key is used to specify that an indication of the allocation being a DUMMY data set or terminal allocation be returned. When this key is specified, # and LEN must be one, and PARM is a one byte field. Upon return to the caller, PARM is set as follows:

X'00' otherwise
X'40' if terminal
X'80' if DUMMY data set

Example: to specify that data set type be returned, code

```
KEY     #       LEN     PARM
000E    0001    0001    -
```

Relative Request Number specification - Key = X'000F'

This key is used to specify a relative request number that identifies the allocation about which information is to be returned. The ddname and dsname specifications are mutually exclusive with this text unit. When this key is specified, # must be one, LEN must be two, and PARM contains the relative number.

Example: to specify information is to be returned about the tenth request, code

```
KEY     #       LEN     PARM
000F    0001    0002    000A
```

Note: One of the three preceding text units must be specified.

Part II: Job Entry Subsystem 2 (JES2)

Introduction to JES2: This topic introduces JES2, and describes the configuration that can be specified during JES2 generation and initialization.

JES2 Processing: This topic describes the asepcts of JES2 processing that can be affected by user programming: controlling job submission and queuing, controlling conversion and execution, execution batch scheduling, controlling system output, and controlling RJE.

Miscellaneous JES2 Facilities: This topic discusses the JES2 patching facility, automatic command processing, flow for time sharing and started tasks, and the multi-access spool.

A job entry subsystem provided with MVS is JES2, generally compatible with HASP II. JES2 serves as the point of entry for all jobs, and the function which produces all hardcopy job output. To accomplish these functions, JES2 controls local and remote job entry devices and output devices. A special job entry source, the internal reader facility, allows MVS to submit system jobs: started tasks and time-sharing LOGONs. Tape and disk input are also supported through the internal reader facility. See figure 17 for input/output relationships to the job entry subsystem and MVS.

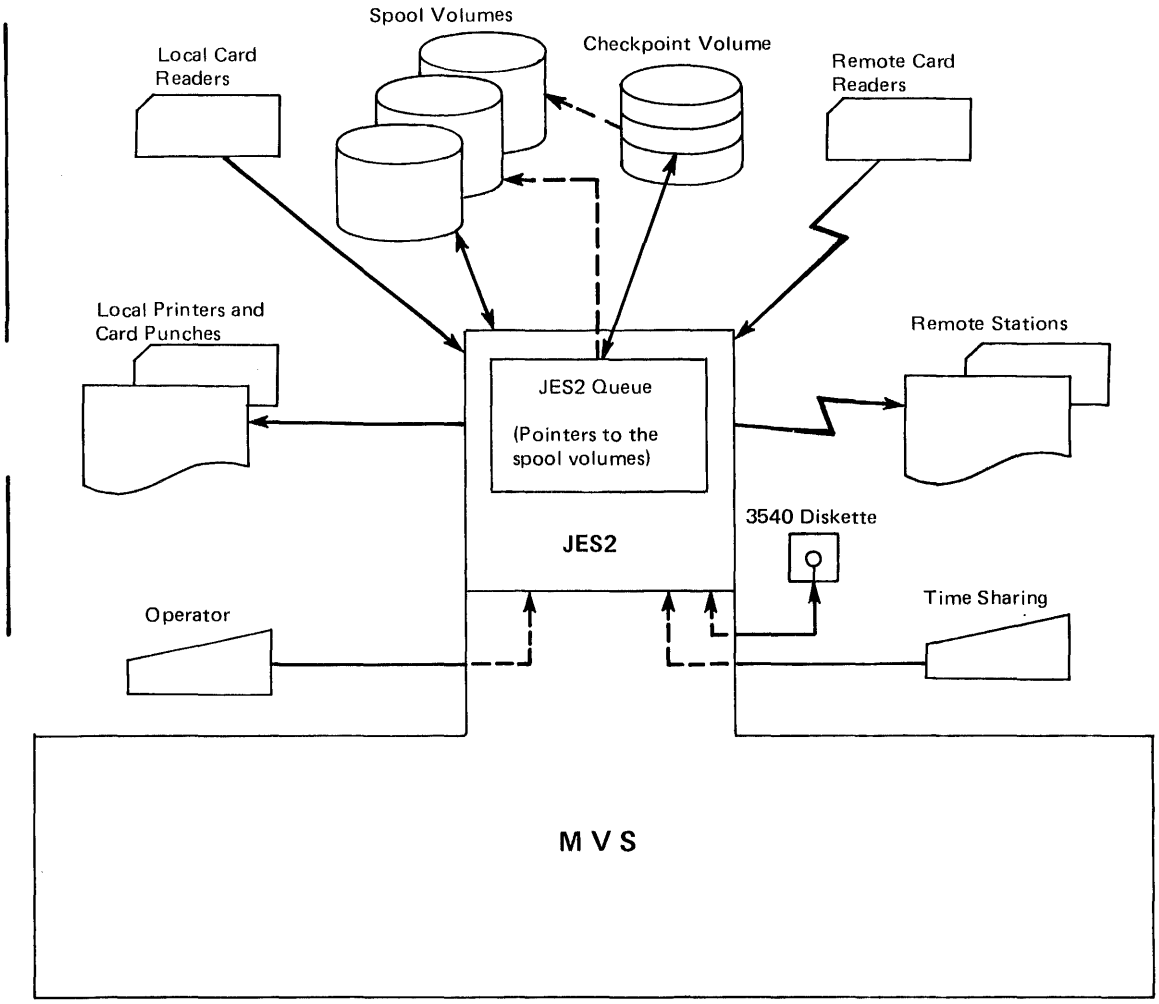

Figure 17. JES2 I/O Relationships

An output interface allows MVS to retrieve output for TSO terminals, and allows a special output facility — the External Writer — to process output to tape, disk, and installation-written writer routines. The output interface also supports an output facility to the 3540 diskette writer (see *OS/VS2 IBM 3540 Programmer's Reference*).

While the job is in MVS, the JES2 job queue residing in pageable storage maintains a record for the job. Job-related system records plus records related to job input and output are maintained on external spool volumes.

The system programmer during JES2 generation and initialization, and the operator during JES2 processing, define and control the configuration of job entry sources and job output destinations. JES2 provides centralized control of job input, queueing, and output, such that all jobs are controlled in the same manner whether submitted from local or RJE (remote job entry) devices, or through the internal reader facility.

Figure 18 describes the organization of this section of the manual. As is suggested in the figure, jobs that are batched for execution (Execution Batch Facility) do not go through the same conversion and execution process as other jobs. The functions described in these sections are:

- Configuration — including configuration for local and RJE devices, generation of JES2, and specification of the internal reader facility and the spool volumes.
- Starting and stopping the job entry subsystem — including starting the default (system generation) subsystem, and initializing through the use of data sets containing initialization parameters.
- Controlling job submission and queueing — including the method of passing a job to JES2, the internal reader facility with support for tape and disks, the RDR procedure, the role of job class and priority in job queuing, priority aging, and the placing of jobs in HOLD status.
- Controlling conversion and execution — including JCL conversion, the job account field scan, defining the procedure library for the job, specifying converter parameters, command authority and recognition for JES2 and VS2, control of initiators, and job monitoring.
- The execution batch facility — including the establishment of the facility and writing an execution batch monitor.
- Controlling output and output devices — including how output is queued and by what function, data set enqueuing, device selection, separator pages and separator cards, overflow, output routing, the external writer and the XWTR procedure.
- Controlling remote job entry — including starting and stopping lines, dedicated versus nondedicated lines, abortive disconnect, printer suspending, password protection, SMF recording, and how to name remote devices.
- Miscellaneous functions such as time sharing control, the automatic operator facility, and modifying JES2 through the card reader.
- Multi-access spool — including an overview, configuration, starting the complex, job submission and queueing, output, and considerations for RJE, TSO, and SMF.

The parameters necessary for controlling the various JES2 functions are described in two manuals: *OS/VS2 System Programming Library: System Generation Reference,* GC26-3792; and *OS/VS2 System Programming Library: Initialization and Tuning Guide,* GC28-0681. These manuals contain detailed descriptions of the implementation of each parameter. When groups of parameters are described in this section, the reader is referred to the System Generation manual or the Initialiaztion manual for implementation details.

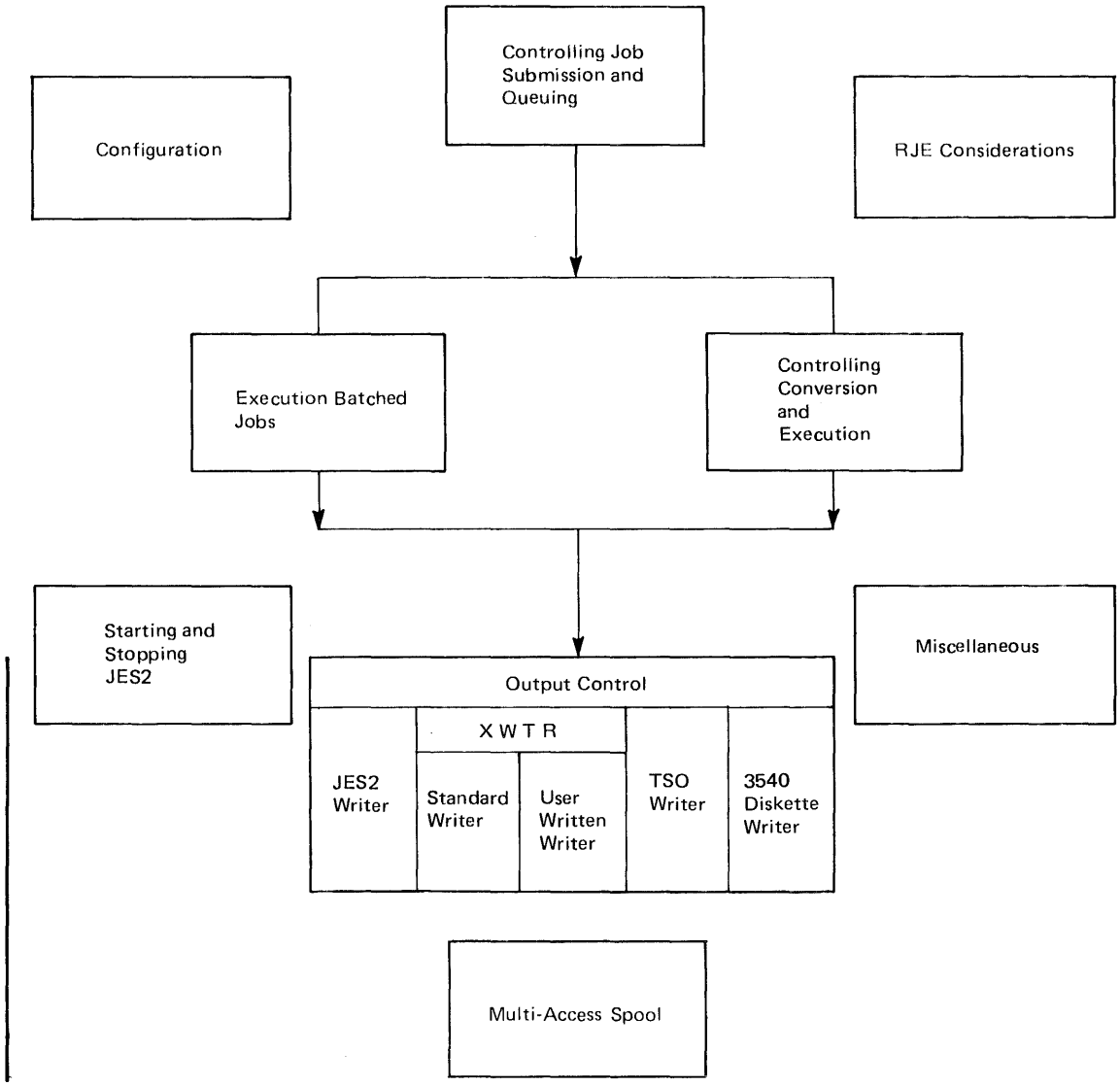

Figure 18. Topics Described Under JES2

Configuration

During JES2 generation and initialization, the system programmer can specify the configuration of JES2 local devices, the JES2 internal reader facility, the JES2 remote lines and devices, and the JES2 spool volumes.

JES2 Generation

JES2 is created by a process called JES2GEN. The system generation process is designed so that the operator can generate JES2 while Stage II of SYSGEN is in progress. Following JES2GEN the operator must issue a START command START JES2BLD in order to linkage edit JES2 into SYS1.LINKLIB and SYS1.LPALIB. The primary JES2 module HASJES20 resides in SYS1.LINKLIB.

The job entry subsystem to be used with MVS must be named in the PRISUB parameter of the SCHEDULR system generation macro instruction. The default name is 'JES2'. To change the name from 'JES2', both the PRISUB parameter and the name of the cataloged procedure in

SYS1.PROCLIB must be changed. To use a subsystem other than JES2, the name must be inserted in PRISUB and the new subsystem placed in SYS1.PROCLIB.

Local Device Configuration

Local devices refer to card readers, printers, and card punches which are attached to the MVS system and which are used for reading jobs and writing output.

During JES2 generation the system programmer specifies the number of readers, printers and punches to be controlled by JES2 via the &NUMRDRS, &NUMPRTS and &NUMPUNS parameters. It is not possible to assign to JES2 more than the number of devices specified without regenerating the JES2 system.

During JES2 initialization, the system programmer can identify the devices which are to be used by JES2 via the READERnn, PRINTERnn and PUNCHnn parameters. The system programmer can also specify JES2 processing parameters to be associated with each device and indicate whether the device is to be considered active or inactive upon completion of JES2 initialization. An active device is dynamically allocated during JES2 initialization, and processing on that device begins as soon as work is available. An inactive device must be activated by the operator via the JES2 START command ($S).

If, during JES2 initialization, the system programmer does not identify as many devices as were specified during JES2 generation, JES2 selects devices and dynamically allocates them. Devices are selected according to lowest device address for each type of device (reader, printer, punch), until the number specified during JES2 generation is obtained or no devices of that type remain. For a device to be selected, it must be physically attached to the system. For devices not identified to JES2 during JES2 initialization, default parameters established during JES2 generation and initialization are used.

During JES2 processing, devices can be activated via the JES2 START command ($S) and deactivated via the JES2 STOP command ($P), resulting in dynamic allocation or deallocation of the device.

Internal Reader

During JES2 generation, the maximum number of jobstreams that can be simultaneously entered through the internal read facility is specified (&NUMINRS). To JES2 it appears that this number of devices is attached. Actually, there is one internal reader facility. Refer to "The Internal Reader Facility" in the section describing job submission.

During JES2 initialization, the internal reader facility is defined with the INTRDR parameter.

The maximum specified by the &NUMINRS parameter does not apply to the system internal readers associated with the time-sharing LOGONs (TSUINRDR) and started tasks (STCINRDR).

Remote Line and Device Configuration

Configuring RJE devices consists of defining both teleprocessing lines and remote station facilities. The remote facility can range from one remote terminal (2270,3780), to a remote workstation consisting of a computer operating many devices including (optionally) a remote console under the control of a JES2 remote terminal program and communicating with JES2 via the MULTI-LEAVING* technique.

---

*Trademark

Teleprocessing lines are either dedicated (permanently attached) to a single station, or nondedicated which implies that multiple stations can use the line. JES2 does not support multiple active remote stations on one line.

During JES2 generation, the system programmer:

- Specifies the maximum number of teleprocessing lines supported by JES2. &NUMLNES parameter.
- Specifies the maximum number of remote stations supported by JES2. &NUMRJE parameter.
- Specifies the maximum number of remote card readers, remote printers, and remote punches simultaneously supported by JES2. This includes devices that make up remote terminals and devices attached to remote work stations. &NUMTPRD, &NUMTPPR, &NUMTPPU parameters.
- Specifies inclusion of support in JES2 for any remote stations supported with the &BSCCPU, &BSC2770, &BSC2780, and &BSC3780 parameters.
- Generates the remote terminal programs by specifying RMTGEN parameters for each remote workstation.

During JES2 initialization, the system programmer can identify and specify characteristics for each line, remote station, and remote device with the LINEnnn, RMTnnn, Rnnn.RDm, Rnnn.PRm, and Rnnn.PUm parameters. A line is dedicated to a remote station by designating the line number in the RMTnnn parameter. Any line not designated by any RMTnnn parameter is a nondedicated line.

The remote station operator can control the remote station and the remote station devices, jobs submitted through the remote station, or data routed to it. This control can be effected through the remote console or the remote card reader. Each remote station is considered an extension of the local JES2 facility.

Spool Configuration

JES2 uses the SYS1.HASPACE data set on each volume identified as a spool volume to store all job input, job output, JES2 control blocks, and system data such as the job journal. Spool volumes are identified to JES2 by volume serial number. A six-character name identifying the primary spool volume is specified in the &SPOOL parameter during JES2 generation. An &SPOOL parameter can be used during JES2 initialization to override the JES2 generation parameter. The primary spool volume must exist during JES2 initialization.

Each volume with a volume serial number matching the first five characters of the &SPOOL parameter is considered a spool volume by JES2, and is searched for a SYS1.HASPACE data set. The maximum number of volumes that can be used as spool volumes is specified during JES2 generation with the &NUMDA parameter.

The system programmer also specifies the manner in which the tracks of the volumes are allocated and subdivided into physical records, by specifying the &NUMTGV and &BUFSIZE parameters. These parameters can be specified only during JES generation.

JES2 also requires one SYS1.HASPCKPT data set on a direct access volume to store a copy of the JES2 queue and other information needed for warm start. This data set may be on the primary spool volume or on another volume as specified by the &CHKPT parameter during JES2 initialization. See *OS/VS2 System Programming Library: System Generation Reference* for a description of how to allocate this data set and the SYS1.HASPACE data sets.

Starting or Stopping JES2

JES2 and the parameters that define its operation are selected during job entry subsystem initialization. Initialization of VS2 and of the job entry subsystem JES2 are two distinct processes. Basically this means that those processes associated with initialization (coldstart, warmstart) are specified separately for VS2 (reloading the LPA, clearing the VIO data sets) and for JES2 (initializing the job queues). Although for any given IPL it is possible to coldstart one (VS2 or JES2) and warmstart the other, the usual procedure is a warmstart for both. A detailed description of the process and options for an IPL generally, and for starting and stopping JES2 particularly, is located in the *OS/VS2 System Programming Library: Initialization and Tuning Guide*. The description given here is an overview with considerably less detail.

