

UNCLASSIFIED

AR-002-005

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

ELECTRONICS RESEARCH LABORATORY

TECHNICAL REPORT

ERL-0143-TR

AUTOMATIC ERASURE OF RELEASED DISK SPACE ON AN
IBM 370 COMPUTER USING THE MVS OPERATING SYSTEM

J.C. Gwatking

S U M M A R Y

The standard MVS operating system supplied for use on IBM 370 computers does not erase or protect data residing on areas of disk that have been released for reuse. This report discusses the problem and describes efficient techniques that have been developed and installed at the DRCS which automatically erase all data when disk space is released by its owner. This has involved modifying the disk space management software and included some restructuring to eliminate contention for system resources while data is erased.



POSTAL ADDRESS: Chief Superintendent, Electronics Research Laboratory,
Box 2151, GPO, Adelaide, South Australia, 5001.

UNCLASSIFIED

TABLE OF CONTENTS

	Page No.
1. INTRODUCTION - THE RESIDUAL DATA PROBLEM	1
2. HARDWARE SOLUTIONS	1
3. SOFTWARE ALTERNATIVES	2
4. MODIFICATIONS TO THE OPERATING SYSTEM	3
4.1 The partial release function	3
4.2 The scratch function	5
5. THE SCRIBBLE MODULE	7
5.1 Error detection	9
6. OTHER CAUSES OF RESIDUAL DATA	9
7. PERFORMANCE	10
7.1 Channel program performance	10
7.2 CPU utilization	11
7.3 Operating statistics	11
8. CONCLUSION	14
GLOSSARY	15
REFERENCES	17

LIST OF APPENDICES

I MODIFICATIONS TO PARTIAL RELEASE	18
II MODIFICATIONS TO SCRATCH	21
III SCRIBBLE PROGRAM LISTING	24
IV GTF RECORD FORMAT	35

LIST OF TABLES

1. COMPARISON OF CHANNEL PROGRAM PERFORMANCE	11
2. AVERAGE SCRIBBLE ACTIVITY PER HOUR	12
3. AVERAGE SCRIBBLE ACTIVITY PER HOUR FOR 90% OF REQUESTS	14

1. INTRODUCTION - THE RESIDUAL DATA PROBLEM

The Laboratories of the Defence Research Centre, Salisbury (DRCS) are engaged on a wide range of scientific and engineering research and development for the defence forces of Australia. The central computer at the establishment provides a general computing service to the hundreds of scientists and engineers involved in the various projects. These users store large amounts of data on disk and on magnetic tape, and much of it is of a sensitive nature. Individual users are allowed access to this data on a "need-to-know" basis.

The operative computer is an IBM 3033 processor with 8 megabytes of real storage and IBM 3350 disk drives are used for all on-line user data. The operating system is MVS Release 3.8, with the Resource Access Control Facility (RACF)(ref.1,2) providing the installation with the means for controlling access to data and services. RACF allows users to nominate which other users are authorized to access their datasets, and at what level (read only, read and write, and so on). It can then monitor the use of each dataset by permitting access only to those users who are authorized and by reporting attempted violations.

However this protection can be lost when space is released from a dataset on disk. Such space immediately becomes available for reuse and no attempt is normally made by the operating system, RACF or the IBM hardware to erase or conceal the residual data contained in it. Another dataset will subsequently be allocated to all or part of the same area and it is a trivial matter for the owner of the new dataset to inspect the information left there by the previous user. Since he is ostensibly accessing his own dataset his actions would attract no suspicion and the normal deterrent of post-violation detection will not exist.

Although the data obtained from such a procedure may not always be useful to the would-be penetrator, because of the effectively random manner in which space is assigned to a new dataset, a user could never be sure that his data had not been accessed without his knowledge.

The standard MVS operating system places the responsibility on the owner of a dataset to ensure that its contents are erased or made unreadable before it is deleted. This has been made reasonably convenient for VSAM datasets with the ERASE parameter(ref.3), but no such feature exists for other types of datasets, which are much more prevalent in most IBM installations. In addition users are certain to make mistakes and forget to erase some of their classified data, so that a more automated solution to the problem is required.

2. HARDWARE SOLUTIONS

Efficient protection of residual data could be provided by means of additions or modifications to IBM hardware components. For example, the control units for the direct access devices could include an automatic encryption and decryption capability, so that while on disk the data would be in an encrypted form and therefore unreadable. It would be decrypted when transferred to working storage and re-encrypted on return to disk. Deletion of the dataset would therefore not expose its contents. A more effective solution, proposed by J.L. Roughan of the DRCS, is for the actual disk drives to be capable of selectively preventing each track from being read, by the use of control bits. In this solution, when a track is released from a dataset the corresponding bit could be set in the drive or on the pack, which would prevent that track from being read until it has been written on again, at which stage the bit would be reset.

Obviously such hardware solutions are beyond the capability of an individual installation. The only viable method of attacking the problem in a

user installation is by modifying operating system software. However such solutions are not universal, a different version being required for each version of the operating system, and may require frequent modification if applied to a component that undergoes regular or substantial IBM maintenance.

3. SOFTWARE ALTERNATIVES

Although an individual installation is restricted to a software solution to the residual data problem, there are several possible approaches. One is to use automatic software encryption techniques for the data on disk. This could either be applied universally or selectively, by user option. The universal approach would introduce an unacceptable CPU overhead, with encryption or decryption required every time a block of data is read or written. On the other hand the selective approach is prone to user error, either by failure to specify that encryption is required or by not realizing that the data could be useful to other interests. There would also be an associated problem of keeping the encryption keys secret.

A more convenient approach is to automatically erase each disk track after it has been deleted but before it is eligible for reuse. Although this too could be applied either universally or selectively, the latter is unacceptable for the reasons already mentioned. The advantage of this technique is that no processing overhead is introduced for normal read or write requests; overhead is only involved when space is released, which is relatively infrequently. The CPU overhead would be slight, though some increase in channel activity is inevitable. In spite of the existing high level of channel utilization it was decided to develop the solution to the residual data problem in this way, after experiments had indicated that the increase could be tolerated.

A secondary consideration of when to perform the actual erasure also needed resolution. Three possibilities were examined.

- (1) The erasure could be performed as the space is freed, but before it is added to the VTOC's free space chain. The disadvantage of this would be the increase in elapsed time for functions that free space, notably the TSO DELETE command. An advantage would be that the current usage details of the space could be available to the erase process, if required. Since the space would not be made available for reuse until it was erased, there would be no period during which any residual data could be examined.
- (2) The erasure could be performed as the space was allocated to a new or existing dataset. This would entail increased elapsed times in other operating system areas, including several less obvious TSO functions. However the primary disadvantage is that the residual data would still be on disk up to the time the space was reallocated. During this period it could be accessible to a dump utility program.
- (3) The third possibility could be to erase the space by a low priority continuously running background job. The deletion of a dataset would involve reassigning the space to a dummy dataset and posting the background job to indicate that it had work to do. After erasing the space the background job would then make it available for reuse. The advantage of this technique is that there would be no increase in elapsed time for any function. However, the disadvantages include the extra complexity of the implementation and the problem that the disk space would be unavailable for reuse for some time. This could affect some jobs that delete a large dataset and immediately attempt to reallocate it on the same volume, for example. The space might not be erased in time and the job could fail if there was not enough other free space available.

The first technique was preferred to the others because the majority of datasets deleted from TSO are fairly small and the increase in elapsed time for commands that entail deletion should be tolerable. Larger datasets tend to be deleted in batch jobs, which can absorb the increase in elapsed time without obvious effect.

