GC26-3784-1

Systems

OS/VS Checkpoint/Restart

VS1 Release 2
VS2 Release 1

IBM

Second Edition (July 1972)

This edition, as amended by technical newsletter GN26-0754, applies both to Release 2 of OS/VS1 and to Release 1 of OS/VS2, and to all subsequent releases of either system unless otherwise indicated in new editions or technical newsletters.

Significant system changes are summarized under "OS/VS1 Summary of Changes" or "OS/VS2 Summary of Changes" following "About This Book."

Information in this publication is subject to significant change. Any such changes will be published in new editions or technical newsletters. Before using the publication, consult the latest *IBM SRL Newsletter,* GN20-0360, that amends *IBM System/360 and System/370 Bibliography,* GA22-6822, to learn which editions and technical newsletters are applicable and current.

Requests for copies of IBM publications should be made to the IBM branch office that serves you.

Forms for readers' comments are provided at the back of the publication. If the forms have been removed, comments may be addressed to IBM Corporation, Programming Center—Publishing, Department D58, Monterey and Cottle Roads, San Jose, California 95114. All comments become the proper of IBM.

ABOUT THIS BOOK

This publication describes checkpoint/restart, a technique for recording information about a job at programmer-designated checkpoints so that, if necessary, the job can be restarted at the beginning of a step or at a checkpoint within a step.

The major parts of this publication and the information in them are as follows:

- Chapter 1 describes in general terms checkpoint/restart and its components.
- Chapter 2 describes how to establish a checkpoint.
- Chapter 3 describes the restrictions that must be observed when a checkpoint is taken or a restart performed on user data sets.
- Chapter 4 describes how to request restart.
- Chapter 5 describes what the operator must do to authorize restart.
- Chapter 6 contains storage estimates.
- Chapter 7 contains miscellaneous information about checkpoint/restart.
- Appendixes A and B list completion codes and describe how to establish checkpoint at end-of-volume.

Checkpoint/restart is intended for use by programmers and system analysts. A general understanding of job control language and data management is prerequisite knowledge for understanding the information in this book. See *OS/VS Job Control Language Reference,* GC28-0618, and *OS/VS Data Management Services Guide,* GC26-3783, for background information on these subjects.

The following publications are referred to in this book:

- *OS/VS Data Management Macro Instructions,* GC26-3793, which contains information about coding DCBs
- *OS/VS Data Management for System Programmers,* GC28-0631, which contains information about preallocated data sets
- *OS/VS1 Planning and Use Guide,* GC24-5090, which contains information about the RESERVE macro instruction and creating or modifying a list of resident modules
- *OS/VS2 Planning and Use Guide,* GC28-0600, which contains information about the RESERVE macro instruction and creating or modifying a list of resident modules.
- *OS/VS Supervisor Services and Macros,* GC27-6979, which contains information about the list and execute forms of the CHKPT macro instruction
- *OS/VS Tape Labels,* GC26-3795, which contains information about tape labels

OS/VS1 SUMMARY OF CHANGES

Release 2

- A return code of X'14' has been added to indicate EOV on a tape checkpoint data set. The method of recovering from this condition has also been changed.
- The checkpoint/restart work area has been enlarged by 16 bytes.

CONTENTS

Page

FIGURES

DIFFERENCES BETWEEN OS/VS CHECKPOINT/RESTART AND OS/MFT AND OS/MVT CHECKPOINT/RESTART

- The BLKSIZE parameter in the DCB macro instruction for the checkpoint data set need not be coded in VS.
- Module IGG019HT, a page-fix appendage, was added to the resident access method (RAM) list. IGG019HT is required when virtual storage has been specified for a task.
- If real storage is specified for the job when a checkpoint is taken, it should also be specified when the job is restarted.
- In VS1, the checkpoint/restart work area must begin and end on a 2K-byte boundary; the size of this work area has increased.
- In VS2, the user need not provide space in his region for the checkpoint/restart work area; the system will obtain the necessary space outside the user's region.

CHAPTER 1: INTRODUCTION

Checkpoint/restart is a technique for recording information about a job at programmer–designated checkpoints so that, if necessary, the job can be restarted at one of these checkpoints or at the beginning of a job step.

A checkpoint is taken when a user program issues the CHKPT macro instruction. This macro causes the contents of the program's virtual–storage area and certain system control information to be written as a series of records in a data set. These records can then be retrieved from the data set if the job terminates abnormally or produces erroneous output, and the job can be restarted. Restart can take place immediately (initiated by the operator at the console) or be deferred until the job is resubmitted. In either case, the time–consuming alternative of rerunning an entire job is eliminated.

Types of Restart

The checkpoint/restart program allows four types of restart:

- automatic step restart
- automatic checkpoint/restart
- deferred step restart
- deferred checkpoint/restart

Automatic restarts are initiated by the operator at the console. Automatic step restart, which is restart at the beginning of a job step, is requested in the job control language. Automatic checkpoint/restart, which is restart at the last checkpoint taken before the job failed, is requested in the CHKPT macro instruction.

Deferred restarts take place when a job is resubmitted to be run. Deferred step restart takes place at the beginning of the job step specified in the job control language. Deferred checkpoint/restart takes place at the checkpoint specified in the job control language.

Components of Checkpoint/Restart

CHKPT Macro Instruction

The CHKPT macro is coded in the user's program to cause a checkpoint to be taken. In addition, it may request automatic restart at the last checkpoint taken.

When a CHKPT macro is executed, the contents of the program's virtual–storage area and certain system control information are written, as a series of records, in a data set. The series of records is called a checkpoint entry, and the data set in which they're written is called a checkpoint data set. The checkpoint entry, which has a unique programmer–specified or system–generated identification called a checkid, is retrieved from the data set when restart occurs.

Chapter 2 explains in detail how to establish a checkpoint.

End–of–Volume Exit Routine

The end–of–volume exit routine is coded in the user's program to allow execution of the CHKPT macro instruction each time the processing of a multivolume physical sequential user data set is continued on another volume. Appendix B contains more detailed information about the end–of–volume exit routine.

RD (Restart Definition) Parameter

The RD parameter is coded in the JOB or EXEC statements and is used to request automatic step restart if job failure occurs and/or to suppress, partially or totally, the action of the CHKPT macro instruction. Chapter 4 contains more detailed information about this parameter.

RESTART Parameter

The RESTART parameter, coded in the JOB statement, is used when a job is resubmitted for restart (deferred restart). It specifies either the step (for deferred step restart) or the step and the checkpoint within that step (for deferred checkpoint/restart) at which restart should begin. Chapter 4 contains more detailed information about this parameter.

SYSCHK DD Statement

The SYSCHK DD statement is required when a job is being submitted for deferred checkpoint/restart. It defines the checkpoint data set for the job being restarted. Chapter 4 contains more detailed information about the SYSCHK DD statement.

CKPTREST System Generation Specification

The CKPTREST macro instruction specifies, at system generation, which of the completion codes accompanying abnormal step termination indicates that the step is eligible to be restarted. During system generation, a standard, IBM–defined set of system completion codes (codes emitted when the system executes ABEND) is placed in a table of eligible codes. The table becomes part of the control program. CKPTREST, which is optional, can be used to delete system completion codes from the table and to add user completion codes (codes emitted when the user's program executes ABEND) to the table. The syntax of the macro instruction is:

CKPTREST [NOTELIG = (*hex–code*(,*hex–code*)...)**][,ELIGBLE** = (*dec–code*(,*dec–code*)...)**]**

The NOTELIG operand can be used to delete any number of system completion codes from the table of eligible codes; *hex–code* is specified as a three–character hexadecimal number.

The ELIGBLE operand can be used to add up to ten user completion codes to the table; *dec–code* is specified as a decimal number having a maximum value of 4095.

If multiple codes are specified in either operand, the codes can be specified in any order.

The IBM–defined set of eligible system completion codes is listed in Figure 1.

001	106	2F3[1]	422	714	926
031	113	313	513	717	937
033	117	314	514	737	A14
03A	137	317	613	806	B14
0A3	20A	32D	614	813	B37
0B0	213	413	626	837	C13
0F3	214	414	637	906	E37
100	217	417	700	913	

1. Code 2F3 indicates that a job was executing normally when system failure occurred. The code is included in a console message displayed during system restart.

Figure 1. Standard Eligible System Completion Codes

Note: Whether or not the CKPTREST macro instruction is used, the SUPRVSOR macro instruction must be used to specify resident access methods. For details, refer to "Resident Access Methods" in Chapter 6.

If CALL IEHREST is used in PL/I programs, the CKPTREST macro instruction must specify 4092 as an eligible user completion code.

CHAPTER 2: HOW TO ESTABLISH A CHECKPOINT

This chapter explains how a user may establish checkpoints at which to restart job steps. The topics discussed are:

- CHKPT macro instruction
- Cautions in taking a checkpoint
- DCB for a checkpoint data set
- DD statement for a checkpoint data set
- Use of checkpoint data sets

CHKPT Macro Instruction

The CHKPT macro instruction is coded in the user's program. When the CHKPT macro is executed, job step information about the user's program, virtual–storage data areas, data set position, and supervisor control is written as a checkpoint entry in a checkpoint data set. The point at which this information is saved becomes a checkpoint from which a restart may be performed if the job terminates abnormally or the system fails. After the checkpoint entry is written, control returns to the user's program at the instruction following the CHKPT macro.

The CHKPT macro instruction refers to the data control block (DCB) for the checkpoint data set. The checkpoint data set can be opened for output before the CHKPT macro instruction is executed. If the data set is not open, the checkpoint routine opens it and then closes it after writing the checkpoint entry. If the data set is open, the checkpoint routine writes the checkpoint entry, but does not close the data set.

The checkpoint data set must be on one or more magnetic tape volumes or on one direct–access volume. A checkpoint data set can reside on a magnetic tape with IBM standard labels, nonstandard labels, or no labels. American National Standard labels cannot be used for a checkpoint data set.

The standard form of the CHKPT macro instruction is:

```
[symbol] CHKPT {dcb address[,checkid address  [,checkid length  ]]    }
                                              [,'S'              ]
               {CANCEL                                                 }
```

The operands are defined as follows:

dcb address
is the address of the DCB for the checkpoint data set.

CANCEL
cancels the request for automatic checkpoint/restart. Automatic step restart can occur if RD=R was specified. If CHKPT without **CANCEL** is then executed before abnormal termination, a request for automatic checkpoint/restart is again in effect. Checkpoint entries written before a CHKPT with **CANCEL** are left intact and may be used to perform a deferred checkpoint/restart.

checkid address

specifies the address of a programmer–provided field that is to contain a unique, printable identification of the checkpoint entry. The identification is called a checkid. The checkpoint routine writes the checkid as part of the entry and prints it in a message on the operator's console when it finishes writing the entry. The programmer must subsequently use the checkid by coding it in the JOB statement RESTART parameter if he wishes to use the corresponding entry to perform a deferred restart at a checkpoint. If the *checkid address* operand is omitted, the checkid length or **'S'** operand is invalid.

checkid length or **'S'**

Checkid length is the length in bytes of the field that contains the checkid. The maximum length of this field is 16 bytes when the checkpoint data set is physical sequential, 8 bytes when it is partitioned. (For a partitioned data set, the field can be longer than the actual checkid identification if the unused low–order portion of the field contains blanks.) By coding this operand or by omitting this operand entirely (in which case a length of 8 bytes is implied), the programmer specifies that his program will form an identification and store it into the checkid field before CHKPT is executed. If the *checkid address* operand is omitted, this operand is invalid.

By coding this operand as **'S'**, the programmer specifies that the checkpoint routine is to generate an identification 8 bytes in length and store it in the checkid field. If the *checkid address* operand is omitted, this operand is invalid.

Programming Notes on the CHKPT Macro Instruction

If both checkid address and checkid length or 'S' are omitted, the checkpoint routine generates an identification and writes it in the checkpoint entry and on the operator's console, but does not return it to the user's program.

If the programmer provides the checkpoint identification and the checkpoint data set is sequential, the identification can be any combination of up to 16 alphanumerics, special characters, and blanks. For a partitioned data set, it must be a valid member name of up to eight alphanumerics. The identification for each checkpoint should be unique. If two identifications differ only by having a different number of trailing blanks, the control program considers them to be the same.

A checkpoint identification generated by the checkpoint routine consists of the letter C followed by a seven–digit decimal number. The number, except in the case of a deferred step restart, is the total number of checkpoints taken by the job; it includes the current checkpoint, checkpoints taken earlier in the job step, and checkpoints taken by any previous steps of the job. When a deferred step restart takes place, this number is reset to 0.

The checkid address operand allows a user's program to select fields in the records of an input data set and use them as checkids. Alternatively, the user's program may use the checkid address and the 'S' operands and include a system–generated checkid in the current record of an output data set.

Exceptional Conditions

The CHKPT macro instruction returns a code in register 15 to indicate whether the CHKPT macro instruction was executed successfully. Appendix A contains a list of these codes and their meanings.

List and Execute Forms of CHKPT

The CHKPT macro instruction may be coded in the list and execute forms as well as in the standard form. The dcb address, checkid address, and checkid length operands can be coded in the list and execute forms; the CANCEL operand must not be coded.

A complete description of the list and execute forms of this macro instruction appears in *OS/VS Supervisor Services and Macros.*

Cautions in Taking a Checkpoint

The following discusses certain cautions that should be observed when taking a checkpoint. These cautions relate to the operation of certain macro instructions, serially–reusable resources, and special operating system features. Cautions that relate to user data sets are listed in Chapter 3.