The starter system provided for installation of a VS2 system contains a JES2 procedure in SYS1.PROCLIB that automatically starts JES2. This first start requires that the operator respond with the COLD or FORMAT options to the message: SPECIFY OPTIONS. After the initial IPL, it may be better to provide the system with one or more initialization data sets, which can quickly specify initialization parameters suited to the conditions under which the system will run. Utility programs are used to place the initialization data set(s) in a library such as SYS1.PROCLIB, and the JES2 procedure is modified to reference them. The JES2 procedure provided with the starter system is shown in figure 19.

```
//JES2        PROC   MEMBER = JES2PARM
//IEFPROC     EXEC   PGM = HASJES20,DPRTY = (15,15),TIME = 1440
//PROC00      DD     DSN = SYS1.PROCLIB,DISP = SHR
//HASPPARM    DD     DSN = SYS1.PARMLIB( & MEMBER),DISP = SHR
//HASPLIST    DD     DDNAME = IEFRDER
```

Figure 19. JES2 Procedure provided with the Starter System

Different initialization data sets can be created to suit various operating conditions and workloads. Although the HASPPARM of figure 19 contains only comments (null data set) immediately after system generation, initialization parameters can be specified to configure batch, remote, or time sharing operations.

To change initialization parameters, the JES2 queues can be quiesced and then JES2 stopped. JES2 can then be restarted (by command) with different parameters and/or a different initialization data set. For an end-of-day halt, it would also be necessary to quiesce the VS2 system — a separate operation. A different initialization data set can then be specified as part of the next IPL.

The automatic starting feature can be removed by removing "//START prisubname" from the MSTRJCL. JES2 parameters can then be entered on an operator-issued START command that must be issued before MVS processing can occur.

Controlling Job Submission and Queuing

Jobs are submitted through the job entry subsystem and queued in priority order. The system programmer can use various parameters and facilities to control input streams, to control the specification of job classes and generation of priorities for jobs, to hold or release jobs, to set the default performance group for a job, and to change these specifications by changing entries in the JES2 job control table using the JES2 job statement accounting field scan exit.

Submitting Jobs

Jobs are submitted to JES2 in three ways:

- through card readers allocated to JES2.
- through RJE devices allocated to JES2.
- through a JES2 internal reader facility.

Local Device Submission

Local card readers can be supported via the JES2 automatic starting facility by specifying AUTO on the READERnn JES2 initialization parameter. Jobstreams can then be entered simply by readying the card reader. No other operator action is necessary. The card reader is deactivated and deallocated from JES2 with the JES2 stop ($P) command.

Remote Job Submission

The RJE support is described under separate sections (configuration; controlling remote job entry). It should be noted that JES2 processes remote jobs no differently from those received from local card readers or the internal reader facility.

The Internal Reader Facility

An internal reader jobstream is identified to JES2 by the fact that an output data set specifying a special user writer (INTRDR) has been allocated dynamically, or via SYSOUT=(x,INTRDR) coded on a DD card. JES2 recognizes such data sets and places them in the input stream, thus allowing jobs and system tasks to enter jobs in the input stream.

A job entered through an internal reader is delimited beginning with a //JOB statement and ending with the next //JOB statement, a /*EOF statement, or the closing of the internal reader data set. Abnormal closing or closing after a WRITE error causes deletion of the last job. A /*DEL statement may be used to explicitly delete the last job.

The class to which the Internal Reader data set is allocated, e.g. class X if SYSOUT=(X,INTRDR), becomes the MSGCLASS for the submitted job unless the JOB statement contains a MSGCLASS parameter. If the Internal Reader data set is dynamically allocated without a class specified, the MSGCLASS of the submitting job or TSO user becomes the default. Two exceptions to this are time sharing LOGONS and started tasks. These are assigned the TSUMCLASS and STCMCLASS JES2 Initialization parameter values. The DEST parameter, if specified for the Internal Reader allocation, becomes the default print data destination for all jobs submitted via that Internal Reader.

JES2 provides the capability of receiving multiple jobs simultaneously via the internal reader facility. The system uses it to pass started tasks, TSO logon, and TSO background jobs to JES2. Also, jobstreams can be read from tape and disk (any QSAM-supported device) and submitted through the internal reader via the RDR procedure, and any job executing in MVS can use the internal reader facility to pass a jobstream to JES2.

Controlling the Internal Reader Facility Although the internal reader facility appears to JES2 logically as multiple input devices (maximum specified the &NUMINRS parameter during JES2 generation), the facility is controlled as one entity. The number (&NUMINRS) of internal readers is the number of jobs that can be received simultaneously through this facility.

Characteristics of the facility are specified during JES2 initialization as subparameters of the INTRDR parameter.

Using The RDR Procedure The procedure supplied by IBM for using the Internal Reader facility to read jobs from tape or disk is named RDR. The starter system provides the RDR procedure in SYS1.PROCLIB to allow the operator to start JES2 generation (see figure 20). Basically the same procedure can be used to read a jobstream from any QSAM-supported device. The operator uses the RDR procedure as follows:

```
//IEFPROC     EXEC   PGM = IEBEDIT
//SYSUT1      DD     DDNAME = IEFRDER
//IEFRDER     DD     DSN = NULLFILE,DISP = OLD
//SYSUT2      DD     SYSOUT = (A,INTRDR)
//SYSPRINT    DD     SYSOUT = A
//SYSIN       DD     DUMMY
```

Figure 20. The RDR Procedure

- To read a jobstream from the second file of a tape named JOBTAP on device 180:
  ```
  START RDR,180,JOBTAP,LABEL=2,DSN=JOBS
  ```
- To read a jobstream from a cataloged library of jobs:
  ```
  START RDR,3330,DSN=PRODUCTN(PAYROLL)
  ```
- To read a jobstream starting with a specific job on a tape named JOBTAP, the operator must submit a job to JES2:

```
//READJOBx  JOB     ......
//          EXEC    RDR
//IEFRDER   DD      DSN=JOBS,VOL=SER=JOBTAP,
//                  UNIT=3400,DISP=OLD
//SYSIN     DD      *
          EDIT    START=JOBx
/$
```

The system programmer can define internal readers on EXEC statements in such a manner that they are started conditionally. This allows the formation of a set of dependent jobs that can execute without operator intervention. For example:

- To submit Jobs B and C if the first four steps of Job A complete successfully.

```
//JOBA      JOB     ...
//STEP1     EXEC    ...
            -
            -
            -
            -
//STEP5     EXEC    RDR,COND=(8,LE)
//IEFRDER   DD      DSN=JOBS(JOBB),DISP=SHR
//          DD      DSN=JOBS(JOBC),DISP=SHR
```

- To submit Job B if Job A terminates normally, Job C if it terminates abnormally, and Job D in either case.

```
//JOBA      JOB     .....
//STEP1     EXEC    ....
            -
            -
            -
//STEPN     EXEC    RDR
//IEFRDER   DD      DSN=JOBS(JOBB),DISP=SHR
//STEPN1    EXEC    RDR,COND=ONLY
//IEFRDER   DD      DSN=JOBS(JOBC),DISP=SHR
//STEPN2    EXEC    RDR,COND=EVEN
//IEFRDER   DD      DSN=JOBS(JOBD),DISP=SHR
```

Installation-written procedures and programs can further exploit the internal reader facility to select particular jobs, to generate special job streams, and to allow operator submission of production jobstreams.

Controlling Job Enqueuing

In JES2, jobs are enqueued by priority sequence and, when ready for execution, within individual classes. The system programmer can control job selection through determining job class and priority.

The JES2 Queue

A job received by JES2 is enqueued in job priority order on the JES2 queue residing in the JES2 address space in main storage. A job is considered received by JES2 when it has been totally read in and the JES2 control blocks placed on the spool.

The queue entry for each job contains the job name, job priority, a flag to indicate the job is held, pointers to JES2 control blocks on the spool, and the JES2 process (JCL conversion, execution, output processing, purge) for which the job is next eligible. Jobs are selected in priority order for each JES2 process. Logically, the one JES2 job queue has queues for each process, plus 38 (one for each class) within the execution process.

A job which is held is not removed from the queue; instead it is made ineligible to be selected for any JES2 processing. A job can be held at any time. Thus a job in execution that is held by the operator, is not eligible for output processing until released. The holding of a job that is read for execution can occur by job, by class, or by holding all jobs.

Job Class

There are 38 classes of jobs possible under JES2. Two are used by the system: STC for started task control, TSU for time sharing logon. The other 36 classes, A-Z and 0-9, are for normal jobs and can be used to help control the job mix.

The job class is specified on the JOB statement (CLASS=jobclass). If not specified, a default based on the particular device through which the job is entered into JES2 is assigned. All jobs entered through the internal reader facility are considered to be entered through a device described by the INTRDR parameter.

There are no absolute rules for assigning job classes, and some experimentation is necessary. Generally, jobs of similar characteristics and specifying identical processing requirements should be assigned to the same class. For example, if several jobs are time-dependent and execute in nonpageable dynamic storage, it may not be desirable to tie up all of nonpageable dynamic storage by having these jobs running concurrently. These jobs may all be assigned to class B (or C or D — class names have no inherent meaning); then, if only one initiator is started that can handle class B jobs, there will never be more than one of these jobs executing at once.

Suppose the following assignments are made:

Class B = jobs that are time-dependent.
Class C = jobs with high CPU requirements.
Class D = jobs with high I/O requirements.

The system programmer can specify initiator parameters such as:

```
I1        CLASS=BCD
I2        CLASS=CDB
I3        CLASS=DCB
```

If the three initiators are processing jobs with the same priority and all necessary resources (for example, I/O devices and data sets) are available, then three jobs, one from each of the three different classes, run concurrently. If a job within one of the classes has higher priority than the others in the class, it will be initiated first.

During JES2 initialization, the system programmer can assign job characteristics to jobs enqueued in each class. Characteristics that can be assigned are:

- A default performance group for each job.
- JCL conversion parameters.
- Whether a JES2 job log is to be produced for jobs in this class. The JES2 job log is a list of all messages and replies issued by, or on behalf of, a job.
- Whether a system journal is to be saved for this job. If it is not saved, the overhead is avoided but the job may not be automatically restarted in case of job failure or system restart.
- Whether this class is reserved for the execution batch scheduling facility. (See Execution Batch Scheduling.)
- Whether output is suppressed for jobs in this class (e.g., started tasks).
- Define the procedure library (PROCnn).
- SMF options.
- Whether the job is held.
- Whether the job is to be simply copied to message class output or converted but not executed.

The system programmer should assign separate job classes to jobs that are to be assigned separate characteristics, and to jobs that have different execution characteristics such as:

- rate of CPU to I/O processing
- use of special devices
- number of devices used
- use of real storage

JES2 Job Scheduling Priority

Job priority is determined through use of the PRIORITY statement, or by an algorithm that uses programmer-supplied or generation-defaulted job characteristic data.

In addition an increment can be added to the priority and a priority limit enforced, depending on the device through which the job entered JES2. Those parameters are associated with the input device during JES2 initialization.

Specifying Priority

Priority is specified on the /*PRIORITY statement. If specified, it must immediately precede the JOB statement, or the input stream is flushed until another JOB or PRIORITY statement is found.

Calculating Priority

When the scheduling priority is not specified on the /*PRIORITY statement, priority is calculated using estimated execution time and the estimated number of output lines and cards. These values can be entered on the JOBPARM statement, in the accounting information on the JOB card, or can be entered or defaulted with JES2 generation parameters ($ESTIME, $ESTLNCT, $ESTPUN).

To calculate priority, these estimates are used in conjunction with four JES2 generation parameters, each of which is supplied (or defaulted) as a table of values. These are the &XLIN(m), &RPRT(n), &RPRI(n), &XPRI(m) parameters. The default values for these tables are used for the examples of priority calculation that follow. The default values are:

For &XLIN:
(&XLIN values are estimates of the number of output lines and cards for a job.)

&XLIN(m)	m
2000	1
5000	2
15000	3
$2^{24}-1$	4
⋮	⋮
$2^{24}-1$	9

For &RPRT:
(&RPRT values are estimates of the number of minutes a job will take to run.)

&RPRT(n)	n
2	1
5	2
15	3
$2^{24}-1$	4
⋮	⋮
$2^{24}-1$	9

For both &XPRI and &RPRI:
(The m and n values are determined from the &XLIN and &RPRT tables.) Note that the &XPRI and &RPRI tables can have two different sets of values. The values described here are the default values.

m or n	&XPRI(m) or &RPRI(n)
1	9
2	8
3	7
⋮	⋮
9	1

Priority is calculated by using the values specified for &XLIN and &RPRT to determine m and n from the table. These m and n values are then used with the &XPRI and &RPRI tables to determine values for &XPRI(m) and &RPRI(n). Priority is then calculated as:

```
PRIORITY=[&XPRI(m)+&RPRI(n)]/2
```

The following examples illustrate the use of the various parameters in this calculation.

Example 1. The programmer estimates 2000 lines plus cards, 2 minutes execution time.

```
From &XLIN, 2000 lines implies m=1
From &RPRT, 2 minutes implies n=1
From &XPRI, m=1 implies &XPRI=9
From &RPRI, n=1 implies &RPRI=9
```

Therefore PRIORITY=(9+9)/2=9

Example 2. The programmer estimates 4000 lines plus cards, 15 minutes execution time.

```
From &XLIN, 4000 lines implies m=2
From &RPRT, 15 minutes implies n=3
From &XPRI, m=2 implies &XPRI=8
From &RPRI, n=3 implies &RPRI=7
```

Therefore PRIORITY=(8+7)/2=7

(The fraction is ignored; only the integer value is used.)

Example 3. The programmer estimates 15,000 lines plus cards, 10 minutes execution time.

```
From &XLIN, 15,000 lines implies m=3
From &RPRT, 10 minutes implies n=3
From &XPRI, m=3 implies &XPRI=7
From &RPRI, n=3 implies &RPRI=7
```

Therefore PRIORITY=(7+7)/2=7

If priority is computed for the job during input, it is recomputed for output. Output priority is based on the count of lines and cards (&XLIN) actually produced during job execution. The execution time parameter &RPRT is ignored.

The system programmer, by specifying other values for the tables during JES2 generation, can more closely control priority specification. Values specified on the JOBPARM statement supercede those in the account field of the JOB statement. During JES2 generation the &RJOBOPT parameter can be specified to ignore the account field on the JOB card.

Priority Aging

The priority of a job can be increased as a function of the length of time that it has been in the system. The &PRIHIGH, &PRILOW, and &PRIRATE JES2 generation parameters specify respectively a limit above which there is no incrementing, a limit below which there is no incrementing, and an integer representing the number of times that the priority is incremented in a 24-hour period — subject to the upper limit. The default of zero for the &PRIRATE parameter specifies no priority aging.

The PRIRATE parameter specifies whether the feature is used and, if so, how many times in a 24 hour period the priority is incremented. For example, PRIRATE=48 specifies a priority increment of one unit every 30 minutes. The PRIHIGH parameter specifies the upper limit; a priority lower than the value of the PRILOW parameter specifies that the job is not subject to priority aging.

The JOB Statement Accounting Field Scan

During JES2 generation the &RJOBOPT parameter is specified to determine whether JES2 is to scan the account field of the JOB statement, and to set the conditions under which JCL scan errors cause job termination prior to JCL conversion. Figure 21 describes the &RJOBOPT options.

The account field is considered valid if its format is that specified for HASP II. The format is assumed to be as follows:

```
(pano,room,time,lines,cards,forms,copies,log,linect)
```

where:

pano
: programmer's accounting number. One to four alphameric characters.

room
: programmer's room number. One to four alphameric characters.

time
: time estimated execution time in minutes. Up to four numeric digits (example: ",30" for 30 minutes). If omitted, a standard value is assumed.

lines
estimated line count in thousands of lines. Up to four numeric digits (example: ",5" for 500 lines). If omitted, a standard number of lines is assumed.

cards
estimated number of cards to be punched. Up to four numeric digits. If omitted, a standard number of cards is assumed.

forms
special forms for printing entire job. From one to four alphameric characters (example: ",5" for 5-part forms). If omitted, standard forms are assumed.

copies
number of times output is to be printed or punched. Up to three numeric digits not exceeding an installation-specified limit (maximum 255) (example: ",2" for two copies). If omitted, one copy is assumed.

log
Job Log option. This subfield should consist of one character. If this character is an "N", the HASP Job Log is not produced. If any other other character, or omitted, the log is produced.

linect
to be printed per page. Up to three numeric digits not exceeding 255. If coded as zero, no automatic overflow is produced. If omitted, a standard value is assumed.

The JCL scan is not exhaustive; only JOB, DD *, and DD DATA statements are scanned. Job termination on a JCL error at this point does not guarantee that all JCL errors have been found. If the job is not terminated on a JCL error at this point in the process, it can still fail during JCL conversion when all JCL is scanned.

&RJOBOPT	Scan Account Field	Terminate if Account Field Invalid	Terminate if JCL Invalid
0	Yes	Yes	Yes
1	Yes	Yes	No
2	Yes	No	Yes
3	Yes	No	No
4	No	-	Yes
5	No	-	No

Figure 21. Job Statement Accounting Field Scan Exit

Job Statement Accounting Field Scan Exit A routine can be written that allows the installation to control a job by modifying data in the JES2 Job Control Table (JCT) immediately following the scan of the user's JOB statement. (Figure 22 shows selected fields from the JCT.)

The name of the CSECT must be HASPRSCN and must replace the existing CSECT of that name in the HASJES20 load module. This routine receives control from JES2 with registers set as follows:

R0	A binary number giving the length (in bytes) of the accounting field
R1	Address of the accounting field from the JOB card
R2	Address of the SMF job management record (as defined in the description of the VS2 Common Exit Parameter Area, *OS/VS System Management Facilities*, GC35-0004.
R8	Addressability
R10	Address of the (JES2) JCT
R13	Save area (18 words)
R14	Return address

Registers 3-14 must be restored when control is returned to JES2. The save area addressed by R13 can be used to store registers.

There are no return codes necessary.

Programming Notes: Parameters of the JOBPARM statement or ROUTE statement can override changes made with this routine. Otherwise the change is effective for the duration of the job.

Data placed in the user identification field of the SMF record at this time, is available to the programmer at all SMF exits.

If system services that have implied WAITs (e.g., WTO, the SMF WTM) are used by this routine, severe system degradation may occur.

Since this routine runs under the JES2 task, if abnormal termination occurs the system must be restarted.