4. MODIFICATIONS TO THE OPERATING SYSTEM

There are two different methods by which disk space may be freed - by deletion of a dataset (via SVC 29) and by the partial release of space from a dataset. Both actions invoke DADSM (Direct Access Device Storage Management) modules, though not the same ones, to remove the space from the dataset and return it to the free space chain, making it available for reuse, and to update VTOC entries to indicate the changes. The modifications are therefore restricted to the DADSM component of the operating system, which is currently relatively stable and free of major IBM maintenance.

The basis of the modifications is to invoke a module in the Link Pack Area named SCRIBBLE to perform the actual erasure, passing to it the information it needs. However other rearrangements to the logic of both functions were necessary to ensure that the VTOC of the disk volume was not enqueued or reserved during the possibly lengthy erasure process, thereby preventing other accesses to the disk volume.

4.1 The partial release function

It is necessary to understand the standard processing performed by DADSM in response to a partial release request before the modifications can be described. The relevant steps are therefore outlined below.

- (1) Enqueue on the VTOC of the disk volume exclusively, using the RESERVE SVC. This prevents any access to the VTOC from now until it is released in step(11).
- (2) Read the format 4 DSCB from the VTOC.
- (3) Set the DIRF bit and rewrite the format 4 DSCB, if it is not already set. The DIRF bit indicates that VTOC extent errors may exist and is set by DADSM prior to updating the VTOC and reset afterwards. If the operating system crashes while the VTOC is being updated the bit will be left on. The next time space is allocated on the volume DADSM will detect this and rebuild the format 5 DSCB free space chain before attempting the allocation. It will then reset the DIRF bit.
- (4) Enqueue exclusively on the dataset and from its format 1 DSCB build a table of extents to be released.
- (5) Read and process the dataset's format 3 DSCB if it has one, adding to the extent table.
- (6) Delete the format 3 DSCB, if necessary, by overwriting it with a format 0 DSCB, or
- (7) Rewrite the format 3 DSCB, if necessary, to reflect the reduced extents.
- (8) Rewrite the format 1 DSCB to reflect the new extent descriptions.

- (9) Sort the table of extents to be freed and use it to update the format 5 DSCB free space chain, unless the DIRF bit was already set prior to step(3).
- (10) Reset the DIRF bit and rewrite the format 4 DSCB if necessary.
- (11) Release the disk VTOC.

Obviously it is preferable to use the extent table already built by DADSM to indicate to SCRIBBLE the areas to be erased, to avoid repeating the rather complex processing required to derive the information. This means that the erasure must be performed somewhere between steps (5) and (9), at which stage the space is added to the free space chain. However the VTOC is reserved for this entire period, which is unacceptable. The reason for the reservation is to ensure that any DSCB read will not be altered by another task by the time the information in it is used, and to protect the VTOC while it is being updated. However the VTOC update proper does not commence until step(6), so that the setting of the DIRF bit could be delayed until this time. In addition the format 1 and format 3 DSCB belonging to the actual dataset are effectively protected by the dataset reservation which is acquired at step(4). The only other DSCB involved, the format 4, needs no protection before step(6) since the only information derived from it is constant.

These observations form the basis of the rearrangement to the DADSM logic. Delaying the VTOC reservation until absolutely necessary, just prior to step(6), creates an opportunity to erase the space in the desired environment. The modified processing follows.

- (1) Read the format 4 DSCB, without any VTOC reservation. The only data required at this stage are several constants that will not be updated by other tasks. These include the VTOC extents, the number of DSCBs per track and the number of tracks per cylinder on the device.
- (2) Enqueue exclusively on the dataset and from its format 1 DSCB build a table of extents to be released.
- (3) Read and process the format 3 DSCB, if necessary, adding to the extent table.
- (4) Invoke the SCRIBBLE module to erase the space. Note that the VTOC is not yet reserved.
- (5) Reserve the VTOC of the disk volume exclusively.
- (6) Reread the format 4 DSCB from the VTOC.
- (7) Set the DIRF bit and rewrite the format 4 DSCB if it is not already set.
- (8) Delete the format 3 DSCB, if necessary, or
- (9) Rewrite the format 3 DSCB, if necessary.
- (10) Rewrite the format 1 DSCB to reflect the new extent descriptions.
- (11) Sort the table of extents to be freed and use it to update the format 5 DSCB free space chain, unless the DIRF bit was already set prior to step(7).
- (12) Reset the DIRF bit and rewrite the format 4 DSCB if necessary.

(13) Release the disk VTOC.

This revised logic still provides full integrity for the VTOC while reserving it for the shortest possible time. The cost is an extra read of the format 4 DSCB. In addition the erasure is performed even if the DIRF bit was originally set, indicating a previous VTOC error. This ensures that all unallocated areas on the disk will be clear when the VTOC is rebuilt. If a system failure occurs while the space is actually being erased it will still be allocated to the original dataset when the system is restarted, so there is no exposure.

4.2 The scratch function

A similar reorganization was made to the DADSM scratch logic, which is slightly more complex than for the partial release case. The following steps outline the standard processing performed by DADSM.

- (1) Enqueue exclusively on the dataset.
- (2) Reserve the disk volume VTOC exclusively, locking out further accesses to it.
- (3) Read the format 4 DSCB and the format 1 DSCB from the VTOC, in the same channel program.
- (4) Set the DIRF bit and rewrite the format 4 DSCB, if it is not already set.
- (5) Process the format 1 DSCB, building a table of extents to be freed.
- (6) Delete the format 1 DSCB by overwriting it with a format 0 DSCB, reread it as a check and then read the next DSCB chained from it, or the first format 5 DSCB if there are no more in the chain. A format 1 DSCB may have a format 2 or a format 3 DSCB chained from it, while a format 2 DSCB may in turn have a format 3 chained. All this is done with a single channel program, of the following form:-
 - (a) Search for the VTOC address of the format 1 DSCB (in the variable named OUTCCHHR)
 - (b) TIC (a)
 - (c) Write a format 0 DSCB at this address
 - (d) Search for the VTOC address in OUTCCHHR
 - (e) TIC (d)
 - (f) Read Key and Data into the area INDSCB
 - (g) Seek the track containing the next DSCB (in the variable named INLINESK)
 - (h) Read Home Address
 - (i) Search for the address of the next DSCB (in the variable named INCCHHR)
 - (j) TIC (i)

(k) Read Key and Data into the area INDSCB

- (7) If the next DSCB read is a format 3 process it, adding to the extent table. If it was a format 2 or a format 3 DSCB repeat step(6), with the address of the current DSCB in OUTCCHHR.
- (8) Sort the table of extents to be freed and use this to update the format 5 DSCB free space chain, unless the DIRF bit was already set prior to step(4).
- (9) Reset the DIRF bit and rewrite the format 4 DSCB if necessary.
- (10) Release the enqueue on the disk volume VTOC.
- (11) Release the enqueue on the dataset.

As for a partial release it is desirable for SCRIBBLE to be able to use the extent table already built by DADSM. This means that the erasure must be performed after step(7), since the table is not complete until this point. However, unlike partial release, the VTOC update is already in progress at this stage, so that it does require exclusive reservation. The solution involves separating the task of building the extent table from that of updating the VTOC, delaying the latter until the former is complete. The revised logic is as follows:-

- (1) Enqueue exclusively on the dataset.
- (2) Read the format 4 DSCB and the format 1 DSCB from the VTOC.
- (3) Process the format 1 DSCB, building a table of extents to be freed.
- (4) If there is a further format 2 or format 3 DSCB in the chain then save the address of the format 1 DSCB and read the next DSCB, using only the last three channel commands of the program outlined above. Otherwise go to step(6).
- (5) If the next DSCB read is a format 3 process it, adding to the extent table. If there is a further DSCB chained from this one, then repeat steps (4) and (5), saving the current DSCB address.
- (6) Invoke the SCRIBBLE module to erase the space.
- (7) Reserve the VTOC of the disk volume exclusively.
- (8) Reread the format 4 DSCB from the VTOC.
- (9) Set the DIRF bit and rewrite the format 4 DSCB, if it is not already set.
- (10) Delete the DSCBs whose addresses were saved, if any, by overwriting with a format 0 DSCB. Channel commands (a) to (f) are used for this purpose.
- (11) Delete the DSCB that was read last and read the first format 5 DSCB, using the entire channel program (commands (a) to (k)).
- (12) Sort the table of extents to be freed and use this to update the format 5 free space chain, unless the DIRF bit was already set prior to step(9).