Use of CHKPT With Other Macro Instructions

EXTRACT: The EXTRACT macro instruction is used to obtain information from the task control block (TCB). TCB information is subject to change when the task is terminated and the job step is restarted. If the information is needed after restart, the EXTRACT macro instruction should be reissued after the checkpoint is taken, as shown in Figure 2.

```
          .
          .
          .
          EXTRACT  ANSADDR,FIELDS=(ALL)   Obtain TCB information
          .
          .
          .
          CHKPT    CHKPTDCB               Establish checkpoint
          CH       15,=H'4'               Is restart in progress
          BNE      NRESTART               No, branch to NRESTART
          EXTRACT  ANSADDR,FIELDS=(ALL)   Yes, obtain new information
NRESTART
          .
          .
          .
```

Figure 2. Obtaining Updated TCB Information After Restart

SETPRT: The SETPRT macro instruction is used in data management to load the UCS buffer for a 3211 or 1403 Printer with the universal character set feature or the forms control buffer (FCB) for a 3211 Printer. The buffer contents are not saved when a checkpoint is taken. To reload the buffer upon restart, the user must reissue the SETPRT macro instruction.

WTOR: The reply to a WTOR macro instruction must have been received before CHKPT is issued.

STIMER: A time interval established by the STIMER macro instruction must have been completed before CHKPT is issued.

ATTACH: If ATTACH is issued in the program using CHKPT, all subtasks created must have terminated before CHKPT is issued; that is, the job–step task must be the only task of the step.

Use of CHKPT in Exit Routines

The CHKPT macro instruction must not be used in an exit routine other than the end–of–volume exit routine. The user may take a checkpoint when a BSAM or QSAM data set reaches end–of–volume.

Explicit and Implicit Requests for ENQ

When a job step terminates, it loses control of serially–reusable resources. If the step is restarted, it must request all of the resources needed to continue processing. Explicit use of a serially–reusable resource is requested when the user's program issues the ENQ macro instruction. If the program issues the ENQ and takes a checkpoint, it must issue the ENQ again whenever restart occurs at the checkpoint. Figure 3 shows a program that requests a serially–reusable resource by issuing an ENQ before establishing a checkpoint. After the checkpoint, it tests for a restart. If one has occurred, it requests the same resource again. It requests the resource again because the job step has terminated abnormally, has lost control of the resource, and has then been restarted from the checkpoint.

```
          .
          .
          .
          ENQ    (QADDR,RADDR)
          .
          .
          .
          CHKPT CHKPTDCB
          CH    15,=H'4'
          BNE   NRESTRT
          ENQ   (QADDR,RADDR)
NRESTRT
          .
          .
          .
          DEQ   (QADDR,RADDR)
          .
          .
          .
```

Figure 3. Request for a Resource After Restart

Some serially–reusable resources are requested implicitly by issuing data management macro instructions. These resources may be records that the user is processing or tracks on a direct–access device. To ensure correct processing, the user must not establish checkpoints while he has control of these resources:

- If the basic direct access method (BDAM) is used, the user's program must execute the WRITE or RELEX macro instruction to release a record that has been read with exclusive control, before executing the CHKPT macro instruction.

- If BDAM is used to add a record to a data set with variable–length or undefined records, BDAM issues an ENQ macro instruction for the capacity record (R0); therefore, the user's program must execute the WAIT or CHECK macro instruction to check completion of the write operation before it executes CHKPT.
- If the basic indexed sequential access method (BISAM) is used, a checkpoint must not be taken before waiting for completion of a write operation. If a record is read for update, a checkpoint must not be taken before writing the updated record and waiting for the write operation to be checked.
- If the queued indexed sequential access method (QISAM) is used, an ESETL macro instruction must be issued before taking a checkpoint if a SETL macro instruction was issued previously. Another SETL macro instruction may be issued after the checkpoint.
- Use of the RESERVE macro instruction (see "Shared DASD" caution below).

Use of Special Operating System Features

Shared DASD: At some installations, a direct–access storage device is shared by two or more independent computing systems. This device is a serially–reusable resource. If it is being used when a checkpoint is taken, it must be requested after a restart from the checkpoint. This resource is requested by a special macro instruction, RESERVE, described in the *OS/VS1 Planning and Use Guide* and the *OS/VS2 Planning and Use Guide.*

DCB For a Checkpoint Data Set

Required DCB Parameters

The programmer must provide a DCB for the checkpoint data set. (The publication *OS/VS Data Management Macro Instructions* contains detailed information about coding DCBs.) The following parameters must be included in this DCB:

- DOSRG=PS or PO (BSAM or BPAM data set organization)
- MACRF=W (WRITE macro instruction)
- RECFM=U or UT (undefined record format)
- DEVD=DA or TA (direct-access or tape device)
- TRTCH=C (data conversion with odd parity; parameter required only if the data set is on a 7-track magnetic tape)
- DDNAME= (name of DD statement for checkpoint data set)

The programmer must code the DSORG, MACRF, and DDNAME operands in the DCB macro instruction. He may code the RECFM, DEVD, and TRTCH operands in the DCB macro instruction, or he may code, in the related DD statement, the RECFM and TRTCH subparameters of the DCB parameter. Because RECFM and DEVD have default values of U and DA respectively, they need not be provided explicitly in either the DCB macro instruction or the DD statement. The LABEL parameter of the DD statement describes the labels of a data set on magnetic tape. For a checkpoint data set, the programmer can specify IBM standard labels (SL or SUL), nonstandard labels (NSL), or no labels (NL). American National Standard labels (AL or AUL) cannot be

specified for a checkpoint data set. If the label type is not specified, the operating system assumes that the data set has IBM standard labels.

DCB Options

The programmer may optionally provide the following DCB parameters:

- OPTCD=W (write validity checking)
- RECFM=UT (track overflow)
- NCP=2 (number of channel programs)
- NCP=2 and OPTCD=C (chained scheduling)

Notes on DCB

- The checkpoint routine does not use the BLKSIZE parameter; it writes all checkpoint records in 2048–byte blocks.
- Requests for two channel programs or chained scheduling apply only to the writing of virtual–storage records, not to the writing of control records or the reading of records for a restart. Because virtual–storage records are written directly from virtual storage without being buffered, the requests do not cause an increase in the work area used by the checkpoint routine.
- OPTCD=Q cannot be specified in the DCB.

DD Statement For a Checkpoint Data Set

The DD statement for the checkpoint data set must define the data set in a normal way. (*OS/VS Job Control Language Reference* contains detailed information on coding the DD statement.) The only restrictions on the statement are:

- The UNIT parameter must specify a tape or direct–access device supported by BSAM or BPAM. The device can be specified by referring to a specific device, a device type, or a group of devices. DEFER should not be coded in the DD statement.
- Secondary space allocation may be requested (by the increment subparameter), but it will not be used. (See "Notes on DD Statement.")
- The LABEL parameter must not specify ANSI tape labels.
- OPTCD=Q cannot be specified as a DCB subparameter.

Notes on DD Statement:

- The initial disposition of the data set (as specified in the DISP operand of the DD statement) is used in a normal way to position the checkpoint data set when it is opened, regardless of whether the user's program or the checkpoint routine executes the OPEN macro instruction. A more detailed discussion appears in the next section.

- The final and conditional dispositions of the data set have their normal meanings. However, if termination is occurring and an automatic restart at a checkpoint is to occur, the system automatically keeps all data sets that are in use by the job, including the checkpoint data set.
- If end-of-volume (no more primary space) is encountered while writing a checkpoint on a direct-access volume, two actions are possible:
  1. If the programmer requested secondary allocation, the allocation is performed, and the checkpoint routine issues return code 08. The allocated space is not used.
  2. If the programmer did not request secondary allocation, the system executes an ABEND macro instruction applying to the step. The ABEND causes emission of a D37 system completion code, which is not a code that makes the step eligible for restart. Thus, even though secondary space will not be used, secondary allocation should be specified to avoid abnormal termination.

Examples of DD statements for the checkpoint data set are:

```
//ddname   DD  DSNAME=dsname,UNIT=TAPE,DISP=(MOD,KEEP)
//ddname   DD  DSNAME=dsname,UNIT=SYSDA,DISP=(NEW,DELETE,KEEP),    X
//            SPACE=(TRK,(300,1)),VOLUME=SER=CKPTDS
```

Use of Checkpoint Data Sets

How Checkpoint Entries Are Written

If the user's program did not open the checkpoint data set before it executed the CHKPT macro instruction, the checkpoint routine opens it. The checkpoint entry is then written at a position determined by whether the data set is sequential or partitioned, and by the DISP parameter on the related DD statement. If the data set is sequential and its disposition is NEW or OLD, the checkpoint entry is written at the beginning of the data set. If the data set is sequential and its disposition is MOD, or if the data set is partitioned, the checkpoint entry is written after the last entry existing in the data set.

If the checkpoint data set is partitioned, each checkpoint entry is a member, and its checkid is its member name. After it writes a checkpoint entry, the checkpoint routine executes a STOW macro instruction to add the checkid of the entry to the directory of the data set. If an identical checkid already exists in the directory, the related address of a member is changed to be the address of the new checkpoint entry. The initial disposition specified for the checkpoint data set has no effect on the STOW operation.

If the checkpoint routine opens the checkpoint data set, it also closes it.

If the user's program opens the checkpoint data set for output, the checkpoint routine simply writes a checkpoint entry at the data set's current position and does not close the data set. If the user opens the checkpoint data set, he need not close it after taking the last checkpoint for the job step. If many checkpoints are taken, leaving the data set open will save time. All of the checkpoint entries will be saved in this case, thus providing the ability to request a deferred restart from any of the checkpoints. If the data set is partitioned, the checkpoint routine executes a STOW macro instruction as it would if it had opened the data set.

If end-of-volume is encountered during writing of a checkpoint entry on tape, a return code of 14 is placed in register 15 and the checkpoint is terminated. The system then issues a message to the operator explaining this unfortunate event and requests mounting of a new volume. The next checkpoint will be written entirely on the new volume. Only checkpoint data sets on tape may be multivolume data sets. The previous section, "DD Statement for the Checkpoint Data Set," discusses what occurs if end-of-volume (no more primary space) is encountered during writing of a checkpoint entry on a direct-access volume.

The status (open or closed) and position of a checkpoint data set remain the same at restart as they were after execution of the CHKPT macro instruction that established the checkpoint.

Note that a checkpoint data set must contain only checkpoint entries. A checkpoint entry must not be written in one of the user's data sets. Conversely, the program must not write its own data in a checkpoint data set. Note also that a checkpoint data set may not be a concatenated data set.

How to Ensure Restart

To ensure that restart at the most recent checkpoint will be possible, a checkpoint entry must not be written over a preceding checkpoint entry, because abnormal termination or system failure may occur while the new entry is being written. Three methods by which the programmer can ensure that restart will be possible is suggested below. All three methods involve the use of sequential checkpoint data sets.

Figure 4 shows the use of one sequential checkpoint data set, one data control block, and one DD statement (CHECKDD) specifying MOD disposition. The user allows the checkpoint routine to open and close the data set each time it writes a checkpoint entry. Checkpoint entries will be written sequentially in the data set. Performance would be improved if the user's program opened the data set and kept it open; the disposition could then be NEW or OLD.

```
        Program
            .
            .
            .
            CHKPT CHKDCB
            .
            .
            .
CHKDCB      DCB   DDNAME=CHECKDD,MACRF=W,DSORG=PS
            .
            .
            .
        DD Statement
//CHECKDD DD    UNIT=TAPE,DISP=(MOD,KEEP)
```

Figure 4. Using One Sequential Checkpoint Data Set to Ensure Restart

Figure 5 shows a way to alternate data sets when all checkpoints are taken by one CHKPT macro instruction. The data sets are opened by the control program and are identified by two DD statements, CHECKDD1 and CHECKDD2. The data control block initially refers to CHECKDD1. Before the second checkpoint, it is changed to refer to CHECKDD2; before the third checkpoint, it is again changed to refer to

CHECKDD1, and so forth. In this way, one data control block can be used for two data sets that are not open at the same time.

```
        Program
            .
            .
            .
            DCBD  DSORG=PS              Define IHADCB (dummy section
*                                       that defines DCBDDNAM)
            CSECT                       Resume original control section
            .
            .
            .
            LA    2,CHECKDCB            Establish CHECKDCB as base
            USING IHADCB,2              address for IHADCB
            XC    DCBDDNAM(8),DDHOLD    Exchange ddname in CHECKDCB for
            XC    DDHOLD(8),DCBDDNAM    ddname in DDHOLD
            XC    DCBDDNAM(8),DDHOLD
            CHKPT CHECKDCB              Open, checkpoint, close
            .
            .
            .
DDHOLD      DC    C'CHECKDD1'
CHECKDCB    DCB   DSORG=PS,MACRF=(W),DDNAME=CHECKDD2

        DD Statements

CHECKDD1 DD    UNIT=SYSDA,DISP=NEW
CHECKDD2 DD    UNIT=SYSDA,DISP=NEW
```

Figure 5. Using Two Sequential Checkpoint Data Sets to Ensure Restart

An alternate method of using two sequential data sets is to use two DCBs and two DD statements specifying NEW or OLD dispositions, and to execute alternately two CHKPT macro instructions, each referring to a different data set. Performance would be improved when using direct–access data sets if the user's program opened the data sets, kept them open, and used the POINT macro instruction to reposition them.