Displacement (hex) in JCT	Length (bytes)	Notes		Field
8D	1		SMF Flags:	
		-	bit 0-1	Reserved
		1,2	2	If set, IEFUSO exit not taken
		-	3-4	Reserved
		1,2	5	If set, no Type 6 SMF records produced
		1,2	6	If set, IEFUJP exit not taken
		1,2	7	If set, no Type 26 SMF record produced
8E	1		Job Flags:	
		-	bit 0	Background job
		-	1	TSO (foreground) job
		-	2	Started task
		1,2	3	No job journalling
		1,2	4	No output
		1,2,3	5	TYPRUN=SCAN
		2,3	6	TYPRUN=COPY
		-	7	Reserved
8F	1		Job Options:	
		-	bit 0	/*PRIORITY card was read, value is in Priority field (B6)
		-	1	/*SETUP card was read
		1,2,4	2	TYPRUN=HOLD was specified
		1,2,6,8	3	No job log for this job
		1,2	4	Execution batch job
		-	5	The job was read through an Internal Reader
		-	6-7	Reserved
90	8	-	JES2 JOB identifier	
98	8	3	Job name	
A0	20	3	Programmer name	
B4	1	1,4	Message class	
B5	1	1,4	Job class	
B6	1	1,5	Priority	
.				
.				
BA	2	-	Route code of input device	
BC	8	-	Input device name	
C4	4	1,6	Account number (for HASP compatability)	
C8	4	1,6,8	Room number	
CC	4	1,6,8	Estimated real time job will run	
D0	4	1,6,8	Estimated count of output lines (in thousands)	
D4	4	1,6,8	Estimated number of output cards punched	
D8	4	1,6,8	Job Forms	
DD	1	1,6,8	Job copy count (binary)	
DF	1	1,6,8	Lines per page (binary)	
E0	2	1,7	Default print routing (binary)	
E2	2	1,7	Default punch routing (binary)	
			0	Any local device
			1-999	Remote devices
			1001-1999	Specific local devices
EC	8	1,2,8	Procedure DD name	

Notes:

1. Can be modified by installation routine.
2. Preset from $X Initialization Parameter according to job class.
3. Preset from JOB Statement.
4. From JOB Statement, if specified; otherwise according to input device as established at JES2 Initialization (e.g. in READERn).
5. Preset from /*PRIORITY Statement, or an "*".
6. The HASPRSCN routine is used by JES2 to scan the account field of the JOB statement. If the HASPRSCN routine is replaced by an installation-written routine, the account field is empty.
7. Preset according to an input device initialization parameter (e.g. READERn). If not set at Initialization the parameter defaults to the job input source value (LOCAL or RMTnnn). Can be modified by a ROUTE statement after the scan exit.
8. Can be modified by a JOBPARM statement after the scan exit.

Figure 22. Selected JES2 Job Control Table Fields

Controlling Conversion and Execution

The JCL for a job, logon, or started task is passed through the converter and converted into internal text. The job is then available for execution, which occurs as soon as an initiator eligible to process the job is available.

JCL Conversion

A job is eligible for JCL conversion as soon as it is placed on the queue. The converter is invoked separately for each job. JES2 passes to the converter, the converter parameters and a pointer to a catalog procedure library.

Converter Parameters

If not defaulted (&RDROPSU, &RDROPST, &RDROPSL JES2 generation parameters), the converter parameters are specified for each class on the CONVPARM subparameter of the &STC, &TSU, or &X parameters during JES initialization. Converter parameters specify defaults such as execution time estimate, and JCL and allocation MSGLEVEL options. Command disposition and authority and the bypass label options are specified. The specific parameters are described in the *OS/VS2 System Programming Library: Initialization and Tuning Guide.*

Procedure Library Selection

The JES2 Procedure is located in SYS1.PROCLIB. It defines job-related procedure libraries such as:

```
//PROC00    DD      ...
//PROC01    DD      ...
     .
     .
     .
//PROCnn    DD      ...
//anyname   DD      ...
```

The programmer can specify any of the libraries included in the JES2 procedure on the JOBPARM statement by specifying the library DDNAME. If multiple data sets are required they must be specified as concatenations in the JES2 Cataloged Procedure.

If there is no procedure specification on the JOBPARM statement, class-related initialization parameters can specify the library as PROCnn. If the procedure is not specified or specified and not found, PROC00 is used.

Execution Control

Execution is controlled through controlling the initiators and the jobs on the queue (see Controlling Job Enqueuing), as well as by monitoring the job and issuing commands.

JES2 associates one logical initiator residing in JES2, with each system initiator interfacing with JES2. The maximum number of logical initiators is specified during JES2 generation (&MAXPART parameter). The number of active logical initiators, subject to the maximum, is controlled by the operator ($S Inn). The operator can also associate with logical initiators the order in which the classes are selected by JES2.

Classes are associated with each initiator during JES2 initialization or dynamically by the operator. During execution, the initiator selects non-held jobs in priority order within their class, and the non-held class in the order specified for that initiator. That is, the lowest priority job in the first non-empty class is selected ahead of the highest priority job of the next class — assuming neither job nor class is held.

The Initiator Cataloged Procedure

One initiator cataloged procedure (INIT) must be contained in SYS1.PROCLIB for use by JES2 in creating job address spaces into which a system initiator is initialized. JES2 uses the START command (system command) to create one system initiator for each active JES2 logical initiator. The number of active initiators must be controlled by starting and stopping JES2 logical initiators.

The standard initiator cataloged procedure supplied by IBM is named INIT. The procedure is:

```
//IEFPROC    EXEC     PGM=IEFIIC,DPRTY=12
```

Job Monitoring

A job can be monitored by elapsed (wall clock) time, execution time, and by output in terms of lines and cards.

During JES2 generation, the &TIMEOPT parameter can be specified to cause JES2 to write a message to the operator when the elapsed time specified on the JOBPARM statement is exceeded, and an additional message at each interval specified by the &TIMEXS parameter. The system programmer can use the SMF accounting exit to enforce these values, if the time was placed in the SMF userid field during the JCL scan exit. The SMF exits are described in *OS/VS2 System Management Facilities (SMF)*, GC35-0004.

Execution time can be specified on either the EXEC statement or the JOB statement, or in the converter (CONVPARM) parameters. If the time is exceeded, the SMF exit is entered and the job can be cancelled or continued.

The &OUTXS JES2 generation parameter is used to specified the total number of printed lines and punched cards that a job can print before action is taken by JES2. The &OUTPOPT parameter specifies the action that is taken. The job can be allowed to continue after a message is written to the operator, or the job can be cancelled with or without a dump.

The installation can specify SMF output limiting by class, with the & x, &STC, and &TSU JES2 generation parameters. Output (OUTLIM DD) can be monitored for each data set by SMF.

Entering Commands in the Jobstream

JES2 commands and standard system commands are accepted at different points in a jobstream, with different types of control. A jobstream is defined as the set of jobs submitted between the physical start of a reader and physical end-of-file, or between the opening and closing of an internal reader data set. Refer to figure 23 for a pictorial representation of the following paragraph.

JES2 commands are accepted in the jobstream only if they are in front of the first //JOB statement of a jobstream. The commands that are accepted from any given device are controlled by a command authority associated with the device ($T command). The command authority associated allows various combinations of display and system, job, or device control commands to be entered. JES2 commands are in the form /*$command.

System commands in the form /*$VS 'systemcommand' are accepted in front of the JOB statement, subject to the same authority described for JES2 commands.

JES2 commands found in the jobstream between the first JOB statement and EOF are ignored.

System commands (//system cmd) which appear in the jobstream after the first Job statement, are executed in the converter. Whether they are issued is subject to control by the converter via the converter parameters. System commands appearing before the first JOB statement are ignored.

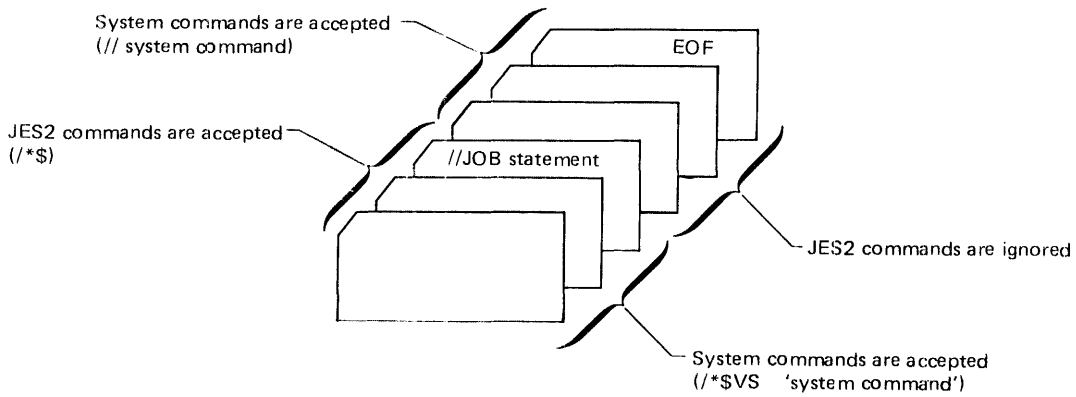

Figure 23. Entering Commands in the Jobstream

Execution Batch Scheduling

Execution batch scheduling is an extension of normal job scheduling that may provide improved system performance. It is the process of gathering psuedo-jobs, called execution batch jobs, into a single input stream for processing by an execution batch processing program. The execution batch jobs are submitted to JES2 one at a time; they may have different input sources, and different print and punch output routing. Execution batch scheduling collects these numerous related batch jobs into a single data stream and passes them as a SYSIN data set to the user-written execution batch processing program. This reduces the overhead associated with setting up for, and processing, numerous individual jobs and/or job steps. Another advantage is that individual accounting for all but type 4 and 5 records is available.

The processing programs to be used with the execution batch scheduling feature may cover a wide variety of application areas such as:

- Compile-and-go debugging compilers.
- File inquiry programs.
- Hardware or software system emulators.

It is desirable that the program process jobs or transactions of relatively short duration. If not, the saving in job management overhead between successive jobs may not be a large enough percentage of total job execution time to justify use of this feature.

At JES2 initialization, the installation defines the job class or classes that are to be dedicated to execution batch scheduling. One class or group of classes is assigned to each type of execution batch processing program. Subsequently, the batch user identifies the program requested by the class stated on the JOB statement.

JES2 can support more than one execution batch processing program to process various kinds of batch jobs. Each exectuion batch processing program must be associated with at least one JES2 initiator. The system initiators request a job, any job, and JES2 decides which job is to be processed.

To determine which jobs are to be processed by an execution batch processing program, JES2 recognizes jobs assigned to eligible classes. Instead of sending these jobs directly to an initiator, JES2 invokes an appropriate procedure from PROCLIB to initiate the execution batch processing program. The job as submitted is now considered part of the input data of the execution batch processing program.

For example, consider an order entry system that requires an inventory update and an invoice for each order. With standard processing, the normal procedure would be to batch all orders and submit them as a data stream at the end of the day to an order entry system.

However, this causes delay. Alternatively, the installation can periodically batch together orders received during a certain time period and run the job several times a day. By using the execution batch scheduling facility, orders can be processed as if the order processing program were scheduled for every order, but without the overhead of scheduling the order program for the runs.

The order entry program would become an execution batch processing program. The orders themselves would be submitted as execution batch jobs by taking the order data that would have been submitted in batch, and putting a system JOB statement in front of it. In an order entry program accustomed to reading batch jobs at the end of the day, the only programming changes would be (1) to use the ddname SYSIN for the input stream (this may be accomplished by JCL in PROCLIB), (2) to recognize the null statement as an order separator or establish other defined terminators, and (3) ignore all other JCL (//) cards. The program would have to print all related information for each order before processing the next order, to distinguish one from another. JES2 automatically schedules the order entry program when it is needed and concatenates all orders into the input stream data regardless of where they have originated.

Submitting Input to an Execution Batch Processing Program

A representative input stream follows:

```
        input
     JES2 control statements
//   JOB
```

To submit data to an execution batch processing program, follow these rules:

- The first statement of each job must be a standard JOB statement that includes a CLASS=jobclass parameter. The jobclass identifies which program is to receive the input. The installation associates jobclasses with an execution batch facility via the procedure library. It associates jobclasses with initiators at initialization. The accounting field is interpreted by JES2 just as it is for normal jobs.
- All JES2 control statements are effective with batching jobs except /*OUTPUT, which is ignored.
- No other JCL is used. All other statements are input to the execution batch processing program. These can be read just as if they had been placed in a DD DATA data set and the execution batch program has been invoked by standard JCL. If the execution batch program requires it, each transaction can be terminated by a statement with $$ in columns 1 and 2.

In the order entry system example mentioned earlier, code the following:

```
//JOBxx          JOB          (INVO1,667),CLASS=X
/*ROUTE          PRINT        RMT47
order 1
order 2
```

The /*ROUTE statement will cause the invoice to be printed at the remote location.

Execution Batch Scheduling Operations

Special actions take place when JES2 recognizes input for an execution batch program.

If the execution batch program is not already active, JES2 submits an internal job which uses JCL from SYS1.PROCLIB to invoke the execution batch processing program when an initiator capable of processing it becomes available. JES2 control cards are converted to JCL comment statements. The entire input, plus a JCL null statement added by JES2, is allocated to the execution batch processing program as an input data set with the ddname of SYSIN.

If the execution batch program is already active and simply waiting for another job, JES2 makes the input data set allocation as above, and processing begins immediately without any use of job management.

The end of input can be detected by the execution batch program when it reads the JCL null statement added by JES2. After writing any remaining SYSOUT data for the completed job, the execution batch program attempts to read ahead in its input file for another transaction. JES2 detects this condition, temporarily forces the execution batch program into a wait state, and performs job termination actions for the execution batch job (flushes output buffers, releases input spool space, queues the job for printing, and so forth). The execution batch program remains active in the MVS address space.

When an execution batch program is waiting, JES2 job selection is altered. Instead of scanning for all classes eligible to execute in that address space, JES2 first tries to start an execution batch job which may be processed by that execution batch program. If sucessful, processing can begin immediately.

If no jobs of the same execution batch class are available to execute, all other job classes of the address space are scanned in order. If a job is found, JES2 internally cancels the execution batch processing program and normal scheduling, using job management, takes place.

If no jobs of the other classes are found, the address space and execution batch processig program remain idle, awaiting availability of a job in any of its classes. If a job becomes available in the class of the execution batch program still in the address space, processing begins immediately.

If an execution batch program ends (ABEND or normal return to VS), JES2 detects this as a nonbatch termination in the address space. Job management will be used to reinvoke the batch program when another job for its class is selected.

Use of the operator commands $P I or $P In will cause JES2 to cancel an execution batch processing program when it becomes idle, and then delete the address space.

In summary, an execution batch processing program must have certain characteristics:

- It must read all user input from a single sequential data set.
- It must recognize a standard OS JOB statement, or its own control statement, to determine the beginning of a job.
- It must recognize a standard OS null JCL statement (// followed by 78 blanks), or its own control statement, to determine the end of a job.

The execution batch processing program will receive an end-of-file condition when a card with $$ in columns 1 and 2 is read while processing a job. The program may continue to the next logical subfile by simply resetting appropriate bits in I/O control blocks and continuing reading, or by closing and reopening the data set to continue reading at the card following the $$ card.

Preparing for Execution Batch Scheduling

The batching feature is included in JES2 by setting the &XBATCH=YES parameter during JES2 generation. Job classes are reserved for execution batch jobs with the $$x initialization parameter. The &XBATCHN JES2 generation parameter may be set to define the first five characters of the catalog procedure name that contains the JCL necessary to execute an execution batch program. (See the *OS/VS2 System Programming Library: System Generation Reference* for a description of this parameter.)

Each batch class should be associated with one execution batch program. Each batch class should be made eligible to execute in the MVS address space by setting the Inn initialization parameter or by using the $T In operator command.

For each combination of batch class and initiator, there must be a procedure in SYS1.PROCLIB named "nnnnncid", where

- nnnnn are the five characters assigned to &XBATCHN.
- c is the particular batch job class set in $$x.
- id is the 1- or 2-character initiator identification, corresponding to nn of the Inn parameter

These procedures actually call the execution batch program for each class, and define all data sets other than the user input data set.

The procedures may be single step, or may have preliminary steps before the single step that invokes the execution batch program (stepname GO). The execution batch program invoked by this step must read its input from a SYSIN, or the procedure must refer to DDNAME=SYSIN on a DD statement used for input by the processing program.

If a given batch class is eligible (the Inn initialization parameter or $T In operator command defines eligible classes) to be executed by more than one initiator, the requirement for a separate procedure name for each address space/class combination may be satisfied by alias names of a single procedure, or by actual separate procedures which can specify different work fields.

The following example shows the internal job that JES2 generates to initially load a program to process batch class X jobs for Init=3, assuming the default setting for &XBATCHN.

```
//$$$$$X3       JOB     1,SYS,MSGLEVEL=1
//FAKE          EXEC    $$$$$X3
//GO.SYSIN      DD      DATA,DCB=BUFNO=1
//
```

The following is an example of a procedure that an installation might use for a simple file inquiry program that reads inquiry input from SYSIN, checks a file, and prints responses to SYSPRINT.

```
//$$$$$X3       PROC
//GO            EXEC        PGM=FINDPART
//SYSPRINT      DD          SYSOUT=A
//PARTFILE      DD          DSN=PARTFILE.MASTER,DISP=SHR
//SYSUDUMP      DD          SYSOUT=A
```

The following JCL is for the order entry system example.

```
//$$$$$X3       PROC
//MDSE          EXEC        PGM=ORDERIN
//MESSAGE       DD          SYSOUT=M
//INVOICE       DD          SYSOUT=(P,,INVC)
//INVTRY        DD          DSN=MSTRINVT,DISP=SHR
//ORDERS        DD          DSN=ORDERS,DISP=MOD
```

- //MESSAGE — the installation might identify class M as a punch class. This will allow the submitter of the execution batch job to route the invoices and messages separately, as shown in the example in "Submitting Input to an Execution Batch Processing Program".
- //INVOICE — defines the specially prepared output.
- //INVTRY — uses a master inventory list as a base; it is updated as the orders are received.
- //ORDERS — accumulates the day's orders. ORDERS has a disposition of MOD because the execution batch processing program is periodically started and stopped during the day.
- SYSOUT data sets — the messages and invoices.
- SYSIN data sets — the DD DATA input is every execution batch job that is processed by the execution batch processing program.

Controlling System Output

JES2 provides:

- Queueing levels beyond the simple output class queueing provided by the output writer.
- The ability to specify print train and either carriage tape name or forms control buffers for sysout directed to 3211 and 1403 printers plus support of the 3525 print and interpret features for sysout data sets.
- Features that minimize operator interaction due to forms, carriage tape, and print train loading.
- An external writer facility that, although possible to use for writing any sysout data, is specifically intended for writing to devices other than printers and punches and for controlling all output written by installation-written writers.

Queuing Output

The job output elements (JOE)s are created during output processing, or during execution in the case of spinoff, by JES2. Each JOE represents a unit of work to JES2, and is placed in a job output table (JOT) in order of output priority. If the priority was calculated originally, it has now been recalculated with the actual number of lines and cards. See "Calculating Priority".

The JES(S2 writer and the external writer can select only data sets for which JOES have been constructed. Varying the number of JOEs, (&NUMJOES JES2 generation parameter) influences the way output is processed. By specifying a large number of JOEs the output processors are given a large number of output data sets from which to choose. This minimizes the setup changes in JES2-controlled printers and punches by providing a series of data sets of the same class for the external writers. However, a given data set may wait a long time for a printer with the specified setup, an available device destination, or for an available external writer to dequeue its class. This long wait may fill spool space, since most of a job's output-related spool space is freed only when all output data sets have been processed. Specifying many JOEs tends to optimize output device utilization at the expense of throughput for a specific job.

Specifying few JOEs tends to reduce the number of jobs with output eligible for printing while processing the entire job output more nearly together. This specification may minimize the turnaround of a particular job at the expense of operational efficiency.