- (13) Reset the DIRF bit and rewrite the format 4 DSCB if necessary.
- (14) Release the enqueue on the disk volume VTOC.
- (15) Release the enqueue on the dataset.

This technique also provides full VTOC integrity and reduced reservation for the scratch function, at a cost of an extra read of the format 4 DSCB. However in some instances a more efficient chained channel program is split into two, adding some extra overhead. This occurs when a format 2 or format 3 DSCB is chained from the original format 1.

5. THE SCRIBBLE MODULE

The SCRIBBLE program is invoked from both the partial release and scratch functions of DADSM to erase one or more disk extents before they are made available for reuse. The program normally resides in the Link Pack Area and receives control by standard DADSM linkage techniques, using the ICRES macro(ref.4).

The information required by SCRIBBLE to execute the request is passed to it in several registers, as follows:-

- (1) Register 6 has the address of the ICRES save area, which reflects the contents of registers 0 to 14, in that order, of the calling module at the time it relinquished control.
- (2) Register 7 has the address of the DADSM extent table(ref.5), which describes the disk areas being freed. The table has the following format:-

Offset	Length	Field
0	1	Flag indicators
1	1	Number of extents in the table
2	2	Used hole count, which indicates the number of DSCBs freed by the operation
4	2	The relative track address of the first track of the first extent area (RTA1)
6	2	The relative track address of the last track of the area, plus 1 (RTA2)
8	4*n	Up to 15 extra RTA1 and RTA2 combinations, describing other extent areas to be freed

- (3) Register 8 contains the number of tracks per cylinder for the device in byte 0 and its UCB address in bytes 1 to 3.
- (4) Register 10 indicates which function of DADSM invoked SCRIBBLE in byte 0 (by the character 'R' for partial release or 'S' for scratch) and the address of the dataset name from which the space is being removed in bytes 1 to 3.

- (5) Register 11 is relevant only for a scratch request. It contains the first byte of the dataset organization from the format 1 DSCB in byte 0 and the relative address of the last block written (in the form TTR) in bytes 1 to 3.
- (6) Register 14 contains the return address.

On entry SCRIBBLE obtains a work area and uses it to construct a DEB describing the extents being freed, a DCB and an IOB. After chaining the DEB from the Task Control Block (TCB) SCRIBBLE can execute channel programs against the extents described in it, but is prevented from accessing other areas on the disk, thus providing protection for them.

For efficiency SCRIBBLE tries to avoid erasing space that is already clear. It does this by first estimating how many tracks must be erased, erasing them using the ERASE channel command and then reading the next track, if there are any left. If a "no record found" condition is raised by this read the track and all subsequent ones are assumed to already be clear, and the erasure is terminated. However, if data is read then a further 30 tracks are erased and another read performed, and so on, until the end of the dataset.

The number of tracks that are erased initially depends on the type of request, the dataset organization and the last TTR value. For a partial release request the number is always one, since the first track may contain an end of file. The second track then determines whether the remainder is already clear or not.

For a scratch request SCRIBBLE erases each track of an ISAM dataset, which does not maintain the last TTR field of the format 1 DSCB and whose internal organization invalidates the assumption that if one track is clear then all subsequent ones will be. For other dataset types the TTR value is used to determine the number of tracks to be erased initially. If the value is zero, as is always the case for VSAM, then no tracks are erased initially and the first track is read. If the TTR value is invalid (it indicates a size greater than the tracks allocated to the dataset) all tracks are erased. For any other value the number is the number of tracks that contain data (as indicated by the TTR), plus one in case the end of file is on the next track.

The initial read check is necessary to ensure that any dataset whose current TTR value is less than what it was at some stage during its life is completely erased. Performing an additional check every 30 tracks enhances the efficiency of the process by detecting the approximate high-water mark of the data and allowing the erasure to be terminated as early as possible. No experiments have been performed to determine the optimum number of erases between each read check. The figure of 30 was arbitrarily chosen, but despite this it seems to be quite effective, as the results in Section 7 indicate.

Although the VTOC contention problem has been resolved by the logic rearrangements described in the previous section, a further problem of VSAM catalog contention can arise. Deletions occurring as a result of a request to Access Method Services (AMS) enter SCRIBBLE with the catalog allocated exclusively. This includes all deletions initiated from TSO, and could cause prolonged lock-outs. SCRIBBLE detects and avoids such situations by first freeing the catalog if more than 5 tracks are being erased and then reacquiring it prior to returning control to DADSM. It uses the standard catalog management routines IGGPRPLF and IGGPRPLM(ref.4), which must be link-edited as aliases of module IGGCLA1. Both routines expect the address of the Catalog Communications Area (CCA) to be in register 11 and the address of the next available 3-word save area from the CCA in register 13, as do all catalog management modules.

The manner by which SCRIBBLE determines if it was invoked as a result of an AMS request is to scan back through the RB chain to see if SVC 26 (Catalog Management) was issued before SVC 29 (DADSM scratch). If so, AMS is involved and the addresses required in registers 11 and 13 can be extracted from the save area of the appropriate SVRB.

SCRIBBLE avoids channel contention problems by not chaining ERASE commands

together. It erases each track using a separate channel program, ensuring that the channel is held only briefly and then released.

Each time SCRIBBLE is invoked it accumulates statistics on its operation and just before returning control writes this information in a GTF user trace record for subsequent analysis. The format of the GTF record is described in Appendix IV, while Appendices I, II and III contain the DADSM modifications and the source code of the SCRIBBLE module.

5.1 Error detection

Since an error in the functioning of SCRIBBLE could have serious operational effects, the module also checks and reports on a variety of error conditions during processing. If any error is detected a message will be written to the operator console, processing will be terminated and an identifying code returned to the caller in register 15. If a work area has already been obtained the GTF record will still be written and will also include the error code. The meanings of the error codes are explained below. Error types 12 and 13 fail to produce a GTF record since no work area is available in which to construct the record at the time they are detected. However error code 2 will also never appear in a record because it is caused by an operation performed after the record is written.

- 0 - the space was erased successfully.
- 1 - the space was erased successfully but the DEBCHK macro to purge the DEB failed.
- 2 - the space was erased successfully but the FREEMAIN macro to release the work area failed.
- 8 - an error was detected in the channel program used to erase or read check the space. Some of the space may not have been erased.
- 12 - an error occurred attempting to acquire a work area and no space was erased.
- 13 - an error was detected in one of the input parameters and no space was erased.
- 14 - an error was detected in the DADSM extent list describing the areas to be freed and no space was erased.
- 15 - the DEBCHK macro to validate the DEB built by SCRIBBLE failed and no space was erased.

At the time of writing, after 7 months operation, SCRIBBLE had not detected any of these errors, nor had it failed in any other way.

6. OTHER CAUSES OF RESIDUAL DATA

Any operation that involves restoring a backup copy of a disk volume may induce the residual data problem if the receiving volume had not been completely erased first.

One example of this is a disk volume reorganization scheme used at the DRCS(ref.6). This procedure involved copying a disk volume to tape, then restoring the copy to the same disk, optionally reordering the datasets and removing unused space from them. The disk volume was not erased prior to the restore because of the time required and the fact that it would destroy a

valuable source of backup if tape errors prevented the execution of the restore step.

This technique caused two different forms of residual data. The first was the normal problem of data left in free areas of disk. The second occurred when a dataset was not truncated during the restore, which could happen for several reasons, so that the unused but allocated space still contained whatever data was left there.