The method illustrated in Figure 4 saves all checkpoint entries for possible use in deferred restart, while the method illustrated in Figure 5 conserves auxiliary storage. Note that none of the methods requires use of a particular device type.

How Checkpoint Entries Are Identified

Any number of checkpoint entries can be written in a checkpoint data set, and any number of checkpoint data sets can be used concurrently. In a sequential checkpoint data set, checkids of valid or invalid checkpoint entries in one data set should be unique. In a partitioned data set, checkids of valid entries should be unique.

When the control program assigns identifications, the identification for each checkpoint is unique. The identification is 8 bytes in length and consists of the letter C followed by a seven–digit decimal number. This number, except in the case of a deferred step restart, is the total number of checkpoints taken by the job; it includes the current checkpoint, checkpoints taken earlier in the job step, and checkpoints taken by any previous job steps. When a deferred step restart takes place, this number is reset to 0.

If the programmer specifies checkids instead of having the system generate them, he may erroneously specify duplicate checkids. The system does not recognize this error. When deferred restart at a checkpoint occurs and the checkpoint data set is sequential, the system searches the data set from its beginning for the specified checkpoint entry. It uses the first entry it finds that has the specified checkid. If the data set is partitioned, the system searches the data set's directory to find the location of the specified checkpoint entry. If two or more entries having the same checkid were written in the data set, the most recent of those entries is the one pointed to by the directory, and restart occurs from the most recent entry.

Checkpoint entries have two identifications. The primary identification is the programmer–generated or system–generated checkid specified or requested by the CHKPT macro instruction. The secondary identification is identical to the system–generated checkid that might have been requested by CHKPT. The primary identification is used when a search is made for a checkpoint entry. The secondary identification is then used as a base to compute the system–generated checkids of entries written after restart has occurred. This procedure prevents the system from generating checkids that are duplicates of checkids of existing useful entries.

The control program identifies each checkpoint in a message to the operator; on request, it also makes the identification available to the user's program. In Figure 6, the CHKPT macro instruction requests the control program to supply an identification and place it in the 8–byte field named ID. When the checkpoint is successfully taken, the program prints the identification as part of a message to the programmer.

```
          .
          .
          .
          CHKPT CHKDCB,ID,'S'          Take checkpoint
          LTR   15,15                  Was checkpoint taken
          BNZ   PHASE2                 No, branch to PHASE2
          PUT   STEPLOG,MESSAGE        Yes, print checkpoint ID
PHASE2
          .
          .
          .
MESSAGE   DC    H'45,0'                Record length, etc.
          DC    C'SUCCESSFUL CHKPT AT PHASE2...ID='
ID        DS    CL8
STEPLOG   DCB   DSORG=PS,MACRF=(PM),RECFM=V,BLKSIZE=128,          C
                LRECL=124,DDNAME=LOGDD
CHKDCB    DCB   DSORG=PS,MACRF=(W),RECFM=U,DDNAME=CHKDD
```

Figure 6. Recording a Checkpoint Identification Assigned by the Control Program

How to Use the CANCEL Option

After being restarted, the job step may again terminate abnormally. If it does, it may again be restarted from the same checkpoint, subject to operator authorization. If the user wishes to avoid restarting the job step twice from the same checkpoint, the sequence shown in Figure 7 may be coded.

```
            .
            .
            .
            CHKPT   CHKPTDCB   Establish checkpoint
            CH      15,=H'4'   Is restart in progress
            BNE     NRESTART   No, branch to NRESTART
            CHKPT   CANCEL     Yes, cancel restart request
NRESTART    .
            .
            .
```

Figure 7. Canceling a Request for Automatic Restart

After the successful initiation of a checkpoint/restart, the system places a return code of 04 (hexadecimal) in general register 15 and returns control to the user's program at the instruction that follows the CHKPT macro instruction. At this time a request for another automatic restart at the same checkpoint is normally in effect. In Figure 7, the instruction that follows the CHKPT macro instruction tests the return code to determine whether control has been returned as the result of a restart. If the return code is 04, a restart has just occurred, and a second CHKPT macro instruction is executed. This macro instruction has a CANCEL operand, which cancels the existing request for an automatic restart. If the job step again terminates abnormally after a restart from the checkpoint, automatic restart can occur only at a later checkpoint. It will not occur at the checkpoint preceding the canceled checkpoint.

CHAPTER 3: USER DATA SETS

This chapter examines considerations in the handling of the user's data sets. The first part addresses considerations concerned with jobs that will be restarted at a checkpoint, the second, those considerations that apply to both step restart and checkpoint restart.

What to Consider for Checkpoint/Restart

The checkpoint routine records information about all data sets used by the step executing the CHKPT macro instruction. Recorded information includes:

- For all data sets, the information that can be coded on a DD statement, for example, device type and volume serial numbers. (The contents of the step's JFCBs are recorded.)
- For data sets open at the checkpoint, being processed on either magnetic tape or direct–access devices, and using the BSAM, QSAM, QISAM, BPAM, and EXCP access methods, the information needed to reposition the data sets if restart occurs at a checkpoint.

Cautions

- Data sets are repositioned at restart only if they were open when the checkpoint was taken. The Open routine will position normally all data sets opened after the checkpoint was taken.
- Unit record data sets are not repositioned (printer, punch, or card reader) at restart.
- If the programmer uses EXCP to process a tape data set open at a checkpoint, he should ensure that the block count in the data set's data control block is correct. If the block count is incorrect, the system may position the data set incorrectly when restart occurs.
- The system does not save and restore the contents of data sets. Therefore, the programmer must ensure that input data sets and system data sets contain all necessary data when restart occurs. If a data set on a direct–access volume is open at the checkpoint, the data set's label (the DSCB in the VTOC) must have the same location and reflect the same extents upon restart as it did when the checkpoint was taken. (See Chapter 4, the section on "Deferred Checkpoint Restart," and the subsection "JCL Requirements and Restrictions" Footnote 1.)
- When data set records are processed in an update–in–place manner (records are read, changed, and then written back into their original location in the data set), restart will be successful only if all records updated after the last checkpoint was taken are restored to their original state or if the user's program keeps track of the records that are updated and avoids updating them again during restart.
- If a checkpoint is taken and then a MOD data set (tape or direct–access) or a partitioned data set is opened, another checkpoint should be taken before any records are written into the data set. If the second checkpoint is not taken and restart occurs at the first checkpoint, the Open routine will position to the current end of the data set instead of to the original end.

Upon restart and after repositioning a partitioned data set opened for output (necessarily open at the checkpoint if it is to be repositioned), the system deletes member names from the data set's directory if the corresponding members are located in the data set at positions following the data set's current position.

If the user's program writes multiple members in a partitioned data set, it should take a checkpoint not only after it opens the data set, but also after each execution of the STOW macro instruction.

- Members may be deleted from a partitioned data set during a restart. If this action may delete members written by another job (another job may have been executed between the original and restart executions of the subject job), restart at a checkpoint should not be requested.
- When a step using the UCS (universal character set) feature is restarted, the system does not determine whether the UCS buffer is properly loaded, nor does it alert the operator to the UCS requirements of the step.
- If a checkpoint is taken, and then an output data set is extended onto a second direct–access volume (because end–of–volume occurred on the first volume and there was no more space available on the volume, or the data set contained 16 extents), and restart subsequently occurs at that checkpoint, the system does not delete the extension of the data set.
- Checkpoints should not be taken before an ISAM data set is opened in load mode. A checkpoint should be taken immediately after the data set is opened. Otherwise, an ABEND with a code of 03E will result from a restart at a previous checkpoint.
- If checkpoints are taken during loading of an ISAM data set using QISAM in load mode, a restart should not be attempted from one of those checkpoint if:
  - — insertions were made on the ISAM data set after it was loaded, and
  - — the insertions were made using the WRITE macro instruction with a type parameter of KN.
- When a job step is restarted from a checkpoint, the type of device allocated for the data set will depend on what was specified in the UNIT parameter of the DD statement:
  - — If a specific unit address was specified, for example UNIT=130, then the device with that address will be allocated.
  - — If a device type was specified, for example UNIT=2314, then a 2314 will be allocated.
  - — If a name for a mixed group of devices was specified, for example UNIT=SYSDA (which could include 2305s, 2314s, and 3330s), then a device of the same type as that used when the checkpoint was taken is allocated.
  - — If a name for a single type of device was specified, for example UNIT=DISK1 (where DISK1 was defined as a 2314), then a 2314 will be allocated.
- Tape data set repositioning during a checkpoint/restart under VS1 may severely degrade system performance if module IGC0S05B is not resident.

- Checkpoints may be taken with DOS tape files opened with the bypass leading tapemark option LABEL=(,LTM) and/or the bypass embedded DOS checkpoint records option DCB=(OPTCD=H) specified. However, a checkpoint must not be taken when an opened data set:
  - — resides on a DOS 7–track tape,
  - — is written in translate mode, and
  - — contains embedded checkpoint records
- An ISAM data set that is shared must be closed before a checkpoint is taken.
- A user who closes his SAM data set immediately after restarting his program at a checkpoint should be aware that the data set may not be restored to the same condition it was in when the checkpoint was originally taken.

Repositioning User Data Sets

The checkpoint routine records positioning information for user data sets as follows:

- *SYSIN and SYSOUT data sets.* The checkpoint routine waits until all requested input/output operations are complete. In VS1, the checkpoint routine then requests that the job entry subsystem (JES) save positioning information. In VS2, the checkpoint routine then records positioning information.
- *All other user data sets.* If input/output operations were requested but were not begun (for example, if a READ macro instruction was executed, but the related channel program was not started), the checkpoint routine stops any processing associated with the I/O request, records the positioning information, and then reestablishes I/O operations.

  If I/O operations have already begun, the checkpoint routine waits until they are complete before recording positioning information.

User data sets that were open at a checkpoint are repositioned upon restart to the positions existing at the checkpoint, except in the case of data sets on unit–record devices. Upon restart, writing of a data set on a printer or punch, or reading of a data set from a card reader, is simply resumed at the current position of the device.

When QSAM or QISAM is being used to process a data set, an indeterminate number of virtual–storage buffers may contain data when a checkpoint is taken. If restart at a checkpoint occurs, the system's action depends on whether a card reader or another type of device is being used to process the data set:

- *Card reader being used (QSAM only).* Upon restart, existing buffer contents are released. The buffers are reprimed by reading records from the current data set into them.
- *Another device being used.* Upon restart, the buffer contents are restored to virtual storage, and processing continues normally. Note that it is not possible to predict the time—either before or after the checkpoint—when a given record will be transferred between a buffer and the recording medium.

When a basic access method is being used to process a data set, processing resumes normally upon restart. If the user's program wants to ensure that a particular block is read or written before a checkpoint is taken, and if the data set is not SYSIN or SYSOUT, the program should complete the operation by executing the CHECK or WAIT macro instruction before it executes the CHKPT macro instruction. If the program does not complete the operation, the block may be read or written either before or after the checkpoint is taken.

When QSAM or BSAM is being used to read a data set from a card reader, the user's program can reposition the data set upon restart. If the user provides a repositioning routine, he should instruct the operator to position the data set to the beginning if a restart becomes necessary. The program might operate as follows:

- The program saves the first record read from the data set and keeps a count of the number of records read before each checkpoint.
- After a restart, the repositioning routine reads a record from the data set and compares it with the first record read before abnormal termination. If the records are identical, the data set has been positioned to the beginning. The routine then repositions it by reading (without otherwise processing) the number of records read before the checkpoint.

Preserving Data Set Contents

Update in Place: The control program repositions data sets but does not preserve their contents. After taking a checkpoint, the user must ensure that data set contents are not changed in a manner that will make successful restart impossible.

If the user's program reads records from a data set, updates them, and writes them back to their original locations, it may be useless to take a checkpoint before completing this processing. If a checkpoint is taken earlier, restart will be unsuccessful under these circumstances:

- The user's program updates a record before abnormal termination and repeats the update after restart, and
- The updated record contents depend on the original contents.

For example, suppose that the purpose of the update is to switch the positions of two fields in each record. If the record is updated twice, the fields are returned to their original positions, and the results are invalid. In a different application, an update might simply place a value in a record field, regardless of the field's original contents. The user could then restart the step at a checkpoint taken before or during the update procedure, because an updated record would not be changed if updated again after restart.

Updating a PDS: When a partitioned data set is updated, the user must be careful to preserve the contents of the directory. The directory consists of entries that point to each member of the data set.

When a member is added to a partitioned data set, an entry is also added to the directory. If one member is added, the STOW macro instruction may be used to create the entry, or the member name may be specified in the DD statement; in the latter case, the control program creates the directory entry when the data set is closed or when the job step terminates. If more than one member is added, the STOW macro instruction must be used to create an entry for each member.

When one or more members are added to a partitioned data set, a checkpoint must be taken immediately after opening the data set. After taking the checkpoint, the new member may be written and its entry added to the directory. Then, if the step is restarted from the checkpoint, the data set is repositioned; the new member and its directory entry are deleted and are recreated after restart.

To update a member of a partitioned data set, updated records may either be written back to their original locations, or the entire member (in updated form) may be rewritten as a new member of the data set. In the latter case, the directory entry must be updated to point to the rewritten member.

If a checkpoint is taken before rewriting an entire member, one must also be taken immediately after updating the directory, because the control program will delete the updated directory entry if it repositions the data set for restart from the original checkpoint. Since no entry then points to the original member, the restart will be unsuccessful.