A job output element that does not yet describe a unit of work is said to be "free". The $MINJOE parameter specifies the number of JOEs that must be left free to be used when the $I command interrupts an output data set or when a printer is started. When the building of JOEs for a job would drop the number available below the specified minimum, the job or spinoff data is forced to wait until JOES are available.

JES2 queues output data by combinations of data set characteristics such as output class, forms, print train and forms control buffer name. (Data sets are also queued by installation writer name and destination, as discussed in the External Writer section.) These characteristics are obtained from the SYSOUT DD statement or the JES2 OUTPUT statement. With the exception of held data sets and spinoff data sets, a job's, started task's, or time-sharing user's output that has identical characteristics is queued together in a data set group pointed to by a job output element (JOE). (This queueing can be altered by the demand setup option.) Each held and spinoff data set is queued separately and constitutes a "group" of one data set. Each data set group is considered a processing entity with a set of processing characteristics. JES2 selects work by data set groups and will, if the separator option is specified, delimit each group with separator pages or cards.

Figure 24 represents how one JOE can represent one of serveral sysout data sets.

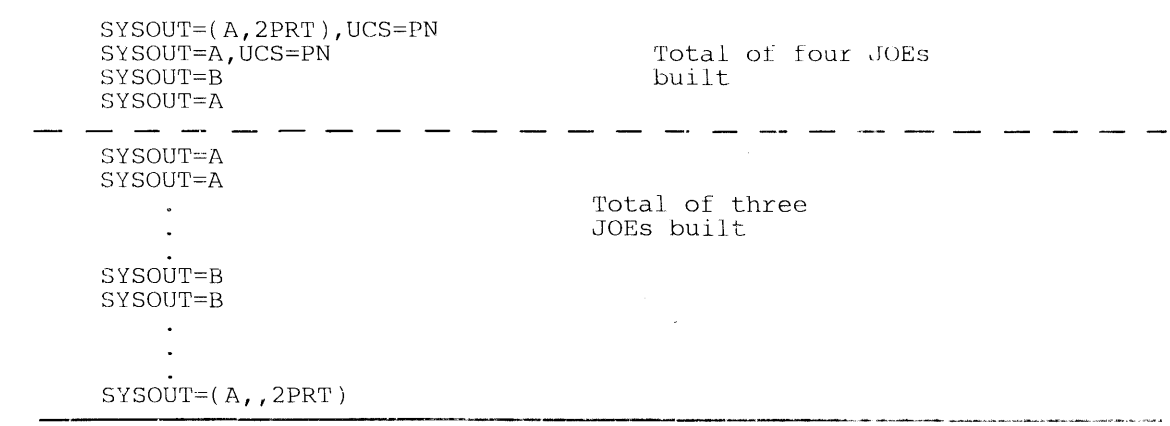

Figure 24. Relationship of SYSOUT Specification to Number of Job Output Elements

JOEs built for job-related output are duplicated according to the number of job copies requested on the /*JOBPARM statement. This allows the number of copies being processed for any job to be governed by the devices available for output.

Output Class Assignment

Output from a problem program is assigned to an output class which is processed by JES2. A maximum of 36 sysout classes can be named by specifying SYSOUT=x on the DD statement, where x is any single letter (A-Z) or digit (0-9). The names have no inherent meaning; they are simply used to group output of similar characteristics. During JES2 initialization, the fact that a class is designated as containing print or punch data is used for output limiting and job accounting purposes only and has no bearing on the actual device to which the class is assigned.

JES2 writers and external writers are assigned to process only designated classes of output. These classes can initially be assigned to JES2 writers during JES initialization, to external writers in the external writer cataloged procedure, or the operator can assign them to either as parameters of the START ($S) or MODIFY ($T) commands.

If output is assigned to a class for which no writer is started, it remains indefinitely on the queue.

System Message Classes

System messages generated during the execution of a program must also be routed to an output device; if messages must appear with their program output, they should be assigned to the same message class as the output. But to guarantee that the messages and data appear together, all data sets for the job must be described to JES2 in a single job output element (JOE).

It is also possible for an installation to specify ($DMNDEST) parameter) during JES2 generation, that all job output of the same class and for the same destination as its system messages (MSGCLASS), JCL statement images, and job log (if any), be placed in a single JOE. This keeps the data sets together on output listings but can cause operational inefficiencies.

The message class is assigned as a parameter of the job statement. Its format is MSGCLASS=x, where x is any single letter (A-Z) or digit(0-9). If no message class is specified, the default class specified for TSUMCLAS, STCMCLAS, or the device through which the job was read, is used.

Output Class Considerations

The system programmer should assign output classes in a manner that distinguishes types of output and results in the most efficient use of devices. JES2 automatically balances output scheduling which makes the assignment of classes less important than in MVT. However, classes should be assigned with the following characteristics:

- Data to be processed by standard JES2 writers should be distinguished from that to be processed by external writers.
- Data placed on different devices and data placed on similar devices but with different characteristics should be in a separate classes. It is not necessary, however, to use classes to separate data with different punch interpretation UCS and FCB requirements if the data is processed by a JES2 writer, since JES2 handles these parameters automatically. Class should be separate if an external writer is used to print this data.
- Class should be assigned to give different priority to different types of data such as that to be printed on a different work shift.
- Classes need not be specified to give priority to short data sets, since JES2 priority calculation can be used for that purpose.

JES2 Output Selection

When assigning priority, classes, form requirements, etc., for data sets, the system programmer should balance the choices against the criteria used by JES2 to select the output data set to be processed.

Setup Characteristics

As established during JES2 initialization and altered by the operator, each JES2 controlled printer and punch possesses setup characteristics and a setup mode-manual or automatic. Setup characteristics are class, destination, forms, print train, and either carriage control tape or forms control buffer. (For a 1403 printer, JES2 uses the FCB parameter as carriage control tape name. The operator is requested to mount carriage tapes by this name in a manner similar to mounting forms.) Setup characteristics determine the data set groups that are eligible for processing on this device. Each locally attached printer and punch possesses either a destination of LOCAL or a specific device-name destination. If the device possesses a specific destination name, it is eligible to process only data sets specifically routed to it by JCL or the operator. Each remotely attached printer and punch possesses a destination that is the name of the workstation to which it is attached or the installation-assigned name of a remote pool of devices. Remote devices are eligible to receive only that output directed to them by JCL or the operator.

Setup mode determines the manner in which data sets are selected, automatically or manually. In automatic mode, after all data sets with characterstics matching the setup for a particular device have been selected, JES2 requests that the operator change the setup for that device.

In manual setup mode (operator-controlled), only data set groups with characteristics matching the setup of the device are selected. Manual mode printers do not request a new setup when there is no more work in the queue. The printer becomes idle.

When a device is available for output, JES2 selects job output elements according to the following algorithm:

1. The setup priority
   - First choice is between JOEs with setup requirements exactly matching those currently on the device.
   - Second choice is a JOE specifying a setup not currently being processed by any output device.
   - Third choice is a JOE specifying the standard forms setup as described by the STDFORM, PRTUCS, and PRTFCB JES2 initialization parameters.
2. When the setup has been selected, the first class specified for this device by the operator, or during JES2 generation, is chosen.
3. When setup and class are selected, the highest output priority JOE with these characteristics is chosen.

Some implication of the setup algorithm are:

- If an output device has been setup explicitly by the operator ($T command), JES2 does not setup another device to process data specifying that setup — unless the explicit setup is the same as that for standard forms. This is true no matter how many devices are idle, unless explicitly setup by the operator.
- Output matching an existing device setup and class is processed before output with no active setup, regardless of relative priority of the jobs producing the output.
- Output with setup requirements not loaded on any device is preferred over output with the setup loaded by the device busy.
- Installations should ensure that class and setup conflicts do not cause data to be overlooked. Commands are provided to determine output backlog.

Demand Setup

For those installations wanting job-related data sets, regardless of setup requirements, to appear together on the output listing, a demand setup option (DMNDSET) can be specified during JES2 generation. The output data sets of the job, possibly with several different setup requirements, are then placed in the same data set group. The message data set is the first one in the group, therefore its characteristics are used as those of the group for setup purposes.

The operator is requested by JES2 to set up the device as different setup requirements become necessary. Responding to demand setup requests is identical to responding to automatic setup requests.

Defaults

Defaults are assumed for any data set characteristics that are not specifically requested on the SYSOUT DD statement or the JES2 OUTPUT statement. Any FCB or UCS image that is specified as default by the installation will be used to print any data set that does not specify an FCB or UCS parameter. JES2 uses the name '****' when requesting from the operator a default FCB or UCS image. The operator satisfies this request by mounting any image specified as default. If one or more of the parameters (FCB,UCS, form) is not specified, then any default will satisfy it.

The form used for all data sets not specifying a form is identified (STDFORM) during JES2 generation. This is the standard from for both printers and punches.

Operation for Printers (Punches)

An installation generally has one or more printers (and/or punches) in manual setup (operator-controlled) mode for processing output that requires the most common combination (standard setup) of form, print train, and carriage control. The remaining printers are in automatic mode. Initially each printer is assigned setup characteristics, and a set of output classes from which to select data sets.

For each printer, data set groups are dequeued that have characteristics matching the printer setup characteristics. As automatic-mode printers exhaust the queue of data sets specifying their current setup, a data set group with a different set of characteristics is chosen and the operator notified (message HASP190) to change the setup.

An operator can respond to a request for a setup change from an automatic mode device by doing one of the following:

- Execute the request, then issue the $S command to the device.
- Allow the use of the setup only for the data set group that requested it, by issuing the $S command followed by the $P command. The $P command causes the device to become idle after printing the current data set group.
- Force an alternate setup on this data set group by issuing the $T command, followed by the $S command. The device must be set up, however, and the $T and $S commands repeated for each data set in the group. Header and trailer pages are considered data sets for this sequence.
- Cause the selection of an alternate data set group by holding ($H command) the job, then issuing the $I or $E command, which causes the data set group to be requeued in a held state. The held group must be released later by the operator.
- Delete the data set group by issuing the $C command to the device. Another data set group is then selected for the setup on this device, or another setup is requested.

The operator can also change characteristics of a manual mode device or change the mode of the device if the device is idle.

Output Routing

A user can route output to a specific local or remote device, to a specific remote station, to a remote workstation, or to a pool of remote workstations. The user can route a specific data set via the DEST parameter of the DD statement, a specific data set or group of data sets via the /*OUTPUT statement, or the entire print/punch output for the job via the /*ROUTE statement. DEST cannot be used to route a specific device.

If the destination for a data set is specifically stated on the /*OUTPUT statement, or via the DEST parameter, it is used. For data sets with no destination specified, the destination on the /*ROUTE statement or a default is used. The default print and punch destinations may be specified on the reader from which the job was received. If not, the default becomes the location (LOCAL or RMTnnn) from which the job was received.

The system programmer specifies the destination number (printer or punch) for each local and remote device, and for each remote station, during JES2 initialization. If a destination number is specified for any device, that device is eligible to receive only data which is specifically routed to it.

Destination names are of the form PRINTERn or PRINTRnn, PUNCHn, RMTnn, or LOCAL. LOCAL indicates any device attached to the local CPU. The n or nn is a numeric destination ID assigned to the device or remote stateion during initialization. The form PRINTERn must be used if the installation has less than ten printers; the form PRINTRnn must be used if ten or more printers are specified during JES2 generation.

By assigning the same destination id to a group of remote stations or a group of devices, the system programmer can create a remote pool or device pool.

Processing Held Data Sets

A data set is explicitly held via the HOLD parameter of a DD statement, or by specifying HOLD during dynamic allocation or deallocation.

A data set can also be implicitly held if it is in a class that is held and the job's MSGCLASS is also a held class. Since for a data set to be implicitly held, both the class in which it is written and its MSGCLASS must be held classes (the $$X JES2 initialization parameter is used to speicfy held classes), a programmer can control the holding of data sets using only the MSGCLASS parameter.

Figure 25 describes a TSO-submitted job. Assuming for this job that class and MSGCLASS are defined as follows:

- MSGCLASSes A and M are defined as held.
- MSGCLASS D is not defined as held.
- Sysout classes D and C not defined as held.
- Sysout class M is defined as held.

Then for the job described in figure 25:

- Since MSGCLASS=A, the SYSUT2 and SYSPRINT data will be held.
- SYSUDUMP will not be held.
- If the MSGCLASS were changed to C, none of the data would be held.
- If this JCL is submitted through TSO, it can be held with MSGCLSS=A. The same JCL can be submitted from an RJE terminal with MSGCLASS=C and the output will be printed at the RJE station.

```
//TSOUSER     JOB   name,MSGLEVEL = 1,MSGCLASS = A
//STEP        EXEC  PGM = IEBGENER
//SYSPRINT    DD    SYSOUT = A
//SYSUDUMP    DD    SYSOUT = D
//SYSUT1      DD    DSN = USERA.DSN1.ASM,DISP = OLD
//SYSUT2      DD    SYSOUT = M,DCB = (RECFM = F,LRECL = 80,BLKSIZE = 80)
//SYSIN       DD    DUMMY
```

Figure 25. Sample JCL for TSO-Submitted Job

A held data set is enqueued in a special queue. Job output elements are not built for a held data set.

Data sets are released from the HOLD state either from a time sharing terminal or by the output operator command ($O). *Only data sets in the HOLD state can be retrieved with the TSO OUTPUT command.*

External Writers

After output is described by job output elements and queued in priority order in the job output table, the output can be written by the JES2 writer or an external writer. An external writer can be standard IBM-supplied external writer processor, or an installation-written writer name on the SYSOUT DD statement. The operator starts an external writer in a private address space, and the data is written using the QSAM access method.

For details on the external writer, see Part III of this manual.

3540 Diskette Writers

When SYSOUT data sets are to be written on 3540 diskettes, the 3540 diskette writer program must be used. See *OS/VS2 IBM 3540 Programmer's Reference* for details.

Output Separation

The JES2 writer uses an 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.

For data processed by a JES2 writer, the JES2 separator pages are written before and after the writing of the output represented by one JOE.

The JES2 Print Separator

JES2 START JOB and END JOB separator pages consist of one-half page of blocked letters specifying the jobname, job id and output class; plus a single line of information duplicated as specified by each installation. All alphanumeric and all national characters are represented in blocked letter format. (The installation specifies the total number of lines on the separator page. If less than thirty, no blocked letters will appear.) The operator may request separator lines or cards via issuing a "$T device,S=Y[ES]" command. This function may be deleted by issuing a "$T device,S=N[O]" command. The default status is "S=YES" unless specified by an initialization option. An example of the information line is as follows:

Columns	Contents
1- 4	asterisks(*)
5	output class
8- 12	START CONT END
15- 22	job id assigned by JES2
25- 32	job name
35- 54	programmer name from job card
57- 60	ROOM
62- 65	room number
68- 78	time of printing the page in the form: hh.mm.ss. AM or PM
80- 88	date of printing the page in the form: day month year
91- 98	name of JES2 output device
101-104	SYS
105-108	system id from SMF
111-118	job id assigned by JES2
121-125	START CONT END
128	output class
129-132	asterisks (*)

The JES2 Punch Separator Card

Each job's punch output will optionally be preceded by an identification card. Below is an example of the current card which contains the programmer room number and internal job number. To make the card easy to identify, it has an 11-punch and a 12-punch punched in all 80 columns. To make the room number and job number easy to read, each digit is extended over ten columns. Alphabetic characters are converted to digits as follows:

Alphabetic Characters	Numeric Punch
A or J	1
B, K or S	2
C, L, or T	3
D, M, or U	4
E, N, or V	5
F, O, or W	6
G, P, or X	7
H, Q, or Y	8
I, R, or Z	9

Remote Job Entry

Remote job entry is the ability to submit jobs and receive system output at remote facilities as if the jobs had been submitted at a local facility. The remote facility must be attached to JES2 by a (point-to-point) binary synchronous communication link. The remote facility becomes a logical extension of the local computer facility and is expected by JES2 to be under the control of a person called a remote operator.

There are two types of remote job entry stations. The first type is the *remote terminal,* that does not have a CPU. A remote terminal, for example, a 2780 or 2770, can be used for entering jobs into and receiving data from JES2. The second type is a *remote workstation* that does have a CPU. A processor, for example, System/3 or System/370, executes a JES2 generated program that allows the processor to send jobs to and receive data from JES2. A remote workstation is established by a JES2 program, RMTGEN, during system generation or later. Also part of the workstation are printers, punches, card readers, and a console. A *remote station* is a composite term for a remote terminal and a remote workstation.

Reading, printing, and punching between the CPU and the remote terminal take place one action at a time. For example, it is either transmitting print data or transmitting punch data or reading an input stream. The remote operator may influence the order of these events. A discussion of how this is done is presented later in this section under, "Altering the Sequence of Operations from a Remote Terminal."

Communication between the local CPU and remote workstations uses a JES2 facility called MULTI-LEAVING that allows multiple print and punch streams to be transmitted at the same time and multiple console messages and input streams to be received by JES2. With MULTI-LEAVING, you can have several operations going simultaneously. Operators at remote terminals and at workstations that have no console can enter commands into the input stream in the normal manner. Command replies will be scheduled back to the remote station for printing on a remote printer.

Remote lines can be configured as dedicated or non-dedicated. This configuration is established during initialization when the remote stations are specified. If the station parameter, RMTnnn, designates a line number, the line is dedicated to that station. Lines that are not pointed to by a station parameter at initialization are non-dedicated lines and are eligible to be dynamically connected to any non-dedicated station.

Remote stations that are not physically connected to the CPU, that is, stations that must be connected via dial facilities, normally do not specify a dedicated line so that the station may be connected to any available non-dedicated line. There are other reasons for specifying a line as non-dedicated even if the line is physically connected to a remote station.

- A sign-on card is not required for connecting stations to dedicated lines, and is ignored, since the station is considered active when the line is started. Therefore, line and station password authorization is only enforced for non-dedicated lines and stations.
- One physically connected station can be initialized as multiple non-dedicated stations for use by different groups or at different times. The period of use of each such logical station would be defined by sign-on and sign-off. Data routed to the logical station will only be transmitted while that logical station is signed on.
- If remote stations are initialized as non-dedicated, one remote station can be used as backup for an inoperable station by being signed on with the inoperable station's id.
- A station attached to a dedicated line is considered active whenever the line is active. Line activation is under control of the central operator. The central operator is not aware of station usage in this case. (He is aware of station usage when non-dedicated stations are signed on and off via the console). Also, JES2 allocates resources for remote lines while they are active, which is only between sign-on and sign-off for non-dedicated lines.

One advantage in specifying lines as dedicated is that the station can be used without signing on the station, a manual process at all remote terminals.

It is possible to configure lines and stations that must be connected by dial facilities as dedicated. However, there can be only one station id and set of station characteristics associated with the dedicated line.