An obvious solution to both problems would be to erase the entire disk volume prior to the restore step. However the reasons already stated for not doing this are still valid. Instead a solution was developed to identify and erase only the areas of disk that actually contained residual data. The solution has two parts. The first addresses the residual data in the free areas of disk, by means of a new program inserted as the final step in the reorganization procedure, just after the restore step. The program gains exclusive control of the VTOC, reads the format 5 DSCB free space chain, allocates dummy sequential datasets over all free areas and then releases the VTOC. Next it writes a single block into each track of all the dummy datasets and then deletes them, allowing SCRIBBLE to erase the space. The second source of residual data was removed by additional logic in the restore program itself. The program now detects when a dataset being restored has more tracks allocated to it than it uses, and generates a channel program to erase the extra ones.

Another case of induced residual data occurred in the dataset backup scheme used at the DRCS(ref.7,8). When a full disk volume was restored using this scheme the same problems arose as with the reorganization procedure. Fortunately, because a common program performed the restore functions of both procedures the problem solutions were also identical.

The only other situation that might induce residual data at the DRCS would be the use of the IBM utility program IEHDASDR(ref.9) to restore one of the system volumes that had been corrupted. However, since the restore would usually be directed to an empty disk volume reserved for such emergencies or to the original volume there should be little risk of residual data. Certainly no user datasets are allowed on such volumes in this installation. If they were a solution could be to reinitialize the VTOC on the receiving volume, allocate a single ISAM dataset over the remainder of the disk and then delete it, thereby invoking SCRIBBLE, before performing the restore.

7. PERFORMANCE

7.1 Channel program performance

Three different techniques of erasing a track were tried before a suitable one was found. Each was tested in a dedicated machine environment, using RMF(ref.10) to monitor channel, control unit and device utilization as well as elapsed time to erase a fixed area of 300 cylinders on a 3350 disk. The first method was to use a Format Write channel command(ref.11) to transfer a few bytes of data to each track and allow the remainder to be erased by the disk drive. The second technique used an Erase channel command to erase each track while the third chained a NOP from an Erase channel command.

The results of the test are shown in Table 1. The control unit busy percentage is computed only during the periods when the channel is not busy. Similarly the device busy percentage is computed only during the periods when the control unit is not busy.

It is obvious from the figures that method 3 is by far the best. It uses minimal channel resource and the effect of the NOP is to disconnect the device from the control unit during erasure, as can be seen by comparing with the results of method 2. Method 1 does not release the channel during the erase process, a totally unacceptable situation. Similarly method 2 releases the channel but not the control unit.

TABLE 1. COMPARISON OF CHANNEL PROGRAM PERFORMANCE

Method	% Channel Busy	% Control Unit Busy	% Device Busy	Elapsed Time (min)
1	48.26	0	97.66	5.01
2	3.06	99.46	50.61	5.02
3	3.56	0.19	98.27	5.00

All three methods erase about 60 cylinders per minute of elapsed time, or 1 cylinder (30 tracks) per second. This is consistent with the 16.8 ms latency time of the device. Since erasure is performed by a channel command program of a Seek and an Erase or Write command, it takes two revolutions per track, or 33.6 ms.

Further tests performed with method 3 verified that overlapped erasure of data on 2 drives connected to the same control unit is possible. Two jobs were started at the same time to erase 300 cylinders each, on different drives. Both jobs would take 5 minutes each, if there were no contention for any resource. In practice one took 5.1 minutes and the other 5.6, indicating that there was some contention but that it was not sufficiently serious to greatly impede either job.

7.2 CPU utilization

The CPU time used by the erase process is extremely small. Numerous tests have established a time of 1.5 seconds of CPU time on an IBM 3033 computer to erase 100 cylinders or 3000 tracks of 3350 disk space. This is a combination of approximately 1 second TCB time and 0.5 seconds SRB time. However this time may or may not be charged to the job that freed the space. For example, a deletion performed during deallocation processing for a batch job step, by coding a disposition of DELETE in the JCL, will not be reflected in the CPU statistics of any SMF record produced by that job. On the other hand, the user will be charged for CPU time incurred by deletions performed under program control, such as by the utility program IEHPRGM or by the TSO DELETE command.

7.3 Operating statistics

It is possible to derive statistics on the operation of SCRIBBLE using the information contained in the GTF record it generates (see Appendix IV). These records were collected for the 9 hour period of 8.30 am to 5.30 pm on each of 10 days taken at random during October and November 1979. From the data the average load handled by SCRIBBLE during an hour of normal operation can be calculated and the results are shown in Table 2. The following observations can be made from the results:-

- (1) Scratch requests far outnumber partial release requests by a factor of nearly 8 to 1.
- (2) About 25% more requests originate from TSO than from batch jobs. However the total number of tracks freed by batch jobs outnumbers that freed from TSO by a factor of nearly 8 to 1. This is mainly due to the deletion of large temporary datasets in batch jobs.

TABLE 2. AVERAGE SCRIBBLE ACTIVITY PER HOUR

	TSO	Batch	Combined
scratch requests	184	157	341
partial release requests	29	15	44
total requests	213	172	385
tracks scratched	1890	14655	16545
tracks released	365	1962	2327
total tracks freed	2255	16617	18872
tracks actually erased	1339	4383	5722
% tracks freed that were erased	59.4	26.4	30.3
requests where at least one read was performed	60	106	166
total number of extra reads required	1	34	35
erases saved by these extra reads	66	1605	1671
total reads	61	140	201
catalog contentions saved	15	1	16
tracks freed per request	10.6	97.0	49.2
tracks erased per request	6.3	25.6	14.9
average seconds per request	0.43	1.41	0.86
average seconds per track erased	0.068	0.055	0.057

- (3) The technique adopted by SCRIBBLE of using Read channel commands to reduce the amount of erasing performed has been extremely successful. In all, only 30% of tracks freed are actually erased, a saving of about 13000 erases per hour.
- (4) The percentage of tracks freed that are erased is markedly different for TSO and batch. The figure for TSO is 59%, indicating a high level of usage of the space allocated to such datasets. However the batch figure of 26% reflects the allocation of large temporary datasets with far more space than is actually required.
- (5) The total number of Read commands issued by SCRIBBLE per hour is only 201. This is a small price for the saving of 13000 erases per hour.

- (6) Only 166 of the 385 requests per hour, or 43%, required any read checks. All space allocated was erased in the remainder of the requests.
- (7) The technique of performing an extra read after erasing each additional 30 tracks beyond the initial amount accounts for only 35 of the total reads per hour. The number of erases saved by this refinement alone is 1671 per hour.
- (8) An average of 16 AMS requests per hour cause SCRIBBLE to release and reacquire the VSAM catalog, to prevent the possibility of prolonged lock-outs.
- (9) The average elapsed time spent within the SCRIBBLE module is 0.43 seconds for TSO and 1.41 seconds for batch. This means that the average TSO deletion response time is increased by nearly half a second. Such an increase would not be noticed since the TSO DELETE command usually requires several seconds to complete. Only the deletions of large datasets are noticeably affected.
- (10) The average time to erase each track is 57 ms, compared with the best possible time of 33.6 ms on a dedicated machine. The increase is due to a combination of device, channel, control unit and CPU contention, as well as overheads in SCRIBBLE itself. The elapsed time per track is noticeably lower for batch than TSO. This is mainly due to the larger size of the average batch request, so that there are more tracks to absorb the initialization and termination overheads of SCRIBBLE.
- (11) Section 7.1 states that at a rate of 30 erases per second, or 1800 erases per minute, only 3.5% of the total capacity of a channel is consumed. The statistics reveal that less than 6000 erases are actually performed per hour, or 100 per minute, distributed fairly evenly over 4 channels. This means that there are 25 tracks erased per minute on each channel, so that the channel capacity actually consumed is negligible, being approximately 0.05%.
- (12) Section 7.1 also states that at the same rate of 1800 erases per minute a disk drive would be saturated. The actual number of erases, 100 per minute, is distributed fairly evenly between 17 disk drives. The average rate is therefore approximately 6 per minute on each drive, so the actual overhead is only 0.3% of the total capacity.