Work Data Sets: Many programs use "work" data sets, which are alternately written and read, rewritten and reread. If a work data set is used, a checkpoint should be taken each time the user has finished reading the data set and before rewriting it.

For example, a program may perform the following sequence of operations to produce different versions of data set A:

1. Write and then read back A1.
2. Write and then read back A2.
3. Write and then read back A3.

A checkpoint should be taken at the very beginning of operations 2 and 3 before any rewriting of data set A takes place. If, for example, the job step is abnormally terminated while operation 2 is in progress, the job step can be restarted from the checkpoint taken at the beginning of operation 2. At this checkpoint there is no need for the data in version A1.

Nonstandard Tape Labels

If tapes with nonstandard labels are used and steps are to be restarted at a checkpoint, the user must provide a routine to process nonstandard labels at restart time. This routine need only perform input header label processing, because output tapes will contain the header labels that were written when the data sets were opened (prior to checkpoint).

At restart time, the control program checks the tape to make sure that the first record is not a standard volume label. If the first record is 80 bytes in length and contains the identifier VOL1 in the first four bytes, the tape is not accepted. The control program issues a message directing the operator to mount the correct tape.

When it is determined that the tape does not contain a standard volume label, the control program's Restart routine gives control to the user's routine for processing nonstandard labels. When this routine receives control, the tape has been positioned at the interrecord gap preceding the nonstandard label (the tape has been rewound).

If the user's routine determines that the wrong volume is mounted, a 1 must be placed in the high–order bit position of the SRTEDMCT field of the unit control block (UCB), and control is returned to the control program. The control program then issues a message directing the operator to mount the correct volume. When the new volume is mounted, the control program again checks the initial label on the tape before giving control to the user's routine.

Before returning control to the control program, the user's routine must position the tape at the interrecord gap that precedes the initial record of the appropriate data set. This applies to both forward and backward read operations. The control program then uses the block count shown in the data control block to reposition the tape at the appropriate record within the data set. This positioning is always performed in a forward direction. If the block count is zero, or a negative number, the control program does no positioning. (If the user wants the control program to reposition the tape during a restart, his normal header label routines—Open and EOV—must properly initialize the block count field of the data control block during the original creation. The block count field of the data control block must not be altered at restart time.) For additional information about tape labels, refer to *OS/VS Tape Labels.*

Input/Output Errors

The checkpoint routine issues return code 0C if it encounters a permanent input/output error during quiescing of queued access method input/output operations or during writing of the checkpoint data set. An exception occurs when QSAM is being used and the skip or accept option is specified in the EROPT parameter of the data set's data control block. In this case, code 00 is returned.

When an access method other than QSAM or QISAM is used, the user's program can ensure that input/output operations are complete before it executes the CHKPT macro instruction, and it can thereby avoid having read or written an erroneous record while quiescing.

If a permanent error occurs when the system reads a checkpoint data set to perform a restart, the restart step is terminated abnormally with the system completion code 13F. Further automatic restart of the step is not attempted.

What to Consider for Checkpoint or Step Restart

Generation Data Sets

The control program of the operating system allows a generation data set to be created in one step of a job and then referred to in a later step by the relative generation number used to create it. For example, a data set can be created in one step as the +1 generation and read in a following step also as the +1 generation, instead of as the 0 generation. The same relative generation number can be used because the system records in an internal table, called the bias table, the number of generation data sets created in each generation data group used by the current job. When the job uses a relative generation number to refer to a generation, the system subtracts the bias value from the specified number to determine the actual number of the desired generation.

If a generation data set is to be referred to later by a relative generation number, the DD statement used to create the generation data set must cause cataloging of the data set at the end of the step creating the data set. The programmer may use a conditional disposition to prevent cataloging at the end of a step that terminates abnormally.

The bias table is updated at the end of the step that creates a generation data set, whether or not the step terminates normally, and whether or not cataloging occurs. (This method of updating must be considered if a step is executed after abnormal termination and refers to a generation data set.) However, the bias table is not updated if automatic step restart or restart at a checkpoint is occurring, nor does cataloging occur in this case. Because the original bias table is used when an automatic restart occurs, generation data sets can be referred to during the restart exactly as they could be during the original execution.

If a deferred step restart is performed, the bias table contents existing during the original execution do not exist during the restart. Therefore, generation data sets created and cataloged during the original execution, in steps preceding the restart step, must be referred to during the restart execution by their actual relative generation numbers. Conditional dispositions should be used during the original execution to delete generation data sets created by the restart step. When a checkpoint is taken, the system records in the checkpoint entry the bias table contents existing at the beginning of the current step. These contents are restored to the bias table if a deferred restart at a checkpoint is performed. Conditional dispositions are used during the original execution to keep (instead of to catalog) generation data sets created by the restart step. Data sets, and generation data sets created and cataloged in steps preceding the restart step, can be referred to during the restart in the same way as they could be originally.

Preallocated Data Sets

In VS2, direct–access space for temporary data sets can be preallocated to save time in scheduling job steps. This facility, however, cannot be used with checkpoint/restart. Checkpoints and automatic restarts are suppressed for any job step that uses a preallocated temporary data set.

SYSIN Data Sets

When restart at a checkpoint occurs, a SYSIN data set (data following a DD * or DD DATA statement) is repositioned. Unit record data sets are never repositioned. When automatic restart is occurring, the system keeps the direct–access data sets that contain the SYSIN data of the job being restarted. During the restart execution, the job can read data from the direct–access data sets as it could during the original execution.

To perform deferred restart, the programmer includes any necessary SYSIN data in the resubmitted deck. In VS2, if the restart is to be at a checkpoint, and a SYSIN data set was open and not completely read at the checkpoint to be used, the attributes of the direct-access data set (into which the system will write the SYSIN data) must be the same as the attributes of the direct-access data set used originally. (The location and number of extents in the data set used during restart need not be the same.)

Information about altering SYSIN data in a restart deck is given in Chapter 4, the sections "Automatic Restarts" and "Deferred Checkpoint Restart," and the subsection, "JCL Requirements and Restrictions." Information about repositioning data sets during checkpoint/restart is given earlier in this chapter under "Repositioning User Data Sets."

SYSOUT Data Sets

The following discussion is about how SYSOUT data sets (data sets having the SYSOUT parameter coded on their DD statements) are handled during the various types of restart.

Automatic Restart

1. With direct system output (DSO), the user's program writes SYSOUT data directly onto a printer, card punch, or magnetic tape unit. None of these devices is repositioned during restart; therefore, data written during the restart execution does not overlay any of the data written during the original execution. All data written during the original and restart executions is printed or punched and made available to the programmer.
2. Without direct system output (DSO), the user's program writes SYSOUT data into one or more direct–access data sets. If step restart is occurring, the direct–access data sets used during the original execution have been deleted; therefore, new direct–access data sets are allocated when the restart step is reinitiated.

   If checkpoint/restart is occurring, the data sets used during the original execution are kept. Then if a SYSOUT data set was open when the last checkpoint was taken, it is repositioned to its position at that time. Data written during the restart execution overlays only the data written between the time the last checkpoint was taken and the time the job step terminated abnormally. If a SYSOUT data set was closed at the checkpoint, the data set is not repositioned. If the restart step opens the same data set again, the data written during the restart follows the data written originally. (The data set has implied MOD disposition.)

Deferred Checkpoint/Restart

1. When checkpoint/restart occurs, and a SYSOUT data set is open at the checkpoint, the data set written into during the restart is different from the data set used originally. The system writes data set header labels and job separators at the beginning of the data set used during the restart. Header labels are written only for direct system output (DSO) on tape. Data written by the restart step follows the job separators.
2. To perform a deferred checkpoint/restart of a step in which a SYSOUT data set was open at the checkpoint, direct system output (DSO) must be used for each data set for which it was used originally, and the device type must be the same.

Information about repositioning data sets during restart at a checkpoint is given earlier in this chapter in "Repositioning User Data Sets."

SYSABEND Data Sets

Whether or not the checkpoint/restart facility is used, abnormal termination will cause the system to write a SYSABEND (or SYSUDUMP) data set if the programmer provides a SYSABEND (or SYSUDUMP) DD statement. The system uses its own data control block to write the data set, and it opens the data set during abnormal termination processing. The programmer may either code or omit the SYSOUT parameter on the SYSABEND DD statement.

When the SYSOUT parameter is coded and automatic restart occurs after abnormal termination, the SYSABEND or SYSUDUMP data set will not be printed for step restart without direct system output (DSO). Because the SYSABEND or SYSUDUMP data set was created by the job step, it is deleted during restart.

In all other cases, the SYSABEND or SYSUDUMP data set is printed, whether or not the restart is successful. If a second abnormal termination occurs, a second SYSABEND or SYSUDUMP data set is written. The second data set is always printed, assuming that a second restart does not occur. If a second restart does occur, the second data set is printed except as described above.

CHAPTER 4: HOW TO REQUEST RESTART

This chapter explains how a user may request restart. The topics discussed are:

- RD (restart definition) parameter
- Restart parameter
- SYSCHK DD statement
- Automatic restart
- Deferred step restart
- Deferred checkpoint/restart

RD (Restart Definition) Parameter

The RD parameter is coded in the JOB or EXEC statements and is used to request that an automatic step restart be performed if failure occurs and/or to suppress, partially or totally, the action of the CHKPT macro instruction. If the RD parameter is used simply to request that an automatic step restart be performed if failure occurs, or if the RD parameter is not coded, the action of CHKPT is normal. (CHKPT writes a checkpoint entry and requests a checkpoint/restart to be performed if failure occurs.)

When coded on an EXEC statement, the RD parameter applies to the step corresponding to the statement or to all steps of the cataloged procedure referred to by the statement. When coded on a JOB statement, the RD parameter applies to all steps of the corresponding job and overrides an RD parameter coded in any EXEC statements of the job. The parameter syntax is:

RD[.*procstepname***] = {R/NC/NR/RNC}**

The possible definitions are:

RD = R
(Restart) Requests an automatic step restart to be performed if failure occurs. If the CHKPT macro instruction is executed in the step, the resulting request for an automatic checkpoint/restart overrides the request for an automatic step restart.

RD = NR
(No Automatic Restart) Does not request an automatic step restart, and suppresses the request for an automatic checkpoint/restart that would otherwise be made when the CHKPT macro instruction is executed in the step. If CHKPT is executed, it writes a checkpoint entry normally. The checkpoint entry can be used to perform a deferred restart.

RD = NC
(No Checkpoint) Does not request an automatic step restart, and totally suppresses the action of the CHKPT macro instruction if the macro instruction is executed in the step. This allows a program containing CHKPT to be used when the action of CHKPT is not wanted.

RD = RNC
(Restart and No Checkpoint) Requests an automatic step restart to be performed if failure occurs, and totally suppresses the action of CHKPT if CHKPT is executed in the step.

If RD= *value* is coded on an EXEC statement that invokes a cataloged procedure, the parameter applies to all steps of the procedure and overrides all RD parameters present in the EXEC statements of the procedure. RD.procstepname= *value* can be coded instead of RD= *value*; it applies to the specified procedure step and overrides the RD parameter that may be coded on the EXEC statement of the procedure step. RD.procstepname= *value* can be coded once for each step of the procedure.

RESTART Parameter

The RESTART parameter is used to perform a deferred restart of a job. It is coded in the JOB statement when the job is resubmitted. If step restart is to occur, this parameter is used to specify at which step to begin. If the restart is to occur at a checkpoint that was taken during a step, both the step and the identification of the particular checkpoint entry are specified. The syntax of the parameter is:

```
RESTART = ( {stepname                 }[,checkid])
          {stepname.procstepname      }
          {*                          }
```

Both operands are used if restart at a checkpoint is to occur. If a step restart is to occur, checkid must be omitted; the enclosing parentheses may be omitted.

The stepname parameter is coded as stepname.procstepname if a step of a cataloged procedure is to be restarted. The parameter can be coded as * if the first step of the job (possibly a step of a cataloged procedure) is to be restarted.

The checkid can contain up to 16 characters in any combination of alphameric characters, printable special characters, and blanks. If it contains any special characters or blanks, it must be enclosed in single apostrophes, and apostrophes within it must be represented as double apostrophes.

SYSCHK DD Statement

The SYSCHK DD statement is used in the resubmitted job to perform a deferred checkpoint/restart and specifies the checkpoint data set that contains the checkpoint entry to be used in the restart. The statement may not be included when a deferred step restart is to be performed. The statement is not needed when an automatic restart at the last checkpoint occurs, because in that case, the system knows the identity and location of the checkpoint data set. (Another DD statement describing the checkpoint data set is always included if the program executes the CHKPT macro instruction.)

The statement must immediately precede the first EXEC statement in the deck that is submitted to perform a deferred restart at checkpoint. It must follow the JOBLIB DD statement if the JOBLIB DD is present. The SYSCHK DD must describe the checkpoint data set that contains the checkpoint entry to be used to perform the restart. The desired checkpoint entry must be named by the checkid subparameter of the JOB statement RESTART parameter.