Starting Remote Job Entry

Since teleprocessing lines are never considered active at JES2 initialization, each line must be activated using a JES2 start command ($S) either by the operator, through a command stream entered, for example, through the JES2 initialization deck, into a job stream, or through the automatic command processor.

The first action taken at the non-dedicated remote station is the submission of a sign-on statement. (Sign-on is ignored for dedicated lines.) The format of this statement must be:

Column	Description
1	/*SIGNON
16	REMOTEnnn
25	password1
73	password2

- REMOTEnnn defines the remote station requesting sign-on. *The numbers must be left justified with no leading zeros.*
- Password1 defines the password established at initialization or changed by the operator for that line. If the line has a password, then password1 is required. To establish password1, set the LINEnnn JES2 initialization parameter. This password can be changed or invoked by the operator with the $T command.
- Password2 defines the password established at initialization that is assigned to each terminal. If the terminal has a password, then password2 is required. To establish password2, set the RMTnnn JES2 initialization parameter. The password ensures that the station signing on is a valid station.

A line is dynamically allocated when activated. A line can be deactivated and deallocated using the operator's JES2 stop command ($P).

A remote device is considered active when its remote station becomes active provided that the device is specified for automatic start (by the START subparameter in the Rnnn.RDm, Rnnn.PRm, or Rnnn.PUm initialization parameters). Otherwise, the device is considered inactive and must be started either by the remote or local operator command.

Altering the Sequence of Operations from a Remote Terminal

Two JES2 options are provided to allow the remote terminal operator control of the sequence of operations at the remote terminal.

During JES2 generation, the system programmer can specify a delay time, using the $WAITIME parameter, that will take effect between printing and punching the output of each job. This delay gives the operator the opportunity to ready the card reader and change the terminal status to transmit data. JES2 will sense this condition and read the input stream before resuming printing or punching.

When each printer or punch device is defined at JES2 initialization, using the Rnnn.PRm or Rnnn.PUm parameters, the suspend mode of operation can be specified or negated. If the suspend mode is in effect, the remote operator can alter the sequence of operations by stopping the output device. When the device is again readied by the operator, JES2 will simulate an interrupt situation by flushing its current I/O buffers and printing the remote separator page, if any. JES2 will then determine if the remote card reader is ready. If so, the input will be read in. If not, the highest priority output will be selected. This can be resumption of the current operation or another data set. The delay must be sufficiently long for the terminal to notify JES2 of the stopped device state. The time depends on the terminal type. If suspend mode is not in effect, the current operation is resumed after the device is readied again.

Options for Disconnecting Remote Lines

At JES2 initialization, the system programmer can use the LINEnn statement to choose whether each line is to have the abortive disconnect feature. If the feature is selected, a line is automatically disconnected by simulating a $E command sequence when the transmission control unit detects a not-ready data set. If the feature is not selected, the line will remain active and wait for the data set to be made ready or for operator action. The conditions under which a transmission control unit may detect a not-ready data set are dependent on line configurations.

The system programmer can also cause JES2 to automatically disconnect an inactive station by coding a non-zero value into the DISCINTV parameter of RMTnnn at JES2 initialization. When this amount of time has elapsed with no data sent or received on the line, JES2 will disconnect the line by simulating a $E command sequence.

SMF Accounting Record

SMF accounting records, types 47, 48, and 49, contain information useful for tracking the use of remote stations.

- Type 47 indicates whenever a line is started or a station signs on.
- Type 48 indicates whenever a line is stopped or a station signs off. It also contains statistical information.
- Type 49 indicates whenever a station uses the wrong password when trying to sign on.

The JES2 patching facility, automatic command processing, the flow for time sharing and started tasks, and the multi-access spool are described in this section.

Automatic Command Processing

The operator may specify from the console or through a local reader that certain commands or strings of commands take effect automatically at specific times or at regular intervals. The procedures for using the following commands to do this are in *OS/VS2 Operator's Library: Reference (JES2),* GC38-0210.

- Start Automatic ($SA): starts automatic command processing.
- Set Automatic ($TA): displays, specifies, or modifies the strings of commands (the "automatic command elements"). This command can also selectively cancel selected commands.
- Cancel Automatic ($CA): cancels all previously entered automatic commands.
- Halt Automatic ($ZA): stops all automatic command processing until it is restarted.

Typical reasons for using automatic command processing are to provide periodic status displays and to cause the operator to do no more work than necessary for common, preset routines or schedules. For example, if it is normal at the installation to do one specific kind of processing at 8 AM, and another at 9 AM every morning, it is possible to preset automatic command processing to issue the operator commands that would ordinarily be necessary at those times.

Writing a Day's Work Scheduler

Establish the use of automatic command processing with the &NUMACE parameter at initialization. Enter the commands with the $T operator command or write a program to put cards through an internal reader.

The following statements represent sample cards placed in the initialization deck:

```
(1) $TA,T=10.30,'$SLNE1,LNE2,LNE3'
(2) $TA,T=12.30,'$TI1,ABC;TI2,XBC;L=A'
(3) $TA,T=16.15,'$PLNE1,LNE2,LNE3;DM1-9,''PLEASE SIGN OFF ASAP'''
(4) $TA,T=16.45,'$ELNE1,LNE2,LNE3'
```

These four statements mean the following:

```
(1) Start the three remote job entry lines defined.
(2) Modify these initiators.
(3) Prepare to stop the remote job entry lines and give a warning to users who
    are currently using the system.
(4) Halt the remote job entry lines.
```

The sample cards show various times of the day set aside for routine processing that are part of the standard day's work.

A common source of these commands may be a user-written program for scheduling the day's work. This program can use the internal reader to get the commands into the subsystem. To write this program, observe the following considerations:

- When more than one command is to execute at approximately the same time, you should combine them into one command text entry of multiple commands.

- The responses to the command within the text will normally be directed to the in-line messages area and to any consoles receiving MCS route code 1 unless you use the L=caa operand as described in the operator's reference manual cited above.
- Multiple commands with responses to out-of-line area on graphic consoles will normally be automatically overlaid too rapidly for the operator to view. Avoid this kind of command sequencing.
- The authority of the internal reader determines which of your commands entered through it will be valid. See the operator's reference manual for discussion of command authority relative to automatic command processing.
- The day's work scheduler program should limit the number of automatic command entries to a value that does not overload the system consoles or leave the operator insufficient resources for his interval status displays.
- An entire automatic command processing entry must fit on an 80-column card image.

Limiting Considerations

When automatic command processing is active, the command entered at system speed (rather than at operator speed) may tend to congest the system. In turn, the system response to the commands may tend to flood the console with the response messages.

If the installation is experiencing difficulty either with congesting the system or flooding the console with messages, re-evaluate the mix of commands submitted with automatic command processing and try some changes.

Automatic command processing may also terminate itself prematurely under the following conditions:

- The operator enters the "$ZA" command to halt automatic command processing and then lets 24 hours or more elapse without restarting it.
- The operator specifies a start time for automatic command processing that is either before midnight of the current day or more than 24 hours later than the current moment.
- The system becomes so congested that the automatic commands are delayed approximately five minutes.

Make sure the operator fully understands the procedures for using automatic command processing. They are fully outlined in the operator's reference manual cited at the beginning of this section.

The JES2 Patching Facility

The JES2 patching facility makes temporary patches to any module in JES2 or to any absolute storage address in the address space into which JES2 is loaded. Because these patches are valid only until a module is reloaded, they must be applied every time that JES2 is started. These patches are applied at the time JES2 is initialized. The patching facility statements are submitted to the JES2 initialization data set.

Modules which are marked refreshable should not be patched since a system refresh will nullify the effect of the patch. Since pages in the Pageable Link Pack Area (PLPA) are not paged out, any patches applied to modules residing in this area will not be effective once the page in which the patch resides has been paged in. For this reason, modules in SYS1.LPALIB data set (for example, HASPSSSM) must be "fixed" via an entry in the SYS1.PARMLIB fixed list before patches are applied via the JES2 patching facility.

The JES2 patching facility in the JES2 initialization data set can be specified in either the JES2 patching format or in the SPZAP format. All patches in the JES2 patching format should appear before any SPZAP format patches. These two methods for patching are explained in the following sections.

Rules for Coding Patching Statements

The following conventions are used in the parameter descriptions:

- Uppercase letters must be coded exactly as shown.
- Lowercase letters represent variables for which you must substitute specific information or specific values.

The following syntax rules apply to the coding of the parameters.

- The size of a patching facility statement is 71 bytes.
- The statements cannot be continued on successive cards.
- The statements may begin in any column, but the operation name must precede the parameters.
- A statement beginning with an asterisk is a comment statement.
- There must be at least one blank between the specified operation name and the first parameter.
- All parameters must be separated by at least one blank space.

Format of the JES2 Patching Facility Statements

The format of the JES2 patch statement is as follows:

```
operation   csect   address     data     comments
```

where:

operation
: defines the operation to be performed as follows:

  REPLACE
  REP
  R
  : The data on the statement will replace the data at the location specified by the "csect" and "address" fields.

  VERIFY
  VER
  V
  : The data on the statement will be compared with the data at the location specified by the "csect" and "address" fields. If the data does not compare, an error message is displayed in the Parameter Library List data set.

  BASE
  B
  : The base used to adjust address values that are to be specified in any subsequent VER and REP statements is to be modified. This offset is initialized to a value which is based upon the distributed CSECT and assembly module relationships of JES2, and the BASE statement need only be used if this relationship is modified locally. The "data" field on the BASE statement is ignored and may be omitted.

csect
: specifies the control section (or control block) in which the data to be verified and/or modified is resident. If an asterisk (*) is coded, the CSECT in effect on the previous JES2 patch statement is used. Figure 26 contains a list of the possible names which can be coded and CSECTs to which these names refer. Note that the patch statement name is simply the CSECT name with the first four characters (always "HASP") omitted.

address
specifies the hexadecimal address of the data to be verified and/or modified. This address does not have to be aligned in any way and can consist of one to six digits (with or without leading zeros). The address should be taken directly from a JES2 assembly listing containing the referenced CSECT. If an asterisk (*) is coded, the address will be interpreted as one greater than the last address reference on the previous JES2 patch statement.

data
specifies the bytes of data that are to be verified and/or modified at the specified location. The number of bytes of data defined must be specified as a multiple of two hexadecimal digits. If desired, the data within the parameter may be separated by commas (never blanks). If all the data will not fit into one patch statement (71 bytes), then another patch statement must be used.
If the data specified contains the address of a location within a JES2 CSECT, the JES2 patch processing routine will relocate this data by the base location of the CSECT if indicated. This relocation is indicated by following the data to be relocated with the name of the CSECT (abbreviated as in "csect" above) enclosed in parentheses. The address specified in the "data" should be taken directly from a JES2 assembly listing containing the referenced CSECT. The data to be relocated should contain at least six hexadecimal digits (three bytes), and, if more than six digits are specified only the last eight digits (four bytes) will be considered in the relocation process. If an asterisk (*) is coded instead of a CSECT name, the CSECT in effect for the location of the current patch statement is used.

comments
following the last required parameter and its blank delimiter, the rest of the control statement space can be used for comments.

Examples of JES2 Patching facility statements:

```
*
*               CORRECT PROGRAMMING ERROR IN HASPRDR
*
VER RDR      1E2 41E00001                      VERIFY INSTRUCTION
REP  *       1E2 4590B258                      BAL TO PATCH SPACE
VER NUC      258 B258,B25A,B25C,B25E,B260      VERIFY PATCH SPACE
REP  *       258 41202000                      ADD INSTRUCTION
REP  *        *  41E00001                      REPLACE INSTRUCTION
REP  *        *  07F9                          RETURN
 *
 *              CORRECT BAD ADDRESS CONSTANT IN HASPPRPU
 *
VER PRPU     32E 58F0C65C                      VERIFY INSTRUCTION
REP  *       330 B264                          MODIFY DISPLACEMENT
VER NUC      264 B264,B266                     VERIFY PATCH SPACE
REP  *       264 00000520(PRPU)                ADDRESS CONSTANT
```

SPZAP Patch Statement Formats

Two formats are required for defining a SPZAP patch. They are the same formats of the control statements for the OS/VS2 AMASPZAP service aid. The first format type defines what module you want to change; the second format type defines what change you want made to the module.

The first format type is used to indicate the control section that is to be the object of subsequent operations. The format of this section is as follows:

```
NAME    member  csect   comments
```

where:

NAME
specifes a keyword that must be coded.

member
specifies the member name on the AMASPZAP control statement. This field is ignored on a SPZAP patch statement, but must be provided for AMASPZAP compatability.

csect
specifies the control section (or control block) in which the data to be verified and/or modified is resident. While this field is optional on the AMASPZAP control statement, it is required on the SPZAP patch statement. Figure 26 contains a list of the possible CSECTs which can be coded.

comments
following the last required parameter and its blank delimiter, the rest of the control statement space can be used for comments.

The second format type is used to indicate what operation is to be performed. The format of this section is as follows:

```
operation   offset  data    comments
```

where:

operation
specifies the operation to be performed as follows:

REP
The data on the statement will replace the data at the offset into the CSECT specified on the previous NAME statement.

VERIFY
VER
The data on the statement will be compared with the data at the offset into the CSECT specified on the previous NAME statement. If the data does not compare, an error message is displayed in the Parameter Library List data set.

BASE
The base used to adjust offset values that are to be specified in any subsequent VERIFY and REP statements is to be modified. This statement should be used when the offsets given in the VERIFY and REP statements for a CSECT are to be obtained from an assembly listing in which the starting address of the CSECT is not location zero. The "data" field on the BASE statement is ignored and may be omitted.

offset
specifies the hexadecimal displacement of the data to be verified and/or modified in the specified CSECT. This displacement does not have to be aligned in any way and can consist of two, four, or six digits.

data
specifies the bytes of data that are to be verified and/or modified at the specified location. As with the offset parameter, the number of bytes of data defined must be specified as a multiple of two hexadecimal digits. If desired, but again, the number of digits between commas must also be a multiple of two. If all the data will not fit into one SPZAP statement (71 bytes), then another SPZAP statement must be used.

comments
following the last required parameter and its blank delimiter, the rest of the control statement space can be used for comments.

JES2 Patch Name	AMASPZAP Patch Name	CSECT Referenced
ABS	HASPABS	Absolute Storage Location
ACCT	HASPACCT	HASPACCT
BLKS	HASPBLKS	HASPBLKS
COMA	HASPCOMA	HASPCOMA
COMM	HASPCOMM	HASPCOMM
CON	HASPCON	HASPCON
INIT	HASPINIT	HASPONIT
MISC	HASPMISC	HASPMISC
NUC	HASPNUC	HASPNUC
PRPU	HASPPRPU	HASPPRPU
RDR	HASPRDR	HASPRDR
RDRO	HASPRDRO	HASPRDRO
RSCN	HASPRSCN	HASPRSCN
RTAM	HASPRTAM	HASPRTAM
SSSM	HASPSSSM	HASPSSSM
SSVT	HASPSSVT	Subsystem Vector Table
XEQ	HASPXEQ	HASPXEQ

Figure 26. Patch Name to CSECT Reference

Time Sharing Logon and Started Task Flow

Time sharing logon and started system tasks appear to JES2 as special form of jobs that are received from designated internal readers. These jobs are enqueued in special job classes (TSU and STC) and are assigned a MSGCLASS that is set during JES2 initialization (TSUMCLAS and STCMCLAS). They are presented to the converter with parameters (&RDROPSU or RDROPST) established first during JES2 generation and later during JES2 initialization.

The time sharing message class (TSUMCLAS) becomes the output class for all dynamically allocated sysout data sets for which a class is not specified, and becomes the MSGCLASS for all submitted jobs with no MSGCLASS parameter in the JOB statement.

Time sharing users can dynamically allocate sysout data sets, dynamically unallocate them (spinoff), and print them at the time sharing terminal (OUTPUT command).

Multi-Access Spool

Previous sections have described JES2 functions on a single system (uniprocessor, MP158, or MP168) operating under a single copy of the MVS control program, as shown in Figure 17. It is also possible to operate from two to seven such systems (each a uniprocessor or MP) as members of a multi-access spool complex, as shown in Figure 27.

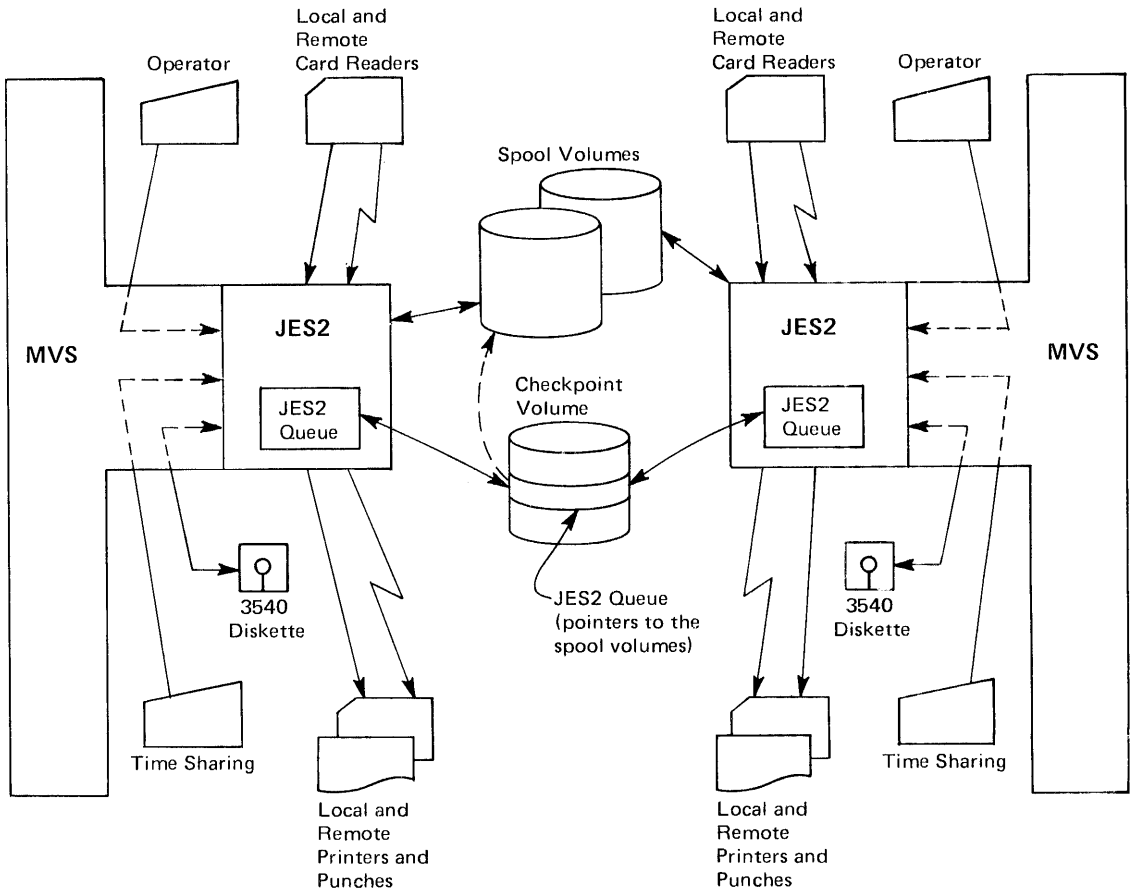

Figure 27. Two-System Shared JES2 Complex

The operation of each system in the complex is independent and includes all functions previously described for single JES2 systems. That is, each JES2 system can read jobs from local and remote card readers, schedule jobs for conversion and execution under MVS initiators, print and punch results at local and remote output devices, and communicate with operators and time sharing users. However, all spool volumes and the volume containing the SYS1.HASPCKPT data set are used by all systems in the complex.