Although the average increase in the response time for TSO deletions is only 0.43 seconds, this and other figures in Table 2 are inflated by a small proportion of very large requests, so that they do not reflect the actual averages seen by the majority of users. Consequently the largest 10 percent of requests initiated from both batch and TSO were discarded and some of the more relevant figures recalculated. The results, presented in Table 3, should be more indicative of what the majority of users experience. From these figures the following points can be observed.

- (1) The 10% of requests discarded accounted for 34% of the total space freed and an extraordinary 84% of the total tracks actually erased.

TABLE 3. AVERAGE SCRIBBLE ACTIVITY PER HOUR FOR 90% OF REQUESTS

	TSO	Batch	Combined
total tracks freed	890	11618	12508
tracks actually erased	282	641	923
% tracks freed that were erased	31.7	5.5	7.4
tracks freed per request	4.6	75.3	36.2
tracks erased per request	1.5	4.2	2.7
average seconds per request	0.14	0.27	0.20
average seconds per track	0.093	0.064	0.074

- (2) The percentage of tracks freed that were also erased has dropped markedly, down to an overall 7.4%. The TSO figure is still much higher than the batch figure, confirming the greater utilization of space in TSO datasets.
- (3) The number of tracks erased is quite small for the typical request (1.5 for TSO and 4.2 for batch). In addition the elapsed time per request has become negligible for both batch (0.27 seconds) and TSO (0.14 seconds).
- (4) The average time to erase each track has understandably risen, due to the fewer tracks per request to absorb the overheads of SCRIBBLE.

8. CONCLUSION

The scheme described has been operating successfully at the DRCS since October 1979 and complements other integrity measures already in force, notably the RACF access control system(ref.12). Its operating characteristics are much better than at first predicted, causing only a slight impact on system performance. This is especially pleasing for the important areas of channel and device utilization and TSO response time for delete operations, where the increases are negligible in the majority of cases. However it is not certain how difficult it will be to maintain the system modifications that form the basis of the solution in the future, although only one minor change has been required during the last six months.

GLOSSARY

Several terms and abbreviations used in this report are unique to IBM computer systems and are briefly described here for quick reference.

AMS (Access Method Services)

This is a group of utility functions used primarily to process VSAM datasets. However some functions, such as deletion, are also applicable to non-VSAM datasets.

CCA (Catalog Communications Area)

This is an operating system control block that passes information between catalog management procedures during the processing of a VSAM catalog.

DADSM (Direct Access Device Storage Management)

DADSM is a component of the IBM operating system that is responsible for allocating, maintaining and releasing storage on direct access devices (disks).

DCB (Data Control Block)

This control block contains information describing the current use of a dataset.

DEB (Data Extent Block)

The DEB is an extension of the information in a DCB. It contains information about the physical characteristics of a dataset, including the areas it occupies on disk.

DIRF

This is the name given to a bit in a VTOC's Format 4 DSCB that indicates that a system failure occurred while the VTOC was being updated and that errors might therefore exist in it.

DSCB (Dataset Control Block)

This is one of a number of 144-byte records that reside in the VTOC of a direct access volume. There are several different types or formats, each with a different purpose. Type 0 denotes a free record, types 1, 2 and 3 describe datasets and their extents, type 4 describes the VTOC itself and contains volume related information, type 5 describes free areas on the disk and type 6 is used for a special type of space allocation known as shared cylinder.

GTF (Generalized Trace Facility)

This is a component of the operating system that can write records to a special dataset describing events that have occurred or processing that has been performed. The records are useful for the analysis of system performance and in problem solving.

IOB (Input/Output Block)

This control block is used for communication between a routine that requests an I/O operation and the I/O supervisor that will perform it.

JCL (Job Control Language)

This is the command language by which a batch job conveys its requirements to the operating system.

LPA (Link Pack Area)

This is an area of virtual storage in an IBM computer system that contains commonly used routines. These routines may be accessed by any user or system task currently executing.

NOP (No-Operation)

NOP is the mnemonic given to a null I/O operation. It causes no action at the addressed device, but does reset certain control information.

RACF (Resource Access Control Facility)

This is a component of the IBM operating system that controls and records accesses made to various system resources, notably disk datasets and tape volumes. It contains provisions for extensive tailoring to an installation's needs.

RB (Request Block)

This control block contains program dependent information needed by the supervisor to transfer control back to that program.

RMF (Resource Measurement Facility)

RMF is an operating system component that records and reports the performance of most aspects of the IBM computer system.

SMF (System Management Facilities)

This is a component of the operating system that gathers and records details of system events. The data can be used for a variety of purposes, including accounting, performance monitoring and activity reporting.

SVC (Supervisor Call)

This is an IBM System/370 machine instruction that invokes a nominated supervisory function.

SVRB (Supervisor Request Block)

This is a particular form of RB that describes the environment of a supervisor program invoked via an SVC machine instruction.

TCB (Task Control Block)

The TCB contains information and pointers associated with the task in process. Numerous components of the control program place information in the TCB and obtain information from it.

TSO (Time Sharing Option)

This is a component of the operating system that provides a wide range of processing functions to a user at a terminal.

UCB (Unit Control Block)

This control block describes the characteristics of a device to the I/O supervisor.

VSAM (Virtual Storage Access Method)

This is a type of dataset organization that can provide direct as well as sequential access techniques, depending on application program requirements.

VTOC (Volume Table of Contents)

This is a special file on a direct access volume that contains information about the volume and the datasets on it. Its records are called Dataset Control Blocks (DSCB).

Only about 30% of space deleted in this installation is actually erased. The remainder is already clear. (We delete about 19000 tracks per hour, erasing about 5700 of them).

REFERENCES

No.	Author	Title
1	IBM Corporation	"OS/VS2 MVS Resource Access Control Facility (RACF) General Information Reference". GC28-0722-3
2	IBM Corporation	"OS/VS2 MVS Resource Access Control Facility (RACF) Installation Reference Manual". SC28-0734-2
3	IBM Corporation	"OS/VS2 Access Method Services". GC26-3841-2
4	IBM Corporation	"OS/VS2 Open/Close/EOV Logic". SY26-3827-2
5	IBM Corporation	"OS/VS2 DADSM Logic". SY26-3828-2
6	Gwatking, J.C. and Collier, R.W.	"Management Procedures for Controlling Data Storage on the IBM System 370 Computer at DRCS". Technical Report, ERL-0055-TR
7	Gwatking, J.C.	"A Selective Dataset Backup Scheme for the DRCS Central Computer". Technical Report, ERL-0021-TR
8	Gwatking, J.C.	"Enhancements to the DRCS Dataset Backup Scheme". Draft Technical Memorandum
9	IBM Corporation	"OS/VS2 MVS Utilities". GC26-3902-0
10	IBM Corporation	"OS/VS2 MVS Resource Management Facility (RMF) Version 2 Reference and User's Guide". SC28-0922-1
11	IBM Corporation	"Reference Manual for IBM 3830 Storage Control Model 2". GA26-1617-4
12	Roughan, J.L. and Gwatking, J.C.	"The Adaptation and Installation of the Resource Access Control Facility (RACF)". ERL-0136-TR, April 1980
13	IBM Corporation	"OS/VS2 MVS System Programming Library : Service Aids". GC28-0674-3