The following requirements and restrictions apply to the SYSCHK DD statement:

- The statement must contain or imply DISP=(OLD, KEEP).
- The statement must define the checkpoint data set in a normal way. For example, it must specify its name, device type, and volume serial number. The catalog may be used.
- If the checkpoint data set is multivolume, the SYSCHK DD must specify, as the first volume of the data set, the volume containing the desired checkpoint entry. The serial number of the volume containing a particular entry is shown in the console message that is written when the entry is written.
- If the checkpoint data set is on a 7–track magnetic tape having nonstandard labels or no labels, the SYSCHK DD must contain DCB=TRTCH=C.
- If the checkpoint data set is partitioned, the DSNAME parameter on the SYSCHK DD must not contain a member name.
- If a RESTART parameter without the checkid subparameter is included in a job, a SYSCHK DD must not appear before the first EXEC statement of the job.
- If a RESTART parameter is not included in a job, a SYSCHK DD appearing before the first EXEC statement in the job is ignored.
- A SYSCHK DD appearing in a step or procedure step of a job is treated as an ordinary DD statement; that is, the name SYSCHK has no special meaning in that case.

An example of a SYSCHK DD statement is:

```
//SYSCHK DD       DSNAME=dsname,DISP=OLD,UNIT=name,                X
//                VOLUME=SER=volser
```

Automatic Restarts

Because automatic step restart and checkpoint/restart are similar in many ways, they are discussed together, where possible, in the information that follows.

Requirements for Automatic Restart to Occur

Automatic step restart or checkpoint/restart will occur if all of the following conditions are met:

- The step requests restart.
- The step is eligible for restart because it was terminated by an ABEND macro instruction that emitted an eligible completion code (specified by the CKPTREST macro instruction), or because system failure occurred.
- The operator authorizes the restart. This authority enables the operator to prevent repeated restarts of the same step or at the same checkpoint.

How to Request Automatic Step Restart

If a step fails when automatic step restart is requested, restart occurs automatically at the beginning of the step that failed.

Automatic step restart is requested by coding the RD parameter (RD=R or RNC) on either the JOB or EXEC statement in the originally submitted job deck. The CHKPT macro instruction is suppressed if RD=RNC.

Figure 8 illustrates a job requesting automatic step restart.

```
//MYJOB    JOB    MSGLEVEL=1,[1] RD=R Requests automatic restart at
*                                     the beginning of any step
*                                     that terminates abnormally
//STEP1    EXEC
//STEP2    EXEC   RD=R[2]             Requests automatic restart of
*                                     STEP2 if it terminates
*                                     abnormally
//STEP3    EXEC
```

1. MSGLEVEL = 1 need not be coded for VS1.
2. Note that if RD = R appears on the JOB statement, it is not required on the EXEC statement.

Figure 8. Requesting Automatic Step Restart

How to Request Automatic Checkpoint/Restart

If a step fails and automatic checkpoint/restart is requested, restart occurs automatically at the last checkpoint taken.

Execution of the CHKPT macro instruction requests this type of restart and establishes the checkpoint. The user must provide an ordinary DD statement for the checkpoint data set.

RD=R may be omitted or included. If it is included and the step fails before or during the time when the first checkpoint is taken, an automatic step restart will occur. Automatic step restart will also occur when RD=R is coded if the last execution of the CHKPT macro instruction specified that a request for checkpoint/restart should be canceled.

Figure 9 illustrates a job requesting automatic restart at a checkpoint.

```
//MYJOB    JOB    MSGLEVEL=1[1]
//STEP1    EXEC
           .
           .
           .
//STEP2    EXEC   PGM=MYPROG    MYPROG issues the CHKPT macro
//NAME1    DD     DSNAME=NAME2  Describes the data set into which
*                                checkpoint entries are to be
*                                written
```

1. MSGLEVEL = 1 need not be coded for VS1.

Figure 9. Requesting Automatic Checkpoint/Restart

JCL Requirements and Restrictions

To allow occurrence of an automatic step restart or automatic checkpoint/restart, the programmer must observe the following rules when he prepares the job deck used in the original execution:

- If a step restart is desired, the RD parameter must be coded to request the restart.

- If a checkpoint/restart is desired, a DD statement for the checkpoint data set must be included in the step that executes the CHKPT macro instruction.
- The EXEC statements in the job deck must have unique names. (Upon restart, the system searches for a named step.)

In VS2, MSGLEVEL=1 must be coded on the JOB statement. (MSGLEVEL=1 produces internal records that are reinterpreted when restart occurs.)

- If commands are included in the original deck, the commands are not reexecuted when restart occurs.
- If a procedure used in the restarting step is in a private library other than SYS1.PROCLIB, the Restart Reader Procedure (IEFREINT) must be modified to indicate the private library.
- If an automatic restart is done in VS2, the default values for SYSOUT data sets will be UNIT=SYSDA,SPACE=(TRK,(50,100)).

Resource Variations Allowed in Automatic Restart

The system's device and volume configuration during a restart execution of a job can be different from what it was during the original execution of the job.

The ability to use a different volume usually exists only in the case of a new data set on a nonspecific volume. Furthermore, if a checkpoint/restart is to be performed, the data set must not have been open at the checkpoint. The ability to use a different device does not apply to the device or devices containing the SYSRES volume and the SYSJOBQE and LINKLIB data sets. Also, if a checkpoint/restart is to be performed, the same type of device must be allocated to the data set during both the original and restart executions.

How the System Works at Automatic Restart

How Data Set Disposition is Determined: When a step requests restart and is eligible for restart, disposition processing of the data sets used by the step or by the job does not occur until the operator has replied to the request for authorization. If the operator denies restart, disposition processing occurs normally; that is, programmer–specified final or conditional dispositions are performed and if the programmer requested that a step be executed after abnormal termination, the step is executed. If the operator authorizes automatic restart, the following special disposition processing is performed:

- If step restart is to occur, all data sets having OLD or MOD dispositions in the restart step and all data sets being passed around the restart step are kept, even if they have been declared to be temporary. Temporary data sets normally cannot be kept.
- All data sets having NEW dispositions in the restart step are deleted.
- If checkpoint/restart is to occur, all data being used by the job (data sets that were not previously disposed of) are kept.

If the operator authorizes restart, execution of the step to be executed after abnormal termination will not occur because, in effect, abnormal termination did not occur.

If the operator performs an operator–deferred restart by replying HOLD to the request for authorization, he later may issue a CANCEL command for the job instead of a RELEASE command. If he issues CANCEL, no further data set disposition processing or step executions will occur. Thus, the disposition of these data sets remains as it was when the HOLD was issued.

How the Job Deck Is Reinterpreted and the Input Work Queue Merged: When it has completed disposition processing for a terminated job that is to be restarted, the system begins the restart by interpreting the job deck again. The system uses its internal records of the job, and the job is not read again.

After it has reinterpreted the job deck, the system merges information from the newly formed input work queue entry for the job (on the scheduler work area data set in VS1, on SYS1.SYSJOBQE in VS2) into the original input work queue entry; then it destroys the newly formed entry. The system inserts a special step before the restart step in the job. The special step, named IEFDSDRP, is executed first; it reads the last checkpoint entry and merges information from it into the original input work queue entry.

When the information is merged, and if a step restart is occurring, the input work queue entry is the same as it was before the original initiation of the restart step. If checkpoint/restart is occurring, the input work queue entry differs from its original form in these ways:

- Data sets specified as NEW in the restart step have had their dispositions changed to OLD, except in the case of data sets that were not opened during the original execution and for which nonspecific tapes were requested.
- In the case of data sets for which nonspecific volumes were requested in the restart step, the work queue entry describes the device type and serial numbers of the volumes assigned to the data sets during the original execution.
- In the case of multivolume data sets, the work queue entry indicates which volumes were being processed at the checkpoint. These volumes, and not the first volumes of the data sets, will be mounted (if they have not remained mounted) during the restart.

How Step Restart Is Initiated

A step being restarted is initiated in the same way as it would be during a normal execution. Therefore, the devices allocated to the restart step can be different (but of the same device type) from the devices allocated originally. If the allocated devices differ, volumes must be moved from one device to another. If AVR is used, devices containing the required volumes are allocated, if the devices are available for allocation.

After devices have been allocated to the restart step, normal mounting messages request the operator to mount the required volumes on the devices, unless the volumes are already mounted. The volumes requested are those on which processing is to be resumed.

How Checkpoint/Restart Is Initiated

If checkpoint/restart is occurring, the restart step must be executed in the virtual–storage area that was used during the original execution. If the required virtual storage is allocated to another step before it is reallocated to the restart step, the restart is delayed until the other step terminates.

In VS1, the partition in which a job is originally executed may be redefined before the job is restarted. The partition used for restart must be at least as large as the original partition. If additional storage is assigned to the low end of the partition, an equal amount of storage must be assigned to the high end of the partition to allow virtual storage supervision routines to build necessary control blocks. After it has initiated a step being restarted at a checkpoint, the system reads the checkpoint entry again. The system uses the contents of the entry to restore virtual storage and to reposition data sets that were being processed at the checkpoint.

After it has initiated a step being restarted at a checkpoint, the system reads the checkpoint entry again. The system uses the contents of the entry to restore virtual storage and to reposition data sets that were being processed at the checkpoint.

How MOD Data Sets Are Handled During Automatic Step Restart

When automatic step restart has been requested for a step, the system saves, for each MOD data set that is on a direct–access volume and used by the step, the TTR (and track balance) of the end of the data set. Saving occurs when each data set is first opened. If restart occurs, the saved TTRs are used to indicate the ends of the data sets when the data sets are first opened again. Thus, if the step writes data in such a data set during the original execution, the step will write over the data during the restart. The action described here does not take place if restart at a checkpoint occurs.

If a MOD data set on tape is used in the restart step, the data set is not repositioned at the start of the restart execution. Therefore, data written into it during the restart execution follows the data written during the original execution. The programmer may wish to reposition the data set so that the data written during the restart execution overlays the data written during the original execution.

Caution Concerning Automatic Step Restart After Checkpoint/Restart

If a step is executing as the result of an automatic or deferred checkpoint/restart, and if you attempt an automatic step restart of this step, the attempt may be unsuccessful if the JCL of the step refers to any new data sets on direct–access volumes. When the step is initiated during the checkpoint/restart, the failure occurs because all the step's data sets that have a NEW disposition are changed to a disposition of OLD by the system. Therefore, when the special disposition processing that prepares for a step restart occurs, all data sets used by the step appear to be OLD and are kept. When the step restart occurs, the scheduler tries to obtain space for data sets specified as NEW in the JCL for the step. If the attempt for data set space is made on the volume that already contains the data set, the failure occurs because of the apparent presence of a "duplicate DSCB on direct–access volume."

Deferred Step Restart

How to Request Deferred Step Restart

The programmer causes a deferred step restart of a job by coding the RESTART parameter on the JOB statement and then by resubmitting the job. The parameter specifies a job step, or a step of a cataloged procedure. The effect of the parameter is simply to restart the job at the beginning of the specified step. Steps preceding the restart step are interpreted, but not initiated.

Original Deck

```
//MYJOB    JOB    MSGLEVEL=1[1]                  No automatic restart requested
//STEP1    EXEC
           .
           .
           .
//STEP2    EXEC   PGM=MYPROG
           .
           .
           .
//STEP3    EXEC
```

Resubmitted Deck

```
//MYJOB    JOB    MSGLEVEL=1, RESTART=STEP2   Causes restart of job at STEP2
//STEP1    EXEC
           .
           .
           .
//STEP2    EXEC   PGM=MYPROG
           .
           .
           .
//STEP3    EXEC
```

1. MSGLEVEL = 1 need not be coded for VS1.

Figure 10. Requesting a Deferred Step Restart

The CHKPT macro instruction may or may not be coded in the user's program. Figure 10 illustrates a job as it is originally submitted and the same job as it is resubmitted for step restart. Assume that the results of STEP2 were unsatisfactory due to abnormal termination or incorrect data when the job was executed originally.

JCL Requirements and Restrictions

To perform a deferred step restart, the user must provide the data set environment required by the restart job. This may be accomplished by using the conditional disposition subparameter in the appropriate DD statements during the original execution of the job. Conditional dispositions in the original deck should be used to:

- Delete all NEW data sets used by the step to be restarted.
- Catalog all data sets that are passed from steps preceding the restart step to the restart step or to steps following the restart step. Abnormal termination of the restart step, when it is originally run, will then cause the passed data sets to be cataloged. Thus, the information will be available to the following steps when the deck is resubmitted.
- Keep all OLD data sets used by the restart step, other than those passed to the step.

If a MOD data set on tape is used in the restart step, the data set is not repositioned at the start of the restart execution and thus data written into it during the restart execution follows the data written during the original execution. The programmer may wish to reposition the data set so that the data written during the restart execution overlays the data written during the original execution.

The following rules apply to the restart deck:

- The RESTART parameter must be coded on the JOB statement.
- If data sets are passed from steps preceding the restart step to the restart step or to steps following the restart step, the DD statements used to receive the data sets must entirely define the data sets. They must explicitly specify volume serial numbers, device type, data set sequence number and label type, unless this information can be retrieved from the catalog. This is why it is recommended that passed data sets be conditionally cataloged during abnormal termination of the original execution. Note that label type cannot be retrieved from the catalog.
- Generation data sets created and cataloged in steps preceding the restart step must not be referred to in the restart step or in steps following the restart step by the relative generation numbers used to create them. They must be referred to by their actual relative generation numbers. For example, a data set created as the +1 data set must be referred to as the 0 data set (assuming that the +2 data set was not also created).
- The EXEC statement PGM and COND parameters and the DD statement SUBALLOC and VOLUME=REF parameters must not be used in the restart step or in steps following the restart step if they contain values of the form stepname or stepname.procstepname, referring to a step preceding the restart step.