The systems logically share a common JES2 queue. The workload may be balanced among systems by allowing jobs to execute on whatever system has an idle initiator with the correct class and print or punch, on whatever system has an idle device with the correct class, routing, setup, etc.

Since all systems are functionally the same, if one system in the complex fails, the others may continue processing from the common queue. Only work in process on the failed system is interrupted; this work may be recovered by a warmstart of the failed system while other systems continue processing, or, as explained later, by operator command on one of the other systems.

Shared DASD hardware features (two channel switch, two channel switch additional, and string switching) are used to access data on all spool and checkpoint volumes. A copy of the JES2 queue and other status information (e.g. spool space allocation maps) is written to the SYS1.HASPCKPT data set for possible warmstart, as with a single JES2 system. This information is available to all systems, one at a time, as needed. RESERVE/RELEASE channel commands are used to prevent simultaneous referencing and updating of information kept in the SYS1.HASPCKPT data set.

Each system in the complex must have at least one channel path to each spool and checkpoint volume, and these devices must be specified as SHARED during MVS system generation. It is recommended that each CPU of an MP158 system in the complex have a channel path to each shared volume.

Configuration

To use the multi-access spool feature, the initialization or generation &SPOOL parameter and the &CHKPT initialization parameter must specify the same volumes for all systems in the complex. To make the common spool and checkpoint data compatible, all systems must specify the same values for the &BUFSIZE, &NUMDA, &NUMTGV, &MAXJOBS, &NUMJOES, &NUMRJE, and &SPOLMSG generation parameters.

For operational consistency, it is recommended that the &MINJOES, &TGWARN, &XBATCH, and &XBATCHN generation parameters be specified the same in all systems of the complex.

It is also recommended that local unit record devices and RJE lines be given unique JES2 device names over the whole complex. The &NUMLNES, &NUMPRTS, &NUMPUNS, and &NUMRDRS generation parameters of each JES2 system should be specified as the total number of each type of device in the complex. This allows all devices to be attached to one system (with appropriate manual switching) if other systems are not operational.

Similarly, the LINEnn, PRINTERnn, PUNCHn, and READERn initialization parameters should be set so that a device has the same name no matter which system it is attached to. For example, if a 3211 printer is one of four local printers on a two system complex, it could be initialized as:

```
PRINTER4 UNIT=102
```

for one system and:

```
PRINTER4 UNIT=302
```

for the other, if it were attached to different channels on the two systems.

A local unit record device or RJE line can only be attached to one system at any instant. JES2 initialization will detect devices which are not online and place them in a DRAINED state. Later, the device may be activated by entering the $P device and VARY OFFLINE commands on the system to which it is attached, performing hardware switching, then entering the VARY ONLINE and $S device commands on the new system. The $S command will fail if no hardware path exists.

The &NUMRJE generation parameter must be the same in all systems of the complex, as previously described. This parameter represents the total number of RJE lines known to the entire complex. Each RJE line has a unique name, no matter which system it is attached to. Therefore, the RMTnnn, Rnnn.RDm, Rnnn.PRm, and Rnnn.PUm initialization parameters should be specified the same in all JES2 systems of a multi-access spool complex.

Starting the Multi-Access Spool Complex

Before starting the complex, the TOD clocks on each system should be carefully synchronized with a single time source. Since this synchronization is externally performed and subject to error, the generation parameter $SYNCTOL is provided to specify the maximum error (in seconds) which JES2 should assume. If the synchronization error is actually greater than $SYNCTOL, then JES2 will not be able to detect certain illegal operator actions (e.g., performing cold start with other systems active). On the other hand, certain legal operator actions (e.g., warm start after system failure) will be disallowed if attempted before $SYNCTOL seconds have elapsed since system failure.

The members of the complex are specified by the Sn initialization parameters. For example:

```
S1  SID=K158
S2  SID=L168
```

defines a two system complex where K158 and L168 are the SMF system ids set during IPL of the systems. One system must initially do a cold JES2 start with no other systems active and must define all members of the complex. Other members join operation by warmstart and must also specify identical Sn parameters. A cold start is required to change or add members of the complex. If only one or no Sn parameter is specified, JES2 operates as a single system.

There are three types of warm starts:

- If a warm start is specified by the operator and JES2 detects that no other members of the complex are active, after operator confirmation a total complex warm start is performed. New spool volumes may be added, all in-process work will be recovered, and all unused spool space will be accounted for, as in single system operation.
- A warm start is performed when warm start is specified and other members of the complex are active. The warmstarting system joins the active complex and recovers only work in-process on the system at a previous failure, if any. No spool volumes may be added.
- Restart for another system is performed when a system has failed and cannot be immediately warm started. The operator enters the $ESYS command on any active member of the complex. In-process work on the specified system is recovered and made available for selection by other members of the complex, subject to system affinity for execution restart as discussed later under Queuing.

The algorithm for using the common JES2 queues and other information in the SYS1.HASPCKPT data set is determined by the HOLD=, MINDORM=, and MAXDORM= keywords of the QCONTROL initialization parameter. These need not be the same for all systems in the complex and should be set according to the number of members in the complex, relative CPU speeds, response requirements, etc. See *OS/VS2 System Programming Library: Initialization and Tuning Guide* for details.

Job Submission and Queuing

In a multi-access spool complex, jobs enter the common queue from any input source (local or remote) attached to any system in the complex. Unless special actions are taken, jobs will normally be eligible to execute in any system in the complex, selected by priority and the classes of idle initiators as in single system operation.

Started tasks and TSO users are an exception which execute only in the system in which they are entered. However, job queue entries also contain a system affinity for up to seven systems on the maximum complex and may contain an independent mode affinity.

Individual jobs may be given affinity to one or more systems (less than the total complex) and may be given affinity for independent mode by the SYSAFF=keyword on the /*JOBPARM card. Any input device (local or remote) may be set by the $T command to give system and/or independent mode affinities to all jobs read from that device. The /*JOBPARM card overrides the input device default.

If a job's affinity is to specific systems in the complex or to independent mode, the job can be selected only by the system(s) specified and only if the mode of the system (independent or not) matches that of the job.

System affinity may be useful for special processing requirements (e.g. emulation) not available on all systems of the complex. Independent mode may be useful for testing new components with selected jobs while in a shared complex.

The display commands ($DA, $DN, $DQ, $DJ) indicate (by SMF system id) the system in which a job is active or the system(s) eligible to process a queued job. The $TJ and $TALL commands permit affinities of jobs or all jobs with given affinity to be changed. The $TSYS command allows a system to be placed in independent mode. The $LSYS command displays the states of all systems in the complex.

If a system fails and jobs in execution are recovered and requeued for automatic restart either by a warmstart or the $ESYS command, those jobs are given affinity only to the failed system. If the failed system is unavailable, the operator may change affinity with the $TJ or $TALL commands to attempt restart on another system.

Priority aging is done only by the lowest numbered active system in the complex.

Duplicate jobname protection extends to all systems; i.e., if a jobname matches another active in execution anywhere in the complex, the job is temporarily delayed. See the TSO section that follows.

Output

Printed and punched output processing is very little different from single system operation. System affinity does not apply to selection of work from the JOEs.

Output work is selected by eligible devices, no matter to which system in the complex those devices are attached. Selection criteria are output class, routing (local or remote number), and set up just as in single systems. The automatic setup algorithm which prevents the same special forms from being requested for more than one local printer operates for all local printers in the complex.

The $CJ command entered from any system in the complex will cancel a job active on an input or output device attached to another system.

RJE

Configuration considerations for RJE lines in the complex were discussed previously.

JES2 enforces that the same remote number cannot signon more than one line any where in the complex at any given time. For dedicated lines, the user must insure this uniqueness by proper setting of line and remote initialization parameters as previously described.

The remote operator message queue operates across the entire complex. That is, any remote operator can send messages to any other remote (even if attached to different systems) and any central operator can send a message to any remote.

TSO

TSO userids are jobnames to JES2 and, in a multi-access spool complex, the duplicate jobname protection extends across the complex. A TSO logon will be rejected if a user of the same id is logged on elsewhere in the complex.

Jobs submitted by TSO users may execute anywhere in the complex, subject to affinities as previously discussed. However, held output data sets are accessible by the TSO OUTPUT command by the submitting user regardless of where logged on or where the job executed. Messages produced by NOTIFY= are also returned to where ever the TSO user is logged on or to where the job was submitted from, if the user is no longer logged on.

SMF

The SMF type 26 record contains system ids indicating which systems in the complex performed each major function of processing for a job: input, convert, execute, post-execute break into output elements, and purge.

The SMF type 6 records contain the system id which processed each element of output work.

Part III: Miscellaneous Job Management Facilities

Miscellaneous Job Management: This topic discusses restarting support, assigning special program properties, limiting user region size, changing the system log processing, updating MSTRJCL data set, and using external writers.

This section discusses the following facilities of JES2 and JES3 job management:

- Job scheduler restarting support
- Assigning special program properties
- Limiting user region size
- Updating the master Job Control Language (MSTRJCL) data set
- External writer

These facilities may be used by a system programmer in meeting the requirements of his installation for specialized job management support.

Job Scheduler Restarting Support

Job scheduler restarting support consists of job scheduler functions that allow either the resumption or the termination of a failing job. These functions gather information about the status of a job and its related control blocks. They provide information about the control blocks so that reconstruction of a failing job's scheduler work area (SWA) can take place to support the following restarting situations:

- Automatic step — permits execution to resume at the beginning of a job step.
- Automatic checkpoint — permits execution to resume from the most recently executed checkpoint in the user's program.
- Deferred checkpoint — permits execution to resume from a user-specified checkpoint upon resubmission of a job.
- System — permits the termination of active jobs in the event of a system failure.
- Continue — permits a job to continue at the next job step if the system fails during step termination.

For a detailed discussion of the checkpoint/restart facility, refer to *OS/VS Checkpoint/Restart,* GC26-3784.

Job Journal

The job journal is a temporary sequential data set that resides on the spool volume of the job entry subsystem (JES). It preserves a set of selected job-related control blocks that are necessary for restart processing.

The job journal is necessary because scheduler control blocks are maintained in the SWA in pageable storage. When the system fails, the address space containing the SWA is lost. When a job abnormally terminates, the job's SWA is released. Reconstruction of the SWA is possible because the job journal preserves up-to-date copies of the essential control blocks. This facility is available in the following restarting situations:

- Automatic step
- Automatic checkpoint
- Continue
- System

Unless the user specifies no job journal via the JES initialization parameters, each job is provided with a job journal. Without a job journal, the capability for automatic restarting is lost. In addition, jobs that are executing when the system fails forfeit their data set disposition processing if there is no job journal. For additional information about a "no journal" environment, refer to the discussion of the NOJOURN initialization parameter, under "Job Class Parameters," in *OS/VS2 System Programming Library: Initialization and Tuning Guide,* GC28-0681.

Two service routines process the job journal. They are:

- Journal write routine - determines which of the scheduler control blocks are necessary to restart a job, and then writes them to the job journal.
- Journal merge routine - merges the control blocks from the job journal to the SWA during restarts. This reconstructs the SWA to its condition prior to the job or system failure.

The job journal contains the following records:

- Step header
- Job control table (JCT)
- Step control table (SCT)
- Step input/output table (SIOT)
- Job file control block and its extension (JFCB and JFCBX)
- Passed data set information block (PDI block)
- Generation data group name table (GDGNT)
- Volume unload table (VUT)
- Account control table (ACT)
- Virtual input/output data set control blocks (VDSCBs - virtual data set control blocks, and DSPCT - data set page control table header)

For additional information about restart processing, refer to *OS/VS Scheduler and Supervisor Logic,* SY28-0624, SY28-0625, SY28-0626 (3 volumes).

Assigning Special Program Properties

The initiator can assign special properties to privileged programs whose names are in the Program Properties Table (IEFSDPPT), a nonexecutable CSECT in load module IEFSD060. Executing the SGIEFOPT macro instruction during system generation makes the table available for the initiator to use. When the initiator initiates a program, it scans the table to determine whether special properties apply to the program.

Each entry in the Program Properties Table consists of 12 bytes with the following contents:

Program Name: the eight-byte name specified in the PGM parameter on the EXEC statement for the job.

Program Properties: a one-byte field that indicates the special properties assigned to a program.

Bit	Meaning When On
0	The program cannot be canceled.
1	A unique protection key is to be assigned to the program. This key is defined in the next byte of the table entry.
2	The program cannot be swapped.
3	The program is privileged and will not be swapped unless address space is in a long wait. (A long wait is a wait that results from specifying LONG=YES as an operand on the WAIT macro instruction.)
4	The program is a system task and therefore will not be timed. (When a program is not timed, the system does not check for time limits and does not time for accounting purposes.) The program must be a one-step program initiated with a START or MOUNT command.
5	The program does not require exclusive use of the data sets that is requests. It must be a one-step program.
6	The program is to bypass password protection.
7	Reserved

Notes:

1. The properties represented by the various bit settings may not always be honored by the system. When a property is not assigned, the program generally will execute, but without the properties specified:
   - The program will not be assigned special properties unless it comes from an APF-authorized library.
   - The system task property (bit 4) will not be assigned to a job unless the job was started by a START or MOUNT command.
   - The system task property (bit 4) and the no-data-set-integrity property (bit 5) will not be assigned unless the job is a single-step job.
2. The requirements of the Initiator influence the need to maintain data set integrity as follows:
   - If one or more data sets requested by a program are not available when the job is to be initiated, the scheduler waits until the job can get exclusive control of all data sets that it requires. Although the job itself may not require data set integrity, the initiation process for the job does.
   - Jobs that request the no-data-set-integrity property (bit 5) will not be initiated if both of the following conditions exist:
     - the job requests a data set whose name is an alias for a data set that is unavailable during the job's initiation.
     - the job contains either a JOBLIB or STEPLIB.

Protection key: a one-byte field that specifies in bits 0-3 the unique protection key to be assigned to the program. (A protection key is assigned if bit 1 of the preceding byte is on.)

Affinity mask: a halfword that indicates the CPU affinity, which is a system generation option. Each bit in the 16-bit mask refers to a corresponding CPU identifier (0-F), assigned during system generation. For example, bit 0 corresponds to CPU 0. If bit 0 is on, the program is eligible to run on CPU 0. The bit mask should be set to X'FFFF' if affinity is not required.

The Program Properties Table includes five dummy entries for assigning additional program names to the list. The dummy entries initially contain the properties necessary for TCAM. The system programmer can execute the AMASPZAP service aid program to change these entries for the installation's purposes. To add more than five program names, the system programmer must update the source module, then assemble and link-edit it again.

The initiator always attaches programs in the problem program state. If a program must execute in supervisor state, it must issue a MODESET macro instruction. For information on MODESET, see *OS/VS2 System Programming Library: Supervisor.*

TCAM Message Control Program (MCP) names other than IEDQTCAM must be added to the Program Properties Table (IEFSDPPT). These names must be added to the PPT after system generation by using the AMASPZAP service aid. (For information on AMASPZAP, see *OS/VS2 System Programming Library: Service Aids.*) If more than six MCPs are required, the PPT must be reassembled to create more entries. An MCP will not operate unless its name is in the PPT. TCAM OPEN routines must run in key 6 and will abnormally terminate any caller who was not initiated in key 6. Prior to MVS, MCP names were put in the PPT to make the MCP non-cancellable, but the MCP would execute properly if its name were not in the PPT.

Limiting User Region Size - IEALIMIT

An installation can enforce a region-size limit by writing an exit routine that is invoked once per step when the initiator is establishing region size. If an installation-written exit routine does not exist, an IBM-supplied routine receives control.

The installation-written exit routine, which replaces the IBM-supplied routine, must be named IEALIMIT and must be link-edited into the nucleus. The routine must observe standard linkage conventions. Upon entry to the IEALIMIT routine, the register contents are as follows:

Register 1	number of bytes requested by the application program for its region (specified explicitly through the REGION parameter or implicitly through the default JCL value)
Register 13	address of standard save area
Register 14	return address
Register 15	entry point address for the IEALIMIT routine

If IEALIMIT receives control and the input register 1 contains a zero, then IEALIMIT returns a zero in register 1 and no limit is assigned (to a job, a started program, or a TSO user). No limit is set only when the REGION parameter is not specified and the default value is zero. If system programs require "no limit" then the installation must not use "small" default REGION sizes.

If the input register 1 is non-zero when IEALIMIT receives control, then IEALIMIT adds 64K to the contents of register 1 and returns. The IEALIMIT routine sets this limit on all types of requests for storage from subpools 0-127, 251, and 252.

After the IEALIMIT routine assigns the appropriate limit, it must pass to IEAVPRT0 via register 1 a numeric value that represents the imposed limit in bytes. (Note that a zero returned in register 1 indicates that a limit is not imposed.) IEAVPRT0 stores the value for future reference (for example, when subsequent GETMAIN requests are issued).

The REGION parameter specifies the amount of space to be allocated to a job. The IEALIMIT value specifies the maximum permissible region size. The system honors the REGION-parameter value unless it exceeds or equals the IEALIMIT value, in which case the REGION parameter is ignored. If the REGION-parameter value is lower than the IEALIMIT value, the result of a GETMAIN request depends on whether it is variable-length or fixed-length, as shown in Figure 28.

Type of GETMAIN	Result
Fixed-length:	
IEALIMIT value minus currently alloc space > request	Satisfied
IEALIMIT value minus currently alloc space < request	Rejected
Variable-length:	
REGION parm minus currently alloc space > max	Maximum is allocated.
min < REGION parm minus currently alloc space < max	Unallocated amount is allocated.
REGION parm minus currently alloc space < min	Minimum is allocated as long as IEALIMIT is not exceeded (in which case the request fails).

Figure 28. The Effects of IEALIMIT and REGION Values on Various GETMAINs

For example, assume that application program A has the following characteristics:

IEALIMIT value	150K
REGION-parameter value	100K
Currently allocated space	80K

Program A issues the following variable-length GETMAIN requests in the order indicated:

1. Request 5K-10K: 10K is allocated; currently allocated space is now 90K. Because the amount currently allocated (80K) does not exceed the REGION-parameter value (100K) and because the amount unallocated (20K - relative to the REGION-parameter value) is greater than the maximum amount requested (10K), the maximum amount is allocated.
2. Request 5K-100K: 10K is allocated; currently allocated space is now 100K. Because the amount unallocated (10K - relative to the REGION-parameter value) is between the minimum and maximum, the amount unallocated is allocated.