APPENDIX I

MODIFICATIONS TO PARTIAL RELEASE

The modifications are expressed in SMP4 format. They apply to MVS Release 3.8 at PTF level 8001. PTF UZ23177 has been applied to CSECT IGG020P1. CSECTs IGG020P2 and IGG020P3 are at 3.8A base level.

```

++USERMOD(LOCZ021) .
++VER(Z038) FMID(EDM1102) PRE(UZ23177) .
++ZAP(IGG020P1) .
**** ZAP TO PARTIAL RELEASE TO ERASE FREED SPACE.
**** NOTE THAT CSECT IGG020P2 MUST BE EXPANDED BY 288 BYTES.
NAME IGG020P1 IGG020P1 ***** PARTIAL RELEASE *****
**** DUMMY OUT THE RESERVE ON THE VTOC
VER 01B2 0A38 SVC 56 (RESERVE)
REP 01B2 1BFF SR 15,15
VER 01B4 96C0B255 OI DSMADTB2,VTOCR+SMCE
REP 01B4 18FF18FF LR 15,15 LR 15,15
****
****
****
**** DON'T RESET DIRF BIT OR REWRITE FMT4
VER 01D4 9704B06E XI DS4VTOCI,DIRFBIT
REP 01D4 47F0C1EA B SKIPWRT
****
****
****
++ZAP(IGG020P3) .
NAME IGG020P1 IGG020P3
**** DON'T REWRITE FMT4 IF NOT ENQ'ED ON VTOC
VER 006C 4110D118 LA 1,DXIOB
REP 006C 47F0C282 B PATCH AREA (+284)
VER 0284 00000000,00000000,00000000,00000000 ***** PATCH AREA *****
REP 0284 91C0B255 TM DSMADTB2,VTOCR+SMCE
REP 0288 4780C086 BZ NOWRT
REP 028C 4110D118 LA 1,DXIOB
REP 0290 47F0C06E B +70
****
****
****
**** DON'T DEQ VTOC IF NOT ENQ'ED ON IT
VER 010C 4110D1C0 LA 1,ENQAREA
REP 010C 47F0C292 B PATCH AREA (+294)
VER 0294 00000000,00000000,00000000,00000000 ***** PATCH AREA *****
REP 0294 91C0B255 TM DSMADTB2,VTOCR+SMCE
REP 0298 4780C12C BZ MSGTEST
REP 029C 4110D1C0 LA 1,ENQAREA
REP 02A0 47F0C10E B +110
****
****
****
++ZAP(IGG020P2) .
EXPAND IGG020P2(288)
NAME IGG020P1 IGG020P2
VER 0350 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0360 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0370 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0380 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0390 00000000,00000000,00000000,00000000 ** PATCH AREA **

```

```

VER 03A0 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 03B0 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 03C0 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 03D0 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 03E0 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 03F0 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0400 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0410 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0420 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0430 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0440 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0450 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0460 00000000,00000000,00000000,00000000 ** PATCH AREA **
**** SAVE CURRENT DXCCW4-6 IN UNUSED PART OF FMT4. THESE CCW'S READ THE
**** FMT4
VER 000A 91FFB24E TM OUTCCHHR+K4,F3IND
REP 000A 47F0C34E B PATCH AREA (+350)
REP 0350 D217B078D188 MVC VTODSCB+24(24),DXCCW4
**** SET EXTENT NUMBER IN DADSM EXTENT TABLE FOR PROCESSING BY EXIT
REP 0356 4250B1D9 STC 5,EXTNUM
REP 035A 91FFB24E TM OUTCCHHR+K4,F3IND
REP 035E 47F0C00C B +0E
****
****
****
**** LINK TO SCRIBBLE EXIT BEFORE UPDATING FMT3
VER 01DA 4100D170 LA 0,DXCCW1
REP 01DA 47F0C360 B PATCH AREA (+362)
REP 0362 4250B1D9 STC 5,EXTNUM
REP 0366 4590C37C BAL 9,CALLEXIT
REP 036A 4100D170 LA 0,DXCCW1
REP 036E 47F0C1DC B +1DE
****
****
****
**** LINK TO SCRIBBLE EXIT BEFORE UPDATING FMT1
VER 02A6 4130C301 LA 3,NEXTXCTL
REP 02A6 47F0C370 B PATCH AREA (+372)
REP 0372 4590C37C BAL 9,CALLEXIT
REP 0376 4130C301 LA 3,NEXTXCTL
REP 037A 47F0C2A8 B +2AA
****
****
****
**** THIS EXIT INVOKES SCRIBBLE AND PROCESSES THE VTOC.
**** LEAVE IF VTOC ALREADY RESERVED (IE. IF WE HAVE ALREADY BEEN THROUGH
**** HERE). THIS WILL HAPPEN IF THE DATA SET HAD BOTH A FMT1 AND FMT3,
**** WHEN THE EXIT WILL BE CALLED TWICE
REP 037E 91C0B255 CALLEXIT TM DSMADTB2,VTOCR+SMCE
REP 0382 0779 BNZR 9
**** DON'T INVOKE SCRIBBLE IF NO EXTENTS
REP 0384 9500B1D9 CLI EXTNUM,0
REP 0388 4780C3D2 BE PASSEXIT
**** ESTABLISH RETURN ADDRESS
REP 038C 41E0C3D2 LA 14,PASSEXIT
**** SETUP PARAMETERS FOR SCRIBBLE
REP 0390 4170B1D8 EXTENT TABLE LA 7,DADSMTBL
REP 0394 5880D230 UCB ADDRESS L 8,DXUCBADR
REP 0398 186B SAVE AREA LR 6,11
REP 039A BF88B075 TRKS/CYL ICM 8,8,DS4DEVSZ+3
REP 039E 41A0D064 DSNAME LA 10,DXJBF

```

```

REP 03A2 BFA8C3C8      'R'          ICM 10,8,SCRIBBLE+2
***** SIMULATE ICRES MACRO USED BY DADSM FOR TRANSFERRING CONTROL
REP 03A6 18FB          LR 15,WRKAREA
REP 03A8 900EF000      STM 0,14,0(15)
REP 03AC 41100020      LA 1,X'20'
REP 03B0 1BF1          SR 15,1
REP 03B2 D20BB054C3C6  MVC WTGMODNM(12),SCRIBBLE
REP 03B8 4160B054      LA 6,WTGMODNM
REP 03BC 58500010      L 5,CVTPTR
REP 03C0 58505110      L 5,X'110'(5)
REP 03C4 47F05014      B 20(5)
REP 03C8 E2C3D9C9,C2C2D3C5,00000000 SCRIBBLE DC C'SCRIBBLE',F'0'
REP 03D4 D207B054C336  PASSEXIT MVC WTGMODNM(8),IGG020P2
***** SAVE THE CURRENT DISK ADDRESS AND SET IT TO THE VTOC ADDRESS
REP 03DA D207B030D138  MVC 48(8,11),DXDAADDR
REP 03E0 D204D13BB23B  MVC DXDAADDR+3(5),VTOCADR
***** SAVE CURRENT DXCCW4-6 AND SET THEM TO REREAD FMT4
REP 03E6 D217B018D188  MVC 24(24,11),DXCCW4
REP 03EC D217D188B078  MVC DXCCW4(24),VTOCDSCB+24
REP 03F2 4110D188      LA 1,DXCCW4
REP 03F6 5010D128      ST 1,IOBSIOCC
REP 03FA 9200D19C      MVI DXCCW6+4,0
***** NOW RESERVE THE VTOC OF THE DISK (THIS CODE IS THE EXPANSION OF THE
***** RESERVE MACRO)
REP 03FE D70FD1C0D1C0  XC ENQAREA(16),ENQAREA
REP 0404 4110D1C0      LA 1,ENQAREA
REP 0408 92061001      MVI 1(1),6
REP 040C 96181002      OI 2(1),24
REP 0410 41E0C45A      LA 14,VTOCNAME
REP 0414 50E01004      ST 14,4(1)
REP 0418 58E0D230      L 14,DXUCBADR
REP 041C 41E0E01C      LA 14,28(14)
REP 0420 50E01008      ST 14,8(1)
REP 0424 41E0D15C      LA 14,DXDEB+32
REP 0428 50E0100C      ST 14,12(1)
REP 042C 92FFD1C0      MVI ENQAREA,255
REP 0430 0A38          SVC 56 (RESERVE)
***** INDICATE VTOC RESERVED, READ FMT4, RESET DIRF BIT AND REWRITE FMT4
***** IF NO PREVIOUS VTOC ERROR
REP 0432 96C0B255      OI DSMADTB2,VTOCR+SMCE
REP 0436 45E0C2D0      BAL RLINK,EXECIO
REP 043A 9704B06E      XI DS4VTOCI,DIRFBIT
REP 043E 9104B06E      TM DS4VTOCI,DIRFBIT
REP 0442 4780C452      BZ EXITEXIT
REP 0446 9205D198      MVI DXCCW6,X'05'
REP 044A 45E0C2D0      BAL RLINK,EXECIO
***** RESTORE DXCCW4-6 AND CURRENT DISK ADDRESS
REP 044E D217D188B018  EXITEXIT MVC DXCCW4(24),24(11)
REP 0454 D207D138B030  MVC DXDAADDR,48(11)
REP 045A 07F9          BR 9
REP 045C E2E8E2E5,E3D6C340 VTOCNAME DC C'SYSVTOC '