Resource Variations Allowed in Deferred Step Restart

A deferred step restart merely allows the restarted execution of a job to begin at other than the first step of the job. Therefore, job step initiation and allocation of resources are accomplished normally. The following variations are allowed upon restart:

- Variation of device and volume configuration
- Variation in JCL and data in the resubmitted deck

Deferred Checkpoint/Restart

How to Request Deferred Checkpoint/Restart

The programmer causes a deferred checkpoint/restart of a job by the following procedure:

- He has the option of coding a special form of the RD parameter (RD=NR) in the original job deck. This specifies that if the CHKPT macro instruction is executed, a checkpoint entry is to be written, but an automatic checkpoint/restart is not to be requested.
- He causes execution of the CHKPT macro instruction, which writes a checkpoint entry.
- The programmer resubmits the job whether or not it terminated abnormally. For example, he might resubmit it because a volume of one of its input data sets was in error and had caused the corresponding part of an output data set to be in error.
- The programmer codes the RESTART parameter (RESTART=(stepname, checkid)) on the JOB statement of the restart deck. Thus, the parameter specifies

both the step to be restarted and the checkid that identifies the checkpoint entry to be used to perform the restart.

- He places a SYSCHK DD statement immediately before the first EXEC statement in the restart deck. It specifies the checkpoint data set from which the specified checkpoint entry is to be read and is additional to any DD statements in the deck that define data sets into which checkpoint entries are to be written. Figure 11 illustrates a job when it is originally submitted and when it is resubmitted for a deferred checkpoint/restart. Assume in Figure 11 that STEP2, when originally executed, terminates abnormally at some time after CH04 has been written. Note that, in the resubmitted deck, the programmer requests that STEP2 be restarted using the checkpoint entry identified as entry CH04.

Original Deck

```
//MYJOB   JOB   RD=NR                 Requests that automatic
*                                     restart not occur (optional)
//STEP1   EXEC
                .
                .
                .
//STEP2   EXEC  PGM=MYPROG            MYPROG issues CHKPT macro
//NAME1   DD    DSNAME=NAME2          Describes checkpoint data set
                .
                .
                .
//STEP3   EXEC
```

Resubmitted Deck

```
//MYJOB   JOB   RESTART=(STEP2,CH04)  Request restart at CH04 in
*                                     STEP2
//SYSCHK  DD    DSNAME=NAME2          Describes data set which
*                                     contains CH04
//STEP1   EXEC
                .
                .
                .
//STEP2   EXEC  PGM=MYPROG
//NAME1   DD    DSNAME=NAME2          Describes data set in which
*                                     new checkpoint entries will
*                                     be written
                .
                .
                .
//STEP3   EXEC
```

Figure 11. Requesting a Deferred Checkpoint/Restart

JCL Requirements and Restrictions

To perform a deferred checkpoint/restart the programmer must provide the data set environment required by the restart job. He may do this by using conditional dispositions during the original execution.

Conditional dispositions should be used to:

- Keep all data sets used by the restart step.
- Catalog all data sets being passed from steps preceding the restart step to steps following the restart step. Even though the step that terminates abnormally is not using the passed data sets, its termination will cause the cataloging of the data sets if the conditional catalog parameter is used in the preceding steps.

Note that temporary data sets cannot be kept.

The following rules must be adhered to when resubmitting a job for deferred checkpoint/restart:

1. A RESTART parameter with a checkid subparameter must be coded on the JOB statement.
2. A SYSCHK DD statement must be placed in the job deck immediately before the first EXEC statement.
3. The EXEC statements in the job deck must have unique names. (The system searches for the named restart step.)
4. The JCL statements and data in steps preceding or following the restart step can be different from their original forms. However, all backward references must be resolvable.
5. The restart step must have a DD statement corresponding to each DD statement present in the step in the original deck, and the names of the statements must be the same as they were originally. However, the restart step can contain, in any position, more DD statements than it contained originally. However, the total number of volumes specified at restart must equal or exceed the number specified originally at checkpoint.
6. If a DD statement in the restart step in the original deck defined a data set that was open at the checkpoint to be used, the corresponding statement in the restart deck must refer to the same data set, and the data set must be on the same volume and have the same extents recorded in its DSCB as it did originally.[1] If the data set is multivolume and was being processed sequentially, only the part of the data set on the volume in use at the checkpoint need be the same as it was originally.

   When there is no need to read or modify a data set after restart, the data set can be replaced by a dummy data set if the original data set was processed sequentially and the job step is not restarted from a checkpoint within the data set's end–of–volume exit routine. Of course, the data set must not be the checkpoint data set used to restart the job step. Also, in VS1 a SYSIN or SYSOUT data set cannot be replaced by a dummy data set. Allocation will be done for each DD statement in the job step where the checkpoint was taken, even if the data set was closed at the time of the checkpoint.

---

1. The extents can differ as follows: In the DD statement, the user can request that additional space be allocated to the data set when the space currently available is exhausted. If space is allocated after a checkpoint is taken, this space is indicated in the DSCB; on restart from the checkpoint, the space is released and the DSCB contents are changed to what they were at the checkpoint.

   In the DD statement, the user can request that unused space be released at the end of the job step. If the space is released, the DSCB may indicate a reduced extent for the data set when deferred restart at a checkpoint occurs; no space is allocated to replace that which was released. Note that space is not released when step termination is followed by automatic restart.

7. Data in the restart step need not be the same as it was originally. If data following a DD * statement was present originally and is entirely omitted in the restart deck, the delimiter (/*) statement following the data may also be omitted. The delimiter statement following a DD DATA statement may not be omitted.

8. Except for the requirements stated in rules 4 through 7, the JCL statements and data in the restart step can be different from their original forms. In particular, the DUMMY parameter can be used for any data set (except, in VS1, SYSIN and SYSOUT data sets) that was not open at the checkpoint.

9. If data sets are passed from steps preceding the restart step to steps following it, the DD statements receiving the data sets must entirely define them. They must explicitly specify volume serial numbers, device type, data set sequence number, and label type, unless this information can be retrieved from the catalog. This is why it is recommended that passed data sets be conditionally cataloged during abnormal termination of the original execution. Note that label type cannot be retrieved from the catalog.

10. The EXEC statement PGM and COND parameters and the DD statement SUBALLOC and VOLUME=REF parameters must not be used in steps following the restart step if they contain values of the form stepname or stepname.procstepname referring to a step preceding the restart step.

Resource Variations Allowed in Deferred Checkpoint/Restart

The system's device and volume configuration can be different from what it was during the original execution of the job. The allowable differences are those described earlier in this chapter under "Resource Variations Allowed in Automatic Restart."

How the System Works During Deferred Checkpoint/Restart

After the system has read and interpreted the restart deck, it reads the specified checkpoint entry and merges information from it into the input work queue entry for the job. As a result, the work queue entry differs from the entry existing during the original execution, as described earlier in this chapter. (Refer to "How the Job Deck is Reinterpreted and the Input Work Queue Merged" in the section "Automatic Restarts," subsection "How the System Works at Automatic Restart.")

Next, the system initiates the restart step normally. The system reads the specified checkpoint entry again and functions as in the automatic restart case. Restart is delayed until the required virtual–storage area is available.

CHAPTER 5: WHAT THE OPERATOR MUST CONSIDER

This chapter describes the system messages and operator functions during various types of restarts and includes discussions on how the operator's decisions and choice of commands can cause variations in the use of system resources. The chapter is divided into two parts: one on the VS2 environment, the other on the VS1 environment.

VS2 Environment

Automatic Restart Message Sequence

During processing related to automatic checkpoint/restart in VS2, the system issues the following sequence of messages to the operator:

1. A message each time a checkpoint entry is written. Each message contains the checkpoint identification.
2. If the job step terminates because of an ABEND condition, an ABEND message for the job step.
3. If the ABEND code makes the job step eligible for restart, an authorization for restart message that requires a reply.
4. Assuming that restart is authorized and MONITOR JOBNAMES is in effect, an IEFREINT STARTED message, followed by an IEFREINT ENDED message. IEFREINT is the name of a system task called the "restart reader." The restart reader reinterprets internal system records of the job to be restarted.
5. A message indicating the virtual–storage requirements (beginning address and ending address) of the job step to be restarted. This allows the operator to determine that the required virtual storage is not currently in use by a "never ending" task.
6. Normal mount messages.
7. A successful restart message.

During processing related to an automatic step restart after a job step has terminated abnormally in VS2, the sequence is the following:

1. An ABEND message for the job step.
2. If the ABEND code makes the job step eligible for restart, an authorization message that requires a reply.
3. Assuming that restart is authorized and MONITOR JOBNAMES is in effect, an IEFREINT STARTED message followed by an IEFREINT ENDED message.
4. Normal mount messages.

Note that the ABEND message, which is issued as:

```
IEF4501 jobname.stepname.procstepname ABEND code
```

is always displayed if a job step terminates abnormally. In addition, if the job step is being executed and the VS2 system fails, this message will be displayed during the next

IPL if system–supported restart is performed. The "code" part of the message has the form Shhh (S followed by a three character hexadecimal number) if the system executed the ABEND macro instruction, or Udddd (U followed by a four–digit decimal number) if the user's program executed the ABEND. It is S2F3 if VS2 system failure occurred.

Operator Options During Automatic Restart

In VS2, if a step requests automatic restart and is eligible for restart, the system displays the following message to request authorization for the restart:

```
xxIEF225D SHOULD jobname.stepname.procstepname[checkid]RESTART
```

Checkid appears in the message only if restart at a checkpoint is requested. It contains from 1 to 16 characters and identifies the checkpoint entry to be used to perform the restart. The operator must reply to the request for authorization as follows:

```
          {'YES' }
REPLY xx, {'NO' }
          {'HOLD'}
```

YES authorizes the restart, HOLD postpones it, and NO prohibits it. During the time that the VS2 system is waiting for the operator to reply to the authorization request, no other task in the system can be initiated or terminated. Therefore, the operator should reply promptly to this message.

If the advisability of allowing the restart is not readily apparent, the operator should reply HOLD to the authorization message. If he later determines that the restart should occur, he can initiate the restart by using the RELEASE command, thereby achieving the same result as with an initial YES reply. If the decision is to deny the restart authorization, the operator can cancel the job in the HOLD queue. However, he must consider that HOLD, as well as YES, causes special disposition processing to occur during the abnormal termination. This processing keeps all OLD (or MOD) data sets and deletes all NEW data sets if a step restart was requested and keeps all data sets if a checkpoint/restart was requested. If the operator subsequently decides to disallow the restart but wants to allow normal disposition (as requested on the job's DD statements) of data sets that were kept, he may release the job, wait until restart has begun, and then cancel the job.

After the authorization request and before the operator replies YES, he may, in some cases, by using the VARY and UNLOAD commands, cause the system's volume and device configuration during a restart execution of the job to be different from what it was during the original execution of the job. Thus, the operator may eliminate use of defective volumes and devices.

The ability to use a different volume usually exists only in the case of a NEW data set on a nonspecific volume. Furthermore, if a checkpoint/restart is to be performed, the data set must not have been open at the checkpoint. The ability to use a different device does not apply to the device or devices containing the SYSRES volume and the SYSJOBQE and LINKLIB data sets. Also, if a checkpoint/restart is to be performed and a data set was open at the checkpoint, the same type of device must be allocated to the data set during both the original and restart executions.

After a YES reply, the job is reinterpreted by a restart reader, named IEFREINT, that is started automatically by the system. At this time, the IEFREINT STARTED and IEFREINT ENDED messages are issued to the operator if MONITOR JOBNAMES is in effect. Before the restart job is reinterpreted and is ready for reinitiation, one or more initiators may select other jobs from the work queue and initiate them. The other jobs may use the virtual storage and devices needed by the restart job and, if they do, the restart will be delayed until the virtual storage and devices are available. If a delay of the restart is undesirable, the operator can hold the queue prior to the YES reply and release the queue after the IEFREINT ENDED message is displayed. This ensures that jobs with the same priority are executed in the sequence in which they were originally submitted.

If a job is to be restarted at a checkpoint, a message specifying the beginning and ending addresses of the virtual storage required for the job step to be restarted is issued after job reinterpretation and any IEFREINT messages. If the required virtual–storage area is currently unavailable because it is being used by other tasks, the restart is delayed until the area is available. If neither the mount messages nor the successful restart message is issued, it is an indication that the required area is currently unavailable. The operator can determine the status of the required area by using the DISPLAY A (Active) command. If a system task is executing in the required area, the operator can either allow the system task to continue to termination (if a reader), or issue the STOP command for the system task (if a reader or writer). If the area is occupied by another job step task, the operator can permit the job step task to continue to termination or he can cancel it.

Note that when an initiator has selected a job for automatic step restart and the job has been reinterpreted, no message is issued to the operator regarding virtual storage requirements since its execution is not location dependent.

Deferred Restart Message Sequence

To perform a deferred checkpoint/restart in VS2, the job to be restarted is resubmitted in an input job stream. Messages that contain checkpoint entry identifications were displayed on the console during the original execution of the job and may then be used by the programmer preparing the job for resubmission. When the resubmitted job is restarted, messages appear on the console in the following sequence:

1. A message indicating the virtual–storage requirements of the job
2. Normal mount messages
3. A successful restart message

To perform a deferred step restart in VS2, the job to be restarted is resubmitted. Normal mount messages are displayed.