3. Request 40K-100K: 40K is allocated; currently allocated space is now 140K. Because the amount unallocated (0K - relative to the REGION-parameter value) is less than the minimum amount requested (40K), the minimum amount is allocated.
4. Request 15K-50K: the GETMAIN request fails. The amount unallocated (0K - relative to the REGION-parameter value) is less than the minimum amount requested (15K). If the minimum amount were allocated, the currently allocated amount would become 155K, which exceeds the IEALIMIT value (150K). Therefore, the request fails.

System Log

The system log is an integral part of MVS. It consists of dynamically-created data sets that record the communications among problem programs, operators, and the operating system. It will contain operating data entered by problem programs using the write-to-log WTL macro instruction. If the log is designated as MCS hardcopy, it can contain:

- Job time, job step time, and data from the JOB and EXEC statements of a job that has ended.
- Descriptions of unusual events that occurred during a shift.
- Write-to-operator (WTO) and write-to-operator with reply (WTOR) messages.
- Accepted replies to WTOR messages.
- Commands issued through operator's consoles and the input stream, and commands issued by the system.

Using the System Log

If the installation does not modify system log operation, it works as follows:

- At IPL, the system automatically allocates the system log data set as a class A SYSOUT data set.
- Subsequently the log keeps track of the number of entries it receives by counting the WTL macro instructions executed against it. After 500 WTLs, the system:
  - opens and allocates a new system log.
  - closes and dynamically unallocates the currently full log.

Changing the System Log Processing

The system programmer can alter the default operation of the system log to control the processing associated with the log data sets. He can change the SYSOUT class of the log data sets and the number of WTLs received before switching log data sets.

The processing of the log data sets can be controlled from the operator console or from a SYS1.PARMLIB member named IEASYSxx, where xx is a unique number (chosen by the installation) that identifies the member. This member must be included in the system during IPL, in response to the request to specify the system parameters.

From the console, the operator can control the processing with commands. (For further information about the operator commands, see *Operator's Library: OS/VS2 Reference (JES2),* GC38-0210.) For example, the operator can issue a WRITELOG command with the START operand after a system failure or after a WRITELOG command with the CLOSE operand.

The following SYS1.PARMLIB parameters initialize or alter the system log control values:

- LOGLMT — which controls the number of WTLs received before the system switches data sets.
- LOGCLS — which controls the SYSOUT class of the system log data set.

The LOGLMT value must be a six-digit number in the range 000001-999999. An all-zero entry value results in the system default of 500. When choosing the LOGLMT value, the system programmer should consider:

- Whether the system log is defined as MCS hardcopy.
- Whether the system log data is sufficiently critical to the system to require frequent allocating, switching, and queuing to a SYSOUT class.

The LOGCLS value must be one alphameric character. The default is class A.

The following example shows the correct format for including the LOGLMT and LOGCLS parameters in the IEASYSxx member of SYS1.PARMLIB when specifying the system parameters during IPL:

```
LOGLMT=004852,LOGCLS=L
```

The preceding example would cause the system log task to switch data sets after 4852 WTLs, and the job entry subsystem to queue the current data set to class L for SYSOUT processing.

Updating the Master Job Control Language Data Set

The master job control language data set (CSECT name and load module name are MSTRJCL) is a nonexecutable module that is created during system generation and resides on SYS1.LINKLIB. As provided by IBM, MSTRJCL contains data definitions for all system input and output data sets necessary for communications with the job entry subsystem. MSTRJCL also contains the START command that starts the job entry subsystem at initialization. Figure 29 shows MSTRJCL as it exists before it is assembled. During system generation, &SSNAME is replaced with the name of the job entry subsystem.

```
DC      CL80'//MSTRJCL          JOB MSGLEVEL=(0,0)'
DC      CL80'//                 EXEC PGM=IEEMB860,DPRTY=(15,15)'
DC      CL80'//STCINRDR         DD SYSOUT=(A,INTRDR)'
DC      CL80'//TSOINRDR         DD SYSOUT=(A,INTRDR)'
DC      CL80'//IEFPDSI          DD DSN=SYS1.PROCLIB,DISP=SHR'
DC      CL80'//IEFPARM          DD DSN=SYS1.PARMLIB,DISP=SHR'
DC      CL80'//SYSUADS          DD DSN=SYS1.UADS,DISP=SHR'
DC      CL80'//SYSLBC           DD DSN=SYS1.BRODCAST,DISP=SHR'
DC      CL80'//SMFMANX          DD DSN=SYS1.MANX,DISP=SHR'
DC      CL80'//SMFMANY          DD DSN=SYS1.MANY,DISP=SHR'
DC      CL80'// START &SSNAME'
DC      CL80'/*'
```

Figure 29. MSTRJCL Data Set

If an installation does not plan to use TSO, the system programmer can delete the TSOINRDR, SYSUADS, and SYSLBC data definitions. He can add other data definitions as necessary, provided that the data sets they define are allocated before the IPL that is to make use of them. In short, if the allocation of any data set defined in MSTRJCL fails, the IPL also fails.

Changes to MSTRJCL fall into two categories: modifying an existing statement, and adding or deleting a statement. To modify a particular statement, the system programmer can use the AMASPZAP service aid program. To delete an existing statement or add a new one, he must reassemble the MSTRJCL statements, including a MSTRJCL CSECT card and an END card with the statements shown in Figure 29. In the figure, &SSNAME should be replaced with the name of the job entry subsystem.

By deleting the statement that contains the START command for the job entry subsystem, the system programmer enables the console operator to specify the job entry subsystem during IPL.

External Writers

After output is queued by JES2 or JES3, the output can be written by the writer associated with the job entry subsystem or an external writer. An external writer can be standard IBM-supplied external writer processor, or an installation-written writer name on the SYSOUT DD statement. The operator starts an external writer in a private address space, and the data is written using the QSAM access method.

With an external writer, SYSOUT data sets can be written to devices other than local and remote printers and punches supported by JES2 or JES3. Installation-written writers can effect special processing of sysout data sets accompanied by special separator pages formatted by an installation routine. The user requests an installation-written writer by specifying the name of the writer (other than INTRDR or STDWTR which are reserved) on the SYSOUT parameter of a DD statement.

There are three rules for selecting data set groups for an external writer:

- Data set groups with data characteristics that include an installation-written writer name are not eligible for selection by a JES2 or JES3 printer or punch. An external writer should be started to the group's output class to process these data sets, or they should be dequeued specifically by the operator who can cause the external writer to dequeue data sets that specify a specific installation writer name, regardless of the output class. The JES2 $DF or JES3 *I,U command can be used to determine the classes in which data sets specifying installation-written writers are queued.
- Data sets that do not specify a special writer can be written by a JES2 or JES3 writer or an external writer, depending on operator action. If the same class is specified for both a JES2 or JES3 writer and an external writer, the two routines compete for data sets on a first-come, first-served basis.
- Print train, carriage, and, optionally, forms specification, are ignored when an external writer selects data set groups. As with the MVT and VS2 Release 1 output writer, FCB and UCS specification must be controlled through separating the data by class.

When there are no more data sets to select, the external writer will notify the operator by issuing a message, which also informs the operator of its current setup.

An external writer can dequeue data sets by any or all of the following characteristics: output class, job (job ID), forms, destination of LOCAL or remote workstation name, and installation-written writer name. Selection by local device name is not available in JES2. A characteristic, if not specified, is ignored when selecting data sets. (For example, the operator can set up an external writer to dequeue data sets for a certain installation-written writer name and a specific set of forms. Since job, class, and destination are not specified, any data set specifying the forms and writer is eligible regardless of its class or destination.)

For compatibility with the MVT and VS2 Release 1 output writer, the external writer will dequeue data sets by class and LOCAL destination, if one or more output classes are specified on either the START command or in the cataloged procedure used to call this external writer. Forms are mounted and installation-written writers are called on demand.

The IBM-supplied name STDWTR can be used by the operator for selecting data sets that have not specified any writer. Similarly, the forms name STD causes the external writer to select only those data sets that have not specified a particular form.

Starting the External Writer

The external writer is started by the operator issuing:

```
START or S
    procname[.id][,devicename][,volumeserial][,classes][,keyword=option]
```

where:

the procedure must reside in SYS1.PROCLIB.

classes

1-8 alphanumeric output classes which override the classes specified in the PARM field of the cataloged procedure used to start the External Writer.

keyword

Any DD keyword such as: DSN = datasetname [(member)], or LABEL = Any symbolic parameter.

Modifying the External Writer

The operator can change the options of the external writer by issuing:

```
MODIFY or F
    [procname.]id[,C[LASS]=[]list]],J[OBID]=[job id]][,F[ORMS]=form
    name]][,D[est]=[LOCAL or remote workstation name]][,W[riter]=[STDWTR
    or user written writer name]][,P[AUSE]=FORMS/DATASET]
```

Note: "id" plus one other parameter must be specified.

CLASS or C

= list
(1 to 8 classes in priority order) the external writer only dequeues data sets from these classes

= null
no selection by class (i.e. all data sets are eligible which meet the other criteria)

JOBID or J

= job
id select only data sets generated by the job with the specified id.

= null
select data sets from any job

FORMS or F

= form name
select only data sets requiring the specified form.

= null
disregard forms when selecting data sets. The operator will be instructed to mount forms as required.

DEST or D

= LOCAL
select only data sets routed to the central CPU (This is the default destination)

= remote workstation
name select only data sets routed to the specified remote workstation.

= null
select data sets irrespective of data set routing.

WRITER or W

= STDWTR
select only data sets not specifying a user-written writer

= user writer name
select only data sets enqueued for the specified user writer

= null
select data sets irrespective of writer required (The external writer attaches the writer program required for each data set.)

PAUSE or P

= FORMS
the external writer will pause whenever forms mounting is required (that is, FORMS = null is the option in effect.)

= DATASET
the external writer will pause between the processing of each data set.

Stopping the External Writer

The operator can stop the external writer by issuing:

```
STOP     [procname.]id
 or
 p
```

Cancelling the Printing of a Data Set

The data set being printed can be cancelled by issuing:

```
CANCEL  deviceaddress
  or
  C
```

The cancel command affects only the data set currently being written. MODIFY and STOP commands take effect between output for a given job.

The External Writer Cataloged Procedure

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

An EXEC statement named IEFPROC specifies the external writer program.

A DD statement named IEFRDER defines the output data set.

The standard external writer procedure supplied by IBM is named XWTR. The XWTR procedure is:

```
//IEFPROC   EXEC    PGM=IASXWR00,REGION=20K,
//                  PARM='PA'
//IEFRDER   DD      UNIT=2400,VOLUME=(,,,35),
//                  DSNAME=SYSOUT,DISP=(NEW,KEEP),
//                  DCB=(BLKSIZE=133,LRECL=133,BUFL=133,
//                  BUFNO=2,RECFM=FM)
```

When creating an external writer procedure, the procedure format and the statement requirements must be maintained. The IBM-supplied procedure can be used as an example. The statements are explained individually in the following sections.

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=IASXWR00,[REGION=nnnnnK,]
//                    PARM='cxxxxxxxx,seprname'
```

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

PGM=*IASXWR00*
specifies the output writer program. The name of the program must be IASXWR00, as shown.

REGION=*nnnnnK* (should be specified only for ADDRSPC=real)
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 requirement depends on the size of the buffers and the data set writer used. An insufficient size specification will result in an abnormal termination.

PARM='*cxxxxxxxx,seprname*'
is a set of parameters for the output writer program. The first part of this parameter field can contain from one to nine characters. The second part of this parameter field, if specified, is separated from the first part by a comma, and contains a program name from one to eight characters. Both parts of this parameter field are explained below.

c
an alphabetic character, either P (for printer) or C (for punch), that specifies the type of control characters for the output of the writer.

xxxxxxxx
from one to eight (no padding required) single-character class names for system output. These characters specify the type of output that the writer can process, and also establish the priority of the output classes, with the highest priority on the left. If class name parameters are included in the START command, they override this entire set of class names in the cataloged procedure. If no classes are specified in the cataloged procedure, and none are specified in the START command, the external writer waits for the operator to enter a MODIFY command before processing any output.

seprname
the name of the program (up to eight characters) that provides job separation in the output data set. The named program must reside in the link library (SYS1.LINKLIB) or the LPA library (SYS1.LPALIB). The name IEFSD094 specifies the output separator supplied by IBM, or the name of a user-written program can be specified. This subparameter may be omitted, in which case no output separator is used.

DD Statement

The procedure for the output writer must include a DD statement that defines the output data set. The format for this statement is:

```
//IEFRDER   DD      UNIT=device,LABEL=(,type)                    X
//                  VOLUME=(,,,volcount),                        X
//                  DSNAME=anyname,DISP=(NEW,KEEP)               X
//                  DCB=(list of attributes),                    X
//                  UCS=(code[,FOLD][,VERIFY]),                  X
//                  FCB=(image-id   {,ALIGN })
                                    {,VERIFY}
```

This DD name must be IEFRDER as shown. The parameter requirements are as follows:

UNIT=device
specifies the printer, magnetic tape, card punch, or direct access device on which the output data set will be written.

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

VOLUME=(,,,volcount)
limits the number of tape volumes that can be used by this writer during its entire operation (from the time it is started to the time it is stopped). This parameter is not required for printer or card punch devices.

DSNAME=anyname
specifies a name for the output data set (tape only, for label purposes), so that it can be referred to by subsequent job steps. This name is also necessary for specification of the KEEP subparameter in the DISP field.

DISP = (NEW,KEEP)
specifies the KEEP subparameter to prevent deletion of the output data set (tape and direct access only) at the conclusion of the job step.

DCB=(list of attributes)
specifies the characteristics of the output data set and the buffers. The BLKSIZE and LRECL subparameter fields must be specified in all cases. The BUFL subparameter field, if not specified, is calculated on the basis of the BLKSIZE value. Other subparameter fields may be specified as needed; if they are not, they will assume the QSAM default attributes which follow:

BUFNO — three buffers for the 2540 device, two buffers for all other devices.
RECFM — U-format, with no control characters.
TRTCH — odd parity, no data conversion, and no translation.
DEN — lowest density.

UCS = (code[,FOLD][,VERIFY])
specifies the code for a universal character set (UCS) image that will be loaded into the UCS buffer. FOLD causes bits 0 and 1 to be ignored when comparing characters between the UCS buffer and the print line buffer. This option allows lowercase alphabetic characters to be printed in uppercase by an uppercase print chain or train. VERIFY causes the specified UCS image to be printed for verification by the operator. The UCS parameter is optional, and is valid only when the output device is a 1403 or 3211.

FCB = (image-id [,{ALIGN}])
[,{VERIFY}]

causes the forms control buffer (FCB) image with the specified image-id to be loaded into the FCB. One of two optional parameters, ALIGN or VERIFY, can be coded. Either parameter allows the operator to align forms. In addition, VERIFY causes the specified FCB image to be printed for visual verification. The FCB parameter is valid only when the output device is a 3211.

For the processing of the output jobs that require special chains for printing, specific classes should be assigned for each different chain. The desired chain can be specified in the writer procedure, and when that writer is started the chain will be loaded automatically. (Printers used with special chains should be named with esoteric device names as defined at system generation time.)

The following sequence is an example of a writer-cataloged procedure for the P11 chain.