```

APPENDIX II

MODIFICATIONS TO SCRATCH

The modifications are expressed in SMP4 format. They apply to MVS Release 3.8 at PTF level 8001. PTF UZ25581 has been applied to CSECT IGG0290E and PTF UZ25591 to CSECT IGG0299A.

```

++USERMOD(LOCZ020) .
++VER(Z038) FMID(EDM1102) PRE(UZ25581,UZ25591) .
++ZAP(IGG0290E) .
**** ZAP TO SCRATCH TO ERASE FREED SPACE.
**** NOTE THAT CSECT IGG0299A MUST BE EXPANDED BY 360 BYTES.
NAME IGC0002I IGG0290E ***** SCRATCH *****
**** DUMMY OUT THE RESERVE ON THE VTOC
VER 02F0 0A38 SVC 56 (RESERVE)
REP 02F0 1BFF SR 15,15
VER 02F2 9640D300 OI STYPEFLG,VTOCENQ
REP 02F2 18FF18FF LR 15,15 LR 15,15
VER 02F6 96C0D381 OI DSMADTB2,VTOCR+SMCE
REP 02F6 18FF18FF LR 15,15 LR 15,15
****
****
****
++ZAP(IGG0299A) .
EXPAND IGG0299A(360)
NAME IGC0002I IGG0299A
**** DO NOT SET THE DIRF BIT OR REWRITE THE FMT4
VER 0166 9704D06E XI DS4VTOCI,DIRFBIT
VER 016A 9104D06E TM DS4VTOCI,DIRFBIT
VER 016E 4780C17C BZ SKPWR
VER 0172 9205D248 MVI CCW3,X'05'
VER 0176 9200D24C MVI CCW3+4,X'00'
VER 017A 45E0C360 BAL RETURN,EXCP10
VER 017E 9704D06E SKPWR XI DS4VTOCI,DIRFBIT
**** BYPASS WRITING DSCB 0 OVER THE LAST DSCB AND REREADING IT. INSTEAD
**** SETUP THE CHANNEL PROGRAM TO JUST READ THE NEXT DSCB
REP 0166 4110D278 LA 1,CCW9
REP 016A 5010D220 ST 1,IOB+16
**** SAVE CCW1-CCW3 IN UNUSED PART OF FMT4. THESE CCW'S READ THE FMT4
REP 016E D217D078D238 MVC VTOCDSCB+24(24),CCW1
**** SAVE THE LAST TTR AND DSORG FIELDS OF THE FMT1
REP 0174 D202D001D122 MVC 1(3,13),DS1LSTAR
REP 017A D200D000D112 MVC 0(1,13),DS1DSORG
REP 0180 18FF LR 15,15
****
****
****
**** GO SAVE THE LAST DSCB ADDRESS
VER 01E0 4780C2B2 BZ LASTDSCB
REP 01E0 47F0C56A B PATCH AREA (+56C)
****
****
****
**** AT END OF DSCB CHAIN BRANCH TO INVOKE SCRIBBLE
VER 02B4 9180D06E LASTDSCB TM DS4VTOCI,DOSBIT
REP 02B4 47F0C43E LASTDSCB B PATCH AREA (+440)
****
****
****

```

```

VER 0440 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0450 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0460 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0470 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0480 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0490 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 04A0 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 04B0 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 04C0 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 04D0 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 04E0 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 04F0 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0500 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0510 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0520 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0530 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0540 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0550 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0560 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0570 00000000,00000000,00000000,00000000 ** PATCH AREA **
VER 0580 00000000,00000000,00000000,00000000 ** PATCH AREA **
**** DON'T INVOKE SCRIBBLE IF NO EXTENTS
REP 0440 9500D301          CLI  EXTNUM,0
REP 0444 4780C492          BE   PASSEXIT
**** ESTABLISH RETURN ADDRESS
REP 0448 41E0C492          LA    14,PASSEXIT
**** SETUP PARAMETERS FOR SCRIBBLE
REP 044C 4170D300          EXTENT TABLE      LA    7,DADSMTBL
REP 0450 5880D1F8          UCB ADDRESS         L     8,WKADEB+UCBADDR
REP 0454 186D              SAVE AREA          LR    6,13
REP 0456 BF88D075          TRKS/CYL           ICM   8,8,DS4DEVSZ+3
REP 045A 41A0D2D2          DSNAME             LA    10,PDSNAME
REP 045E BFA8C486          'S'                ICM   10,8,SCRIBBLE
REP 0462 58B0D000          TTR,DSORG          L     11,0(13)
**** SIMULATE THE ICRES MACRO USED BY DADSM FOR TRANSFERRING CONTROL
REP 0466 18FD              LR    15,WRKAREA
REP 0468 900EF000          STM   0,14,0(15)
REP 046C 41100020          LA    1,X'20'
REP 0470 1BF1              SR    15,1
REP 0472 D20BD054C486      MVC   WTGMODNM(12),SCRIBBLE
REP 0478 4160D054          LA    6,WTGMODNM
REP 047C 58500010          L     5,CVTPTR
REP 0480 58505110          L     5,X'110'(5)
REP 0484 47F05014          END OF ICRES        B     20(5)
REP 0488 E2C3D9C9,C2C2D3C5,00000000 SCRIBBLE DC C'SCRIBBLE',F'0'
REP 0494 D207D054C426      PASSEXIT MVC   WTGMODNM(8),IGG0299A
**** SAVE THE LIST OF DSCB ADDRESSES TO BE DELETED AND CURRENT CCW1-CCW3
REP 049A D20FD018D090      MVC   24(16,13),VTOCDSCB+48
REP 04A0 D217D000D238      MVC   0(24,13),CCW1
**** SET CCW1-CCW3 TO REREAD FMT4
REP 04A6 D217D238D078      MVC   CCW1(24),VTOCDSCB+24
REP 04AC 9200D24C          MVI   CCW3+4,X'00'
REP 04B0 D204D34ED344      MVC   INCCHHR,VTOCADR
REP 04B6 D204D233D34E      MVC   SEEK+3(5),INCCHHR
REP 04BC 41E0D238          LA    14,CCW1
REP 04C0 50E0D220          ST    14,IOB+16
**** NOW RESERVE THE VTOC OF THE DISK (THIS CODE IS THE EXPANSION OF THE
**** RESERVE MACRO)
REP 04C4 D70FD150D150      XC    ENQAREA(16),ENQAREA
REP 04CA 4110D150          LA    1,ENQAREA
REP 04CE 92061001          MVI   1(1),6