Operator Considerations During Deferred Checkpoint/Restart

When a job is resubmitted to perform a deferred checkpoint/restart (the RESTART parameter is coded on the JOB statement with a checkid operand), the processing is essentially the same as during an automatic checkpoint/restart after the restart reader has reinterpreted the job. A message is issued to the operator indicating the virtual–storage requirements of the job. Other tasks executing in the required virtual–storage area can delay the restart. The operator can use the DISPLAY A command in the same manner as in an automatic checkpoint/restart.

The required virtual–storage area can also be unavailable for the following other reasons:

- The REGION size parameter for the step is larger when the job is resubmitted than in the original execution, and the area used in the original execution was adjacent to (immediately below) the Master Scheduler Region. Because the area used by the step is not allowed to expand upward into the Master Scheduler Region, the request for a larger region for the step cannot be satisfied.
- A new IPL was performed and, because of different IPL options specified by the operator, the nucleus expanded upward, or the Link Pack Area expanded downward into the required area.

If these conditions exist, a message is displayed indicating that virtual storage for the job step to be restarted is unavailable. The restart is terminated.

VS1 Environment

Automatic Restart Message Sequence

During processing related to automatic checkpoint/restart in VS1, the system issues the following sequence of messages to the operator:

1. A message each time a checkpoint entry is written. Each message contains the checkpoint identification.
2. If the job step terminates because of an ABEND condition, an ABEND message for the job step.
3. If the ABEND code makes the job step eligible for restart, an authorization for restart message that requires a reply.
4. Assuming that restart is authorized and MONITOR JOBNAMES is in effect, an IEFREINT STARTED message, followed by an IEFREINT ENDED message. IEFREINT is the name of a system task called the "restart reader." The restart reader reinterprets internal system records of the job to be restarted.
5. A message indicating direct system output (DSO) requirements. (If this message is written, the job is placed on the HOLD queue.)
6. Normal mount messages.
7. A successful restart message.

During processing related to an automatic step restart after a job step has terminated abnormally in VS1, the sequence is the following:

1. An ABEND message for the job step
2. If the ABEND code makes the job step eligible for restart, an authorization message that requires a reply
3. Assuming that restart is authorized and MONITOR JOBNAMES is in effect, an IEFREINT STARTED message followed by an IEFREINT ENDED message
4. Normal mount messages

Note that the ABEND message, which is issued as:

```
IEF450I jobname.stepname.procstepname ABEND code
```

is always displayed if a job step terminates abnormally. In addition, if the job step is being executed and the VS1 system fails, this message will be displayed during the next IPL if system–supported restart is performed. The code part of the message has the form Shhh (S followed by a three character hexadecimal number) if the system executed the ABEND macro instruction, or Udddd (U followed by a four–digit decimal number) if the user's program executed the ABEND. It is S2F3 if VS1 system failure occurred.

Operator Options During Automatic Restart

In VS1, if a step requests automatic restart and is eligible for restart, the system displays the following message to request authorization for the restart:

```
xxIEF225D SHOULD jobname.stepname.procstepname[checkid]RESTART
```

Checkid appears in the message only if restart at a checkpoint is requested. It contains from 1 to 16 characters and identifies the checkpoint entry to be used to perform the restart. The operator must reply to the request for authorization as follows:

```
          {'YES'}
REPLY xx, {'NO'}
          {'HOLD'}
```

YES authorizes the restart, HOLD postpones it, and NO prohibits it. During the time that the VS1 system is waiting for the operator to reply to the authorization request, no other task in the system can be initiated or terminated. Therefore, the operator should reply promptly to this message.

If the advisability of allowing the restart is not readily apparent, the operator should reply HOLD to the authorization message. If he later determines that the restart should occur, he can initiate the restart by using the RELEASE command, thereby achieving the same result as with an initial YES reply. If the decision is to deny the restart authorization, the operator can cancel the job in the HOLD queue. However, he must consider that HOLD, as well as YES, causes special disposition processing to occur during the abnormal termination. This processing keeps all OLD (or MOD) data sets and deletes all NEW data sets if a step restart was requested and keeps all data sets if a checkpoint/restart was requested. If the operator subsequently decides to disallow the restart but wants to allow normal disposition (as requested on the job's DD statements) of data sets that were kept, he may release the job, wait until restart has begun, and then cancel the job.

After the authorization request and before the operator replies YES, he may, in some cases, by using the VARY and UNLOAD commands, cause the system's volume and device configuration during a restart execution of the job to be different from what it was during the original execution of the job. Thus, the operator may eliminate use of defective volumes and devices.

The ability to use a different volume usually exists only in the case of a NEW data set on a nonspecific volume. Furthermore, if a checkpoint/restart is to be performed, the data set must not have been open at the checkpoint. The ability to use a different device does not apply to the device or devices containing the SYSRES volume and the SYSJOBQE and LINKLIB data sets. Also, if a checkpoint/restart is to be performed

and a data set was open at the checkpoint, the same type of device must be allocated to the data set during both the original and restart executions.

After a YES reply, the job is reinterpreted by a restart reader, named IEFREINT, that is started automatically by the system. At this time, the IEFREINT STARTED and IEFREINT ENDED messages are issued to the operator if MONITOR JOBNAMES is in effect. Before the restart job is reinterpreted and is ready for reinitiation, one or more initiators may select other jobs from the work queue and initiate them. The other jobs may use the virtual storage and devices needed by the restart job and, if they do, the restart will be delayed until the virtual storage and devices are available. If a delay of the restart is undesirable, the operator can hold the queue prior to the YES reply and release the queue after the IEFREINT ENDED message is displayed. This ensures that jobs with the same priority are executed in the sequence in which they were originally submitted.

In some cases, the partition in which a job is originally executed may be redefined before the job is restarted. For example, the operator may redefine the partition after replying HOLD to the restart authorization message, or redefinition may already be pending when the message is issued. The redefined partition may be unsuitable for use in restarting the job at a checkpoint because the required virtual–storage area may be split between two or more partitions or may be allocated to a resident reader or writer. In either case, the system issues a message that indicates the requirements for defining a suitable partition. The operator can either define the required partition or cancel the job.

When a suitable partition has been provided, the following message may be issued if the job is to be restarted at a checkpoint:

```
IEF390E  DSO(outputclass,jobclass,devicetype)
             NEEDED TO RESTART jobname Pn
```

The message indicates that a DSO (direct system output) data set was open at the checkpoint. The data set was part of the specified system output class, and was assigned to a printer, card punch, or magnetic tape unit, as indicated by the message. The device originally used by the data set is no longer available because:

- The operator issued a STOP command to stop DSO processing on the device.
- The operator issued a MODIFY command to assign the device to a different system output class.
- The operator issued a DEFINE command to redefine partitions, and the job step is not being restarted in the original partition.

The operator can assign a device to the required system output class by issuing a START or MODIFY command. The START command starts DSO processing on a new device for the restart partition. The MODIFY command changes the system output class for a device that has already been started for the partition. When necessary, a STOP command can be issued for a DSO device started for another partition, and a START command issued for the same device in the restart partition. If a STOP command is issued for a DSO device being used by another job, the command will take effect when that job terminates.

Message IEF390E is issued once for each system output class that requires DSO device assignment. The job is then placed on the HOLD queue. The operator must release the job for execution after assigning the required devices. If the required devices cannot be assigned, the operator should cancel the job.

Deferred Restart Message Sequence

To perform a deferred checkpoint/restart in VS1, the job to be restarted is resubmitted in an input job stream. Messages that contain checkpoint entry identifications were displayed on the console during the original execution of the job and may then be used by the programmer preparing the job for resubmission. When the resubmitted job is restarted, messages appear on the console in the following sequence:

1. When required virtual storage is not immediately available, a message indicating the virtual–storage requirements of the job
2. When a direct system output (DSO) device must be started, a message indicating DSO requirements
3. Normal mount messages
4. A successful restart message

To perform a deferred step restart in VS1, the job to be restarted is resubmitted. Normal mount messages are displayed.

Operator Considerations During Deferred Checkpoint/Restart

When a job is resubmitted to perform a deferred checkpoint/restart (the RESTART parameter is coded on the JOB statement with a checkid operand), the processing is essentially the same as during an automatic checkpoint/restart after the restart reader has reinterpreted the job.

If partitions have been redefined since the job was originally executed, there may be no partition suitable for restarting the job because the required virtual–storage area may be split between two or more partitions, or may be included in the partition for a resident reader or writer. In either case, the system issues a message indicating the requirements for defining a suitable partition. The operator can either define the required partition or cancel the job.

The required virtual–storage area may also be unavailable because a new IPL was performed and, because of different IPL options specified by the operator, the nucleus expanded into the required area. If this condition exists, a message is displayed indicating that virtual storage for the job step to be restarted is unavailable. The restart is terminated.

When virtual–storage requirements can be satisfied, message IEF390E may be issued to define direct system output (DSO) requirements. Operator response is the same as in the case of automatic checkpoint/restart.

CHAPTER 6: STORAGE ESTIMATES

Checkpoint/Restart Work Area

In VS1, the user must ensure that enough virtual storage for a special work area is available prior to execution of the CHKPT macro instruction. The algorithm for computing the size of the work area is as follows:

Virtual Storage specified: work area = 1180 + T + 48(N-2) bytes

Real storage specified: work area = 1180 + T + 48(N-2) + 16C bytes

where: T = the size of the TIOT at checkpoint time

N = the number of OPEN data sets at checkpoint time

C = the number of channel programs specified in the checkpoint DCB (if not specified, default is 1)

In VS2, the user need not provide space in his region for the checkpoint/restart work area. The system obtains the necessary space from subpool 253.

Notes:

- For reference purposes, the size of a TIOT in bytes is: 28+20A+4B, where A is the total number of data sets of the job step (including the JOBLIB if present) and B is the sum of any devices exceeding one allocated to each data set.
- N must: (1) have a value of at least 2 and (2) include the checkpoint data set whether it is open or not.
- If checkpoint/restart opens the checkpoint data set, the user must provide space for the IOB. With VS1, the user must also provide space for the DEB and ECB.
- For VS1, the user must provide 344 additional bytes.
- If a user's direct-access output data set requires a new extent after a checkpoint has been taken, and then an automatic restart is attempted, the restart function requires an additional 384 bytes of virtual storage. This virtual storage is acquired from the problem program partition in VS1 and from subpool 253 in VS2.
- For VS1, the checkpoint/restart work area must begin and end on a 2K-byte boundary.

Checkpoint Data Set Storage Requirements

The checkpoint data set may be on any direct–access device or any magnetic tape device that is supported by BSAM or BPAM. The records forming a checkpoint entry are written in undefined format (format U) in the physical order shown in the table that follows. As the table shows, each checkpoint entry contains four types of records. Some types occur more than once in a single checkpoint entry. For each type of record, the table lists the size of one record and the number of records of that type that will appear in a checkpoint entry. From this table, the space required for each

checkpoint entry can be determined, and when multiplied by the expected number of entries, the space required for the checkpoint data set can be determined.

Record Type	Record Size	Number of Records
CHR (Checkpoint Header Record)	400 bytes	1
DSDR (Date Set Descriptor Record)	400 bytes	(A/2)[1]
PCR (Page Control Record)[2]	256 bytes	B/2
ASR (Allocated Storage Record)[3]	2048 bytes	C/8
CIR (Core Image Record)	2048 bytes	D/2048
SUR (Supervisor Record)	200 bytes	1 in VS1 E/70 in VS2

1. Add one record for the first generation data group (GDG) data set and a second record for each additional 4 GDG data sets. Add one record for each data set requiring 6 to 20 volumes and a record for each additional 15 volumes.

2. PCRs are built in VS1 only.

3. ASRs are built in VS2 only.

where:

A = total number of data sets in the job step

B = number of pages in the partition with a storage key of 0

C = number of continuous blocks of storage within the user's region

D = problem program virtual-storage size defined as:

(1) The size of the partition in VS1

(2) The size of the region in VS2

E = number of bytes of system queue space (SQS) used for the problem program

Resident Access Methods (VS1)

The checkpoint/restart facility processes the checkpoint data set using The access method modules required to process the checkpoint data set must be resident in virtual storage.

At system generation, access method modules are made resident by the SUPRVSOR macro instruction. This macro instruction must specify RESIDNT=ACSMETH. As a result, certain access method modules are loaded automatically at IPL.

The modules loaded automatically are those listed in SYS1.PARMLIB These modules are selected by the installation, although a standard list is suggested by IBM. The standard list includes the modules required to process a checkpoint data set, except those required for track overflow or page fixing.

In defining the list IEAIGG00, the installation can include the modules required for track overflow and page fixing. However, it can omit other modules that are required to process the checkpoint data set. Any module that is omitted is not loaded automatically at IPL.

When processing a checkpoint data set requires modules not listed in IEAIGG00, the installation must define an alternate list that includes the required modules. This list must be a member of SYS1.PARMLIB, and must be named IEAIGGxx, where xx represents any two letters or digits.

When an alternate list is defined, OPTIONS=COMM must be specified in the SUPRVSOR macro instruction at system generation. This operand causes the following message to be printed during IPL:

```
IEA101A SPECIFY SYSTEM PARAMETERS
```

The operator must be instructed to reply RAM=xx, where xx represents the last two characters in the name of the alternate list. If the operator replies correctly, the modules listed in IEAIGGxx are loaded and remain resident until the next IPL. If the operator does not reply RAM=xx, the modules listed in IEAIGG00 are loaded. Note that only one of the lists (IEAIGG00 or IEAIGGxx) is used during each IPL.