```
//IEFPROC   EXEC    PGM=IASXWR00,                                    X
//                  PARM='PDEG,IEFSD094'
//IEFRDER   DD      UNIT=SYSPR,DSNAME=SYSOUT,FCB=(STD2,ALIGN),       X
//                  UCS=P11,                                         X
//                  DISP=(,KEEP),DCB=(BLKSIZE=133,BUFL=133,          X
//                  LRECL=133,BUFNO=2,RECFM=FM)
```

If the output device is a 3211, a UCS or FCB image can be loaded dynamically between the printing of data sets. Therefore, a mixture of data sets using different images in a single output class is allowed; however, this may require mounting trains and changing forms, and may not be desirable. When the output device is a 1403, the UCS image is specified at START WTR time and cannot be changed until the writer is stopped; all data sets within an output class must be printed using the same train. This parameter cannot be overridden for a specific data set when using the (asynchronous) sysout writer. The FCB image is ignored when the 1403 is specified.

Writing an Output Writer Routine

The data set writer routine used for a data set can be specified by name (other than INTRDR or STDWTR) in a DD statement. If it is, the data set must be processed by an external writer. If a data set that does not specify an installation-written, or non-standard writer is processed by the external writer, a standard IBM-supplied writer routine is used. The standard routine transcribes the data set to the specified output device, making only those data format and control character transformations required to conform to the attributes specified for the output data set.

The following material describes how to write a nonstandard data set writer routine.

Characteristics of the Standard External Routine

Before writing or modifying an output writer routine, the functions performed by the standard data set writer should be understood. In general, these functions include opening the data set (referred to as an input data set) that contains the processed information, obtaining the records of the data set, making any necessary transformations in record format or control character attributes, and placing these (possibly transformed) records in the output data set, which appears on a specified output device. The standard writer also must close the input data set and restore system conditions to the state they were in before the writer routine was invoked. The external writer cannot be named STDWTR or INTRDR.

The Output Writer Routine

To use the output writer routine, the name of the routine should be specified as a parameter in the SYSOUT operand of a DD statement. The routine must be in the link library (SYS1.LINKLIB) or the LPA library (SYS1.LPALIB).

In VS2, the routine is attached (via the ATTACH macro instruction) when a data set requiring the routine is to be processed. The standard linkage conventions for attaching are used. Any storage required for work areas and tables should be obtained by the GETMAIN macro instruction and released by the FREEMAIN macro instruction. The output writer routines must be reenterable.

When the routine is finished, it must return control to the standard writer by using the RETURN macro instruction.

Parameter List: After job management routines perform initialization requirements and open the output data set into which the writer routine places records, control is given to the routine via the ATTACH macro instruction. At this time, general registers 1 and 13 contain information that the program must use. Register 1 contains the storage address of a 12-byte list. The information in this parameter list follows:

	Output Device	Indicator
Byte 0	Bit 0	(High-order bit): This bit should be off (set to 0).
	Bit 1	If this bit is on, the output unit is either a punch or a tape with a punch as the final destination.
	Bit 2	If this bit is on, the output unit is either a printer or a punch.
	Bits 3-7	No significant information.
Bytes 1-3	Not used, but must be present	
Bytes 4-7	This word contains the address of the data control block (DCB) for the opened output data set to be referred to by the writer.	
Bytes 8-11	This word contains the DCB address for the input data set from which the writer will obtain logical records. (At the time this 12-byte parameter list is given to the writer, the input data set is not open.)	

The switches indicated by the three high-order bit settings in byte 0 should be used to translate control character information from the input data set records to the form required by the output data set records. The high-order three bits of byte 0 signify the type of output device as follows:

011.....	2520 or 2540 punch unit
001.....	1403, 1443, or 3211 printer device
010.....	tape device with punch-destined output
000.....	tape device with printer-destined output

The writer should save and restore registers.

Programming Conventions: An output writer routine must issue an OPEN macro instruction to open the desired input data set residing on a direct access device as a result of the previous execution of a processing program. (Note: The output data set used by a writer is opened by a job management routine before control is given to the writer. This output data set must be given records by a PUT macro instruction operating in the "locate" mode.

If the processing program that produces a given data set (to be used as an input data set by a writer) did not open the data set, the data set contains no records, and the DCBBLKSI and DCBBUFL fields of the input DCB contain zero. The DCBBLKSI field may also be zero even if the data set does contain records — if the processing program did not put the block size value for the input set in the DCB. If both these DCB fields are zero, a value (the standard writer uses the decimal value 18) is inserted in the DCBBLKSI field to permit the open routine to continue. The standard writer does this via a routine pointed to by an entry in the EXLIST parameter of the DCB. Since there is no data set, nothing is put on the output device. The data set writer must provide a SYNAD routine to process errors associated with the output as well as the input data set.

The standard data set writer also includes accounting support for the SMF output writer record (record type 6).

Before the OPEN macro instruction is issued, the DCBD macro instruction can be used to symbolically define the fields of the DCB, and the EXLIST and/or SYNAD routine addresses can be inserted. Other than SYNAD, no modifications can be made to the output DCB.

After the routine finishes writing the output data set, it must close the input data set and return using the RETURN macro instruction. A return code must be placed in register 15. This code should indicate that an unrecoverable output error either has occurred (code of 8) or has not occurred (code of 0).

3525 Note — Interpret Punch: The programming support for the 3525 includes an INTERPRET PUNCH feature that is supported by BSAM and QSAM. The support for this feature includes the punching and printing of graphically printable punched characters on print lines one and three of the card. Line one includes the first 64 characters and line three includes the last 16 characters (right justified). Extraneous characters are printed for non-graphic eight-bit codes.

If the INTERPRET PUNCH function is designated via the FUNC parameter in either a DCB or DD statement, an existing output data set will be interpreted as well as punched.

Note: The output must be 80 bytes, or 81 bytes if first character control is being used.

Processing Performed by the Output Writer

Figure 30 provides a general description of the procedures followed by the standard writer. When writing a writer routine, items can be deleted, modified, or added to some of these procedures, depending on the characteristics of the data set(s). However, the procedures must be consistent with operating system conventions.

Saving Register Contents: Upon entering the writer program, the program must save the contents of the general registers, as previously discussed.

Obtaining Main Storage for Work Areas: In this work area, switches are established, record lengths and control characters are saved, and space is reserved for other uses. Storage is obtained by a GETMAIN macro instruction.

Processing Input Data Set(s): To process a data set, the writer must get each record individually from the input data set, transform (if necessary) the record format and the control characters associated with the record in accordance with the output data set requirements, and put the record in the output data set. Data set processing by the standard writer can be considered in three aspects.

1. The first consideration is what must be done before actually obtaining records from an input data set. If the output device is a printer, provision must be made to handle the two forms of record control character that may accompany a record in an output data set. The printer is designed so that if the output data set records contain machine control characters, a record (line) is printed before the effect of its control character is considered. However, if USASI control characters are used in the output data set records, the control character effect is considered before the printer prints a record.

   Thus, if all the input data sets do not have the same type of control characters, it may be desirable to avoid overprinting of the last line of one data set with the first line of the following data set. If the records of the input data set have machine control characters (mcc) and the output data set records are to have USASI control characters (acc), the standard writer produces a control character that indicates one line should be skipped before printing the first line of output data.

If the input data set records have acc and the output data set records are to be written with mcc, the standard writer prints a line of blanks before printing the first actual output data set record. Following this line of blanks, a one-line space is generated before the first output record is printed. The preceding "printer initialization" procedure (or a similar one based on the characteristics of the data sets) is recommended.

2. After an input data set is properly opened and any necessary printer initialization completed, the writer obtains records from the input data set. The locate mode of the GET macro instruction is used. As each record is obtained, its format and control character must be adjusted, if necessary, to agree with that required for output.

   Note: The MACRF field of the input data set DCB should be checked to see if GET in locate mode can be used. If not, the MACRF field must be overridden.

   Since the output data set is previously opened by another routine (job management), a writer routine must adhere to the established conventions. The data set is opened to receive records from the PUT macro instruction operating in the locate mode. For fixed-length record output, the length of the records in the output data set is obtained from the DCBLRECL field of the DCB. If an input record length is greater than the length specified for the records of the output data set, the standard writer truncates the necessary right-hand bytes of the input record. If the input record length is smaller than the output record length, the standard writer left-justifies the input record and adds blanks on the right end to give the correct length.

   When the output record length is variable and the input record length is fixed, the standard writer constructs each output record by adding control character information (if necessary) and variable record control information to the output record. The record control information is four bytes long and the control character information is one byte long. Both additions are made to the left end of the record. If the output record is not at least 18 bytes long, it is further modified by padding bytes (blanks) added to the right end of the record. If the output record length does not agree with the length of the output buffer, the standard writer makes the proper adjustment.

3. The third aspect is an end-of-input data set routine. The standard writer handles output to either a card punch unit or a printer unit, as required. Output to an intermediate device such as a tape unit is considered in light of the ultimate destination (e.g., punch or printer). If proper consideration is not given, all records from a given data set may not be available on the output device until the output of records from the next data set is started or until the output data set is closed. When the output data set is closed, the standard writer automatically puts out the last record of its last input data set.

Punch Output: Normally, when the standard writer is using a card punch as the output device, the last three output records are not in the collection pockets of the punch when the input data set is closed. To put out these three records with the rest of the data set and with no intervening pauses, the writer provides for three blank records following the actual data set records.

Printer Output: When the standard writer uses a printer as an output device, the last record of the input data set is not normally put in the output data set when the input data set is closed. To force out this last record, the writer generates a blank record that follows the last record of the actual data set.

The problem of overprinting the last line of one data set by the first line of the following data set must also be considered. Depending on the combination of input record control character and required output record control character, a line of blanks and a spacing control character may be used either individually or in combination to preclude overprinting. (*Note: If overprinting is desired for some reason, control characters in the data set records themselves may be used to override the effect (but not the action) of the previously described solutions to overprinting.)*

Closing Input Data Set(s): After the standard writer finishes putting out the records of an input data set, it closes the data set before returning control to the system output writer. All input data sets must be closed.

Releasing Main Storage: The storage and buffer areas obtained for the writer must be released to the system before the writer relinquishes control. The FREEMAIN macro instruction should be used for this.

Restoring Register Contents: The original contents of general registers 0 through 12, and 14 must be restored. The RETURN macro instruction is used for this. To inform the operating sytem of the results of the processing done by the writer, a return code is placed in general register 15 before control is returned. If the writer routine terminates because of an unrecoverable error on the output data set, the return code is 8; otherwise, the return code is 0. Unrecoverable input errors must be handled by the data set writer.

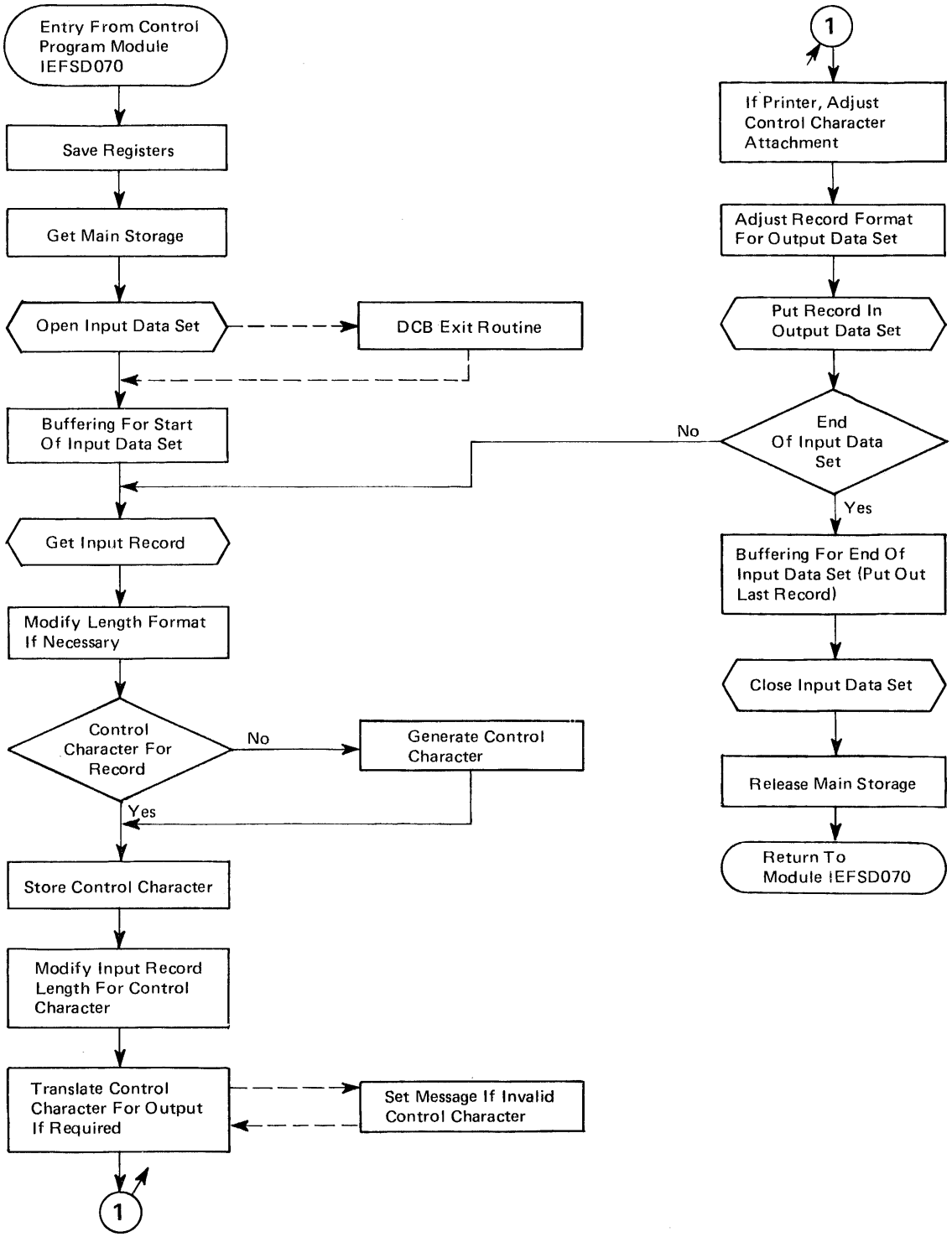

Figure 30. General Logic of Standard External Writer Routine

Output Separation

The External writer uses an 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.

Characteristics of an Output Separator

The external writer can be used by a problem program to channel its output eventually to a printer or punch. When this is done, however, the 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 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 between the writing of each section of a job's output.

For data processed by the external writer, the IBM output separator or the user's own output separator can be used.

The external writer standard separator function operates under control of the external writer. The external writer separator program must reside in the link library (SYS1.LINKLIB) or the LPA library (SYS1.LPALIB). Its name, IEFSD094, must be included as a parameter in the output writer procedure — the second part of the PARM field in the EXEC statement — to separate job output. (A cataloged procedure for the writer is fully described elsewhere in this chapter). The type of separation provided by the separator depends on whether the output is punch-destined or printer-destined.

Punch-Destined Output: The external writer provides three specially punched cards (deposited in stacker 1) prior to the punch 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

Printer-destined Output: The external writer provides three specially printed pages prior to printing the output of each job. Each of these three separator pages is printed in the following format:

- Beginning at the channel 1 location (normally near the top of the page), the jobname is printed in block character format over 12 consecutive lines. The first block character of the 8-character jobname begins in column 11. Each block character is separated by 2 blank columns.
- The next 2 lines are blank.
- The output classname is printed in block character format covering the next 12 lines. This is a 1-character name, and the block character begins in column 55.
- The remaining lines to the bottom of the page are blank.

In addition to the above, a full line of asterisks (*) is printed twice (overprinted) across the folds of the paper. These lines are printed on the fold preceding each of the three separator pages, and one the fold following the third page. This feature provides easy separation of job output in a stack of printed pages.

For *printer-destined output with the IBM-supplied separator, a channel 9 punch should be included in addition to the channel 1 punch on the carriage control tape or in the forms control buffer* (FCB). The channel 9 punch controls the location of the line of asterisks and should correspond to the bottom of the page. To print the line of asterisks on the fold of the pages, the printer registration should be offset.

Writing an Output Separator Program

The External writer separator program can be written by using the information provided by the External writer and by conforming to the requirements explained below. The separator program, when added to the link library (SYS1.LINKLIB) or the LPA library (SYS1.LPALIB), is invoked by specifying its name as a parameter in the EXEC statement of the output writer cataloged procedure.

Parameter List: The output writer provides the separator program with a 4-word parameter list of needed information. When the program receives control, register 1 contains the address of a 4-word parameter list, and the parameter list contains the following:

Bytes 0-3	In this word, byte 0 contains switches that indicate the type of output unit, and bytes 1-3 are reserved for future use.
Bytes 4-7	This word is the address of the output DCB (data control block).
Bytes 8-11	This word is the address of an 8-character field containing the jobname.
Bytes 12-15	This word is the address of a 1-character field containing the output classname.

In the parameter list, the three high-order bits of byte 0 are switches that the separator program uses to determine the type of output unit. The first bit to the left is set to 0. The second bit is set to 1 if the output unit is a punch device or a tape device with punch-destined output. The third bit is set to 1 if the output unit is a printer or punch device. The resulting bit combinations indicate the following:

011.	2520 or 2540 punch device
001.	1403, 1443, or 3211 printer device
010.	tape device with punch-destined output
000.	tape device with printer-destined output

The parameter list also points to the DCB for the output data set. This DCB is established for the queued sequential access method (QSAM), and is already open when the separator program receives control.

The address of the jobname and the address of the output classname are provided in the parameter list so that this information may be used in the separation records written by the separator program.

Programming Conventions: When using the External writer, the separator program, if specified in the External writer cataloged procedure, is brought in by a LINK macro instruction issued by the output writer. The separator program can be any size, but a program over 8K may affect the region requirement of the output writer.

Caution: Since the separator program operates with a privileged protection key, but in the program mode, the separator program must insure data protection during its execution.

When writing a separator program, the following programming conventions must be observed:

- The program must conform to the standard linkage conventions.
- The program must use the PUT macro instruction in the locate mode to write separation records on the output data set. (This method is required by the QSAM DCB that is open for the output data set.)
- The program must establish its own synchronous error exit routine, and the address of this routine must be placed into the DCBSYNAD field of the output DCB. This gives control to the error exit routine in case an uncorrectable I/O error occurs while writing the program's output.

- The program should use the RETURN macro instruction to return control to the output writer. Before returning, the program must free any main storage it obtained during its operation; and the program must place a return code (binary) in register 15. The return codes signify:
  0 — Successful operation.
  8 — Unrecoverable output error (should be set if the error exit routine is entered).

Output from the Separator Program

The separator program can write any kind of separation identification. The jobname and the output classname for each job are available through the parameter list for inclusion in the output, if desired. An IBM-supplied routine can be used that constructs block characters (explained later). As many separator cards can be punched or as many separator pages can be printed as necessary.

The output from the separator program must conform to the attributes of the output data set. These attributes, which can be determined from the open output DCB pointed to by the parameter list, are:

- Record format (fixed, variable, or undefined length)
- Record length.
- Type of carriage control characters (machine, USASI, or none).

For printer-destined output, the separation records should be begun on the same page as the previous job output, or any subsequent page should be skipped to. However, the separator program should skip at least one line before writing any records, because in some cases the printer is still positioned on the line last printed.

After completing the output of the separation records, the separator program should write sufficient blank records to force out the last separation record. This also allows the error exit routine to obtain control if an uncorrectable output error occurs while writing the last record. The requirements are:

- One blank record for printer-destined output.
- Three blank records for punch-destined output.

Using the Block Character Routine

For printer-destined output, the separator program can use an IBM-supplied routine to construct separation records in a block character format. This routine is a reenterable module named IEFSD095, and resides in the module library SYS1.AOSB0.

The block character routine constructs block letters (A to Z), block numbers (0 to 9), and a blank. The program furnishes the desired character string and the construction area. The block characters are constructed one line position at a time. Each complete character is contained in 12 lines and 12 columns; therefore, a block character area consists of 144 print positions. For each position, the routine provides either a space or the character itself.

The routine spaces 2 columns between each block character in the string. However, the routine does not enter blanks between or within the block characters. The program must prepage the construction area with blanks or other desired background before entering the block character routine.

To use the IBM-supplied block character routine, the separator program executes the CALL macro instruction with the entry point name of IEFSD095. Since the block characters are constructed one line position at a time, complete construction of a block character string requires 12 entries to the routine. Each time, the address of a 4-word parameter list should be provided in register 1. The parameter list must contain the following:

Bytes 0-3	This word is the address of a field containing the desired character string in EBCDIC format.
Bytes 4-7	This word is the address of a full word field containing the line count as a binary integer from 1 to 12. This represents the line position to be constructed on this call.
Bytes 8-11	This word is the address of a construction area in main storage where the routine will construct a line of the block character string. The required length in bytes of this construction area is 14n-2, where n represents the number of characters in the string.
Bytes 12-15	This word is the address of a fullword field containing, in binary, the number of characters in the string.

Index

Indexes to OS/VS2 publications are consolidated in the **OS/VS2 Master Index**, GC28-0693, and the **OS/VS2 Master Index of Logic**, GY28-0694. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

OS/VS2 System Programming Library:
Job Management
GC28-0627-0

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

Cut or Fold Along Line

What is your occupation? _____

Number of latest Technical Newsletter (if any) concerning this publication: _____

Please indicate your address in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

GC28-0627-0

OS/VS2 SPL: Job Management (S370-36) Printed in U.S.A. GC28-0627-0

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

OS/VS2 System Programming Library:
Job Management
GC28-0627-0

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

What is your occupation? _____

Number of latest Technical Newsletter (if any) concerning this publication: _____

Please indicate your address in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Cut or Fold Along Line

Cut or Fold Along Line

Your comments, please . . .

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold Fold

First Class
Permit 81
Poughkeepsie
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department D58, Building 706-2
PO Box 390
Poughkeepsie, New York 12602

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

OS/VS2 SPL: Job Management (S370-36) Printed in U.S.A. GC28-0627-0

OS/VS2 System Programming Library:
Job Management
GC28-0627-0

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

What is your occupation? _____

Number of latest Technical Newsletter (if any) concerning this publication: _____

Please indicate your address in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Cut or Fold Along Line

Cut or Fold Along Line

Your comments, please . . .

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold Fold

First Class
Permit 81
Poughkeepsie
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department D58, Building 706-2
PO Box 390
Poughkeepsie, New York 12602

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)