Caution: When real storage is specified for a task, the modules listed below with a † beside them must be resident in the nucleus. Refer to the *OS/VS1 Planning and Use Guide* for informaiton on how to make these modules resident.

Modules Required for Checkpoint Restart: The following modules must be resident:

Module	Approximate Size	Required by
IGG019BA	400	All checkpoint data sets
IGG019BB	300	
IGG019CC	480	
IGG019BC	240	Checkpoint data sets on direct–access devices
IGG019CD	630	
IGG019CH†	130	
IGG019C1*†	350	Track overflow
IGG019C2*	1050	
IGG019C3*†	350	
IGG019HT*†	200	Tasks for which virtual storage is specified. IGG019HT is a page-fix appendage.

If a checkpoint data set is to reside on an RPS (2305 or 3330) device, the following additional modules must be resident:

IGG019C0†	250	Channel end (Format U)
IGG019FN*†	120	Start I/O appendage—RPS
IGG019C4*†	300	End–of–extent appendage for search direct
IGG019FP*†	490	Channel end for search direct

*These modules are not included in the IBM standard list.

Defining a Resident Module List: A list of resident modules can be created or modified by means of the IEBUPDTE utility program. The procedure is described in the *OS/VS1 Planning and Use Guide.*

Resident Checkpoint/Restart Module for VS1

To improve system performance during tape data set repositioning at restart, the following module should be resident:

Module	Approximate Size	Description
IGC0S05B	940	Repositions tape data sets at restart

CHAPTER 7: MISCELLANEOUS INFORMATION

VS2 Track Stacking

The checkpoint/restart facility can be used with the VS2 track stacking facility.

Job and Job Step Accounting and Checkpoint/Restart

In VS2, the system accumulates CPU time used for each job step and job. An installation can provide an accounting routine that will be given control at step initiation, step termination, and job termination for the purpose of accessing these time values. Accounting routines are discussed in detail in the *OS/VS1 Planning and Use Guide* and *OS/VS2 Planning and Use Guide.* The relationship between the checkpoint/restart facility and the step time and job time values available to the accounting routine are as follows:

- At termination (either normal or abnormal) of an original execution, the step time and job time accumulated are available to the accounting routine.
- If a job is to be restarted at a checkpoint, the system executes a special step, named IEFDSDRP, before the restart step. The accounting routine is not given control during initiation or termination of this step.
- At initiation of the restart step during an automatic restart, step time and job time accumulated for the original execution are again available to the accounting routine.
- At initiation of the restart step during a deferred restart, step time and job time are zero.
- At termination of a restart step and at all subsequent times when the accounting routine is given control during the restart execution, the step and job times reflect only the time used during the restart execution. The time used by the IEFDSDRP step is not reflected.

To illustrate these points, assume that, in an original execution, Step A uses 2 minutes of CPU time and Step B uses 3 minutes of CPU time and abnormally terminates. At step termination the step time is 3 and the job time is 5. If automatic restart is performed for Step B, a step time of 3 and a job time of 5 are again available to the accounting routine at the reinitiation of Step B. If Step B then uses 4 minutes of CPU time and terminates, a step time of 4 and a job time of 4 are available to the accounting routine at step termination.

Note that the two values available at the time the restart step is initiated are provided for information purposes only. They are not reflected in the step and job running times presented at termination time of the restarted job. Thus the user need not be charged twice for the time accumulated up to the ABEND.

Another point to be considered in a user's accounting routine is the effect of a restart on the step sequence number available to the accounting routine. The following list

indicates the sequence number presented to the accounting routine under the various restart conditions:

Condition	Step Sequence Value for Step n
Original Execution	n
Automatic Step Restart	n
Automatic Checkpoint/Restart	n+1
Deferred Step Restart	1
Deferred Checkpoint/Restart	2

Whenever an automatic restart is performed the step sequence value accurately reflects the position of the step in the job. In the case of an automatic checkpoint/restart, the IEFDSDRP step has been executed before the restarting step. This accounts for the n+1 value.

In the case of a deferred restart, the restarting step is either the first step of the restart job or, in the case of a deferred restart from a checkpoint, it is the second (having been preceded by IEFDSDRP).

VS2 Job Step Time Limit

If VS2 is used, the EXEC statement TIME parameter can be used to specify a limit on the CPU time to be used by the related step. With any kind of restart, the entire value of the limit specified for the job step applies to the restart step. In the case of a deferred restart, the programmer may specify a limit different from the limit he specified originally.

If the CPU time used by a step exceeds the specified limit while a checkpoint entry is being written, the entry is invalid and abnormal termination occurs. A preceding checkpoint entry can be used to perform a deferred restart. (If it is, and if sufficient checkpoints are taken during the restart execution, the invalid checkpoint entry will be overwritten by a valid entry.)

Completion of Step or Job Termination at System Restart

If a step or a job is terminating when system failure occurs, the termination will be completed during the system restart that the operator may perform after the failure. This will occur whether or not the step or job uses the checkpoint/restart facility. If other than the last step of a job is terminating when the failure occurs, the termination will be completed during the system restart and the next step of the job will subsequently be initiated. If the last step of a job is terminating, or if the job is terminating, all necessary terminations will be completed. If a job requests an automatic restart and then abnormally terminates, and if system failure occurs before the restart processing is complete, the processing will be completed during the system restart.

COBOL RERUN Clause

The COBOL RERUN clause may be used to provide the COBOL user with linkage to the checkpoint/restart facility. Cautions and restrictions on the use of the checkpoint/restart facility also apply to the use of the RERUN clause.

Checkpoint/Restart and the Sort/Merge Program

When performing a sort with the sort/merge program, the user can, by including the CKPT parameter in his sort control statements, cause checkpoint entries to be written and an automatic checkpoint/restart to be requested. The job control language can be used to request automatic or deferred step restarts or a deferred restart at a checkpoint.

PL/I Checkpoint/Restart Capability

The PL/I user can invoke automatic and deferred step restart and can also take checkpoints and invoke automatic and deferred checkpoint/restarts. To cause a checkpoint entry to be written and request an automatic checkpoint/restart, the user codes in his program:

```
CALL IHECKPT
```

Each checkpoint entry in the checkpoint data set is identified by a system–generated checkid. A system message on the console, which includes the checkid, notifies the operator that a checkpoint entry has been written.

The organization of the checkpoint data set is always physical sequential, and the data set may be written on magnetic tape or a direct–access volume. Partitioned organization cannot be used.

A DD statement must be present in the job stream to define the checkpoint data set. The DISP parameter in this DD statement is used to specify whether single or multiple checkpoint entries are to be written. DISP=(NEW,KEEP) specifies a single checkpoint entry, while DISP=(MOD,KEEP) specifies multiple checkpoint entries.

TCAM Data Set Considerations

A successful restart of a telecommunications access method (TCAM) data set depends on the following conditions:

- The message control program (MCP) region must be active and have enough virtual storage to build the required control blocks.
- The QNAME= parameter in the DD statement of the checkpoint job must be available in the Terminal Table of the MCP region.

APPENDIX A: COMPLETION CODES

Return Codes Associated with the CHKPT Macro Instruction

(Hexadecimal) Code	Meaning
00	*Successful completion.* Code 00 is also returned if the RD parameter was coded as RD=NC or RD=RNC to totally suppress the function of CHKPT.
04	*Restart has occurred* at the checkpoint taken by the CHKPT macro instruction during the original execution of the job. A request for another restart of the same checkpoint is normally in effect. If a deferred restart was performed and RD=NC, NR, or RNC was specified in the resubmitted deck, a request for another restart is not in effect.
08	*Unsuccessful completion.* A checkpoint entry was not written, and a restart from this checkpoint was not requested. A request for an automatic restart from a previous checkpoint remains in effect. One of the following conditions exists: • The parameters passed by the CHKPT macro instruction are invalid. • The CHKPT macro instruction was executed in an exit routine other than the end-of-volume exit routine. • A STIMER macro instruction has been issued, and the time interval has not been completed. • A WTOR macro instruction has been issued, and the reply has not been received. • The checkpoint data set is on a direct-access volume and is full. Secondary space allocation was requested and performed. (Secondary space allocation is performed for a checkpoint data set, but the allocated space is not used. However, had secondary allocation not been requested, the job step would have been abnormally terminated.) • In a system with VS2, the job step comprises more than one task. • The CHKPT macro instruction was issued for a data set on a graphics device.
0C	*Unsuccessful completion.* An uncorrectable error occurred in writing the checkpoint entry or in completing queued access method input/output operations that were begun before the CHKPT macro instruction was issued. A partial, invalid checkpoint entry may have been written. If the entry has a programmer-specified checkid, and the checkpoint data set is sequential, a different checkid should be

specified the next time CHKPT is executed. If the data set is partitioned, a different checkid need not be specified. This code is also returned if the checkpoint routine tries to open the checkpoint data set and the DD statement for the data set is missing.

10 *Successful completion with possible error condition.* The task has control, by means of an explicit or implied use of the ENQ macro instruction, of a serially reusable resource; if the task terminates abnormally, it will not have control of the resource when the job step is restarted. The user's program must, therefore, restore the enqueues.

Additional information regarding explicit and implicit use of the ENQ macro instruction may be found in the section "Cautions in Taking a Checkpoint."

14 *Unsuccessful completion.* End of volume occurred while writing the checkpoint entry on a tape data set. The checkpoint is terminated and processing resumes.

When one of the errors indicated by code 08, 0C, 10, or 14 occurs, the system prints an error message on the operator's console. The message indicating code 08 or 0C contains a code that further identifies the error. The operator should report the message contents to the programmer.

Completion Codes Issued by Checkpoint/Restart

The code 13F indicates that an error occurred during performance of a checkpoint/restart. If a SYSABEND card is included in the job, a dump is produced, and the contents of the system control blocks, as shown in the dump, are unpredictable.

The code 2F3 indicates that a job was executing normally when system failure occurred.

APPENDIX B: END–OF–VOLUME EXIT ROUTINE (TAKING A CHECKPOINT AT END–OF–VOLUME)

The user can specify, in the related data control block exit list, the address of a routine that is to be given control when end–of–volume is reached in processing a physical sequential data set (BSAM or QSAM). (See *OS/VS Data Management Services Guide* for information about forming an exit list.) The routine is entered after a new volume has been mounted and all necessary label processing has been completed. If the volume is a reel of magnetic tape, the tape is positioned after the tape mark that precedes the beginning of the data. The end–of–volume exit routine may take a checkpoint by issuing the CHKPT macro instruction. If the job step terminates abnormally, it can be restarted from this checkpoint. When the job step is restarted, the volume is mounted and positioned as upon entry to the routine.

The end–of–volume exit routine returns control in the same manner as any other data control block exit routine. Note that restart becomes impossible if changes are made to SYS1.SVCLIB (in VS1) or the link pack area (in VS2) after the checkpoint is taken. the step is restarted, the TTRs of end-of-volume modules must be the same as when the checkpoint was taken.)

On entry to the user's end–of–volume exit routine, the contents of the registers are:

Registers	Contents
0	Zero
1	Address of data control block
2–13	Contents before execution of the input/output macro instruction
14	Return address (must be preserved by the exit routine)
15	Address of the end–of–volume exit routine

Notes:

1. The contents of registers 0 through 13 and 15 need not be preserved by the exit routine.
2. The exit routine must not use the save area pointed to by register 13 upon entry. If the exit routine calls another routine or executes system macro instructions, it must provide its own save area.
3. The exit routine is not provided for EXCP users, since they must explicitly execute the EOV macro instruction.

INDEX

Indexes for reference manuals are consolidated in *OS/VS Master Index,* GC28-0602. For additional information about any subject listed below, refer to other publications listed for the same subject in the *Master Index.*

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

READER'S COMMENT FORM

OS/VS Checkpoint/Restart

Order Number GC26-3784-1

Your comments about this publication will help us to produce better publications for your use. If you wish to comment, please use the space provided below, giving specific page and paragraph references.

Please do not use this form to ask technical questions about the system or equipment or to make requests for copies of publications. Instead, make such inquiries or requests to your IBM representative or to the IBM Branch Office serving your locality.

Reply requested

Yes ☐

No ☐

Name _____

Job Title _____

Address _____

_____ Zip _____

No postage necessary if mailed in the U S A

Order Number GC26-3784-1

YOUR COMMENTS, PLEASE . . .

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

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 2078
SAN JOSE, CALIF.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Monterey & Cottle Rds.
San Jose, California
95114

Attention: Programming Publications, Dept. D78

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)

Order Number GC26-3784-1

GC26-3784-1 Printed in U.S.A.

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)

IBM Technical Newsletter

Base Publication Number	GC26-3784-1
This Newsletter Number	GN26-0754
Date	December 15, 1972
Previous Newsletter Number (s)	None

OS/VS CHECKPOINT/RESTART

This technical newsletter, a part of Release 2 of OS/VS1, provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent releases unless specifically altered. Pages to be inserted and removed are:

cover-xi (remove xiii)
11,12
47-48.1 (48.1 added)
55,56

Each technical change is marked by a vertical line to the left of the change.

Summary of Changes

Changes to the system are summarized under "OS/VS1 Summary of Changes" following "About This Book".

A new return code has been added to indicate EOV on a tape checkpoint data set, and the recovery procedure for this condition has been changed. The checkpoint/restart work area has been enlarged.

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

IBM Corporation, Programming Center–Publishing, Monterey and Cottle Roads, San Jose, California 95114

PRINTED IN U S A