```

```

REP 04D2 96181002      OI  2(1),24
REP 04D6 41E0C562      LA  14,VTOCNAME
REP 04DA 50E01004      ST  14,4(1)
REP 04DE 58E0D1F8      L   14,WKADEB+UCBADDR
REP 04E2 41E0E01C      LA  14,28(14)
REP 04E6 50E01008      ST  14,8(1)
REP 04EA 41E0D1F8      LA  14,WKADEB+UCBADDR
REP 04EE 50E0100C      ST  14,12(1)
REP 04F2 92FFD150      MVI ENQAREA,255
REP 04F6 0A38          SVC  56 (RESERVE)
**** INDICATE VTOC RESERVED, READ FMT4, RESET DIRF BIT AND REWRITE FTM4
**** IF NO PREVIOUS VTOC ERROR
REP 04F8 9640D300      OI  STYPEFLG,VTOCENQ
REP 04FC 96C0D381      OI  DSMADTB2,VTOCR+SMCE
REP 0500 45E0C360      BAL RETURN,EXCPIO
REP 0504 9704D06E      XI  DS4VTOCI,DIRFBIT
REP 0508 9104D06E      TM  DS4VTOCI,DIRFBIT
REP 050C 4780C516      BZ  SKIPWRT
REP 0510 9205D248      MVI CCW3,X'05'
REP 0514 45E0C360      BAL RETURN,EXCPIO
REP 0518 9704D06E      SKIPWRT XI DS4VTOCI,DIRFBIT
**** RESTORE CCW1-CCW3 WITH COMMANDS TO WRITE DSCB 0
REP 051C D217D238D000  MVC  CCW1(24),0(13)
**** GET NUMBER OF DSCB'S THAT SHOULD HAVE ALREADY BEEN DELETED. RETURN
**** TO MAINLINE IF NONE
REP 0522 4820D302      LH   2,DADSMTBL+2
REP 0526 1222          LTR  2,2
REP 0528 4780C55A      BZ   NONEDEL
**** SAVE CURRENT OUTCCHHR
REP 052C D204D028D353  MVC  40(5,13),OUTCCHHR
**** LIST OF DSCB ADDRESSES TO DELETE
REP 0532 4130D018      LA   3,24(13)
REP 0536 94BFD264      NI   CCW6+4,X'BF'
**** WRITE A DSCB 0 OVER EACH OF THE DSCB'S AND READ CHECK
REP 053A D204D3533000  LOOP MVC  OUTCCHHR,0(3)
REP 0540 41303008      LA   3,8(3)
REP 0544 D204D233D353  MVC  SEEK+3(5),OUTCCHHR
REP 054A 45E0C360      BAL RETURN,EXCPIO
REP 054E 4620C538      BCT  2,LOOP
**** INDICATE COMMAND CHAINING. THERE IS STILL 1 DSCB TO BE DELETED,
**** READ CHECKED AND THEN A DSCB 5 OR 6 TO BE READ USING THE UNMODIFIED
**** CHANNEL PROGRAM
REP 0552 9640D264      OI   CCW6+4,X'40'
**** RESTORE THE CURRENT OUTCCHHR
REP 0556 D204D353D028  MVC  OUTCCHHR(5),40(13)
REP 055C 9180D06E      NONEDEL TM DS4VTOCI,DOSBIT
REP 0560 47F0C2B6      B    +2B8
REP 0564 E2E8E2E5,E3D6C340 VTOCNAME DC C'SYSVTOC '
****
****
****
REP 056C 4780C2B2      BZ   LASTDSCB
**** SAVE THE CCHHR OF THE LAST DSCB IN AN UNUSED PART OF THE FMT4
**** FOR LATER DELETION
REP 0570 4110D090      LA   1,VTOCDSCB+48
REP 0574 48F0D302      LH   WORKREG,DADSMTBL+2
REP 0578 89F00003      SLL  WORKREG,3
REP 057C 4111F000      LA   1,0(1,WORKREG)
REP 0580 D2041000D353  MVC  0(5,1),OUTCCHHR
REP 0586 D204D353D34E  MVC  OUTCCHHR(5),INCCCHHR
REP 058C 47F0C1E2      B    ZEROUT

```

APPENDIX III

SCRIBBLE PROGRAM LISTING

SCRIBBLE START 0

```

*
* THIS ROUTINE IS CALLED FROM DASDM PARTIAL RELEASE (IGG020P2) AND
* DADSM SCRATCH (IGG0299A) TO ERASE SPACE BEING FREED BEFORE IT IS
* PUT BACK ON THE FMT5 FREE SPACE LIST.
* ON ENTRY THE FOLLOWING INFORMATION IS AVAILABLE -
*   REG 6 HAS THE ADDRESS OF A SAVE AREA
*   REG 7 HAS THE ADDRESS OF THE DADSM EXTENT TABLE
*   REG 8 HAS THE NUMBER OF TRACKS PER CYLINDER FOR THE DEVICE IN
*       BYTE 0 AND THE UCB ADDRESS IN BYTES 1 TO 3
*   REG 10 HAS 'S' IN BYTE 0 IF CALLED FROM SCRATCH OR 'R' IF CALLED
*       FROM PARTIAL RELEASE AND HAS THE DATASET NAME ADDRESS IN
*       BYTES 1 TO 3
*   REG 11 HAS THE DATASET ORGANIZATION FROM THE DS1DSORG FIELD IN
*       BYTE 0 AND THE TTR OF THE LAST BLOCK FROM THE DS1LSTAR
*       FIELD IN BYTES 1 TO 3 (FOR A SCRATCH REQUEST ONLY)
*

```

```

        USING *,12
        STM 0,14,0(6)          SAVE THE REGISTERS
        LR  13,6               ADDRESS OF CALLER'S SAVE AREA
        LR  12,15
        SR  15,15

```

```

*
* TEST FOR NON-ZERO PARAMETERS
*

```

```

        LTR  7,7               EXTENT TABLE
        BZ   BADPARM           ERROR
        CLM  8,8,=F'0'        TRACKS PER CYLINDER
        BE   BADPARM           ERROR
        CLM  8,7,=F'0'        UCB ADDRESS
        BE   BADPARM           ERROR
        CLM  10,7,=F'0'       DATASET NAME ADDRESS
        BE   BADPARM           ERROR

```

```

*
* CALCULATE LENGTH OF WORK AREA AND GET IT
*

```

```

        USING DADSM TBL,7
        SR  3,3
        IC  3,EXTNUM           NUMBER OF DATA EXTENTS
        C   3,=F'16'          ENSURE NOT MORE THAN 16
        BH  BADPARM           ERROR
        LA  5,LENDEBEX         LENGTH OF EXTENT SECTION IN DEB
        LA  6,ENDGET-WORK      BASIC WORK AREA LENGTH (1 EXTENT)
        LTR 3,3               ARE THERE ANY EXTENTS ?
        BZ   RETURN           NO - GO BACK
        BCTR 3,0              ALREADY ACCOUNTED FOR 1 EXTENT
        MR  2,5
        AR  3,6               WORK AREA LENGTH
        LA  4,OUTIOVEC-WORK    LENGTH OF NON-DEB WORK AREA
        LR  5,3
        SR  5,4               LENGTH OF DEB
        SRL 5,3               NUMBER OF DOUBLE WORDS IN DEB
        GETMAIN RC,LV=(3),SP=230,RELATED=WORK
        LTR 15,15             OK ?
        BNZ  GETERROR          NO - TERMINATE
        LR  9,1               ADDRESS OF WORK AREA

```

USING WORK,9

*

* ZERO WORK AREA

*

	LR	6,3	LENGTH
REPZERO	LA	4,256	256 BYTES AT A TIME
	CR	4,6	REMAINING AREA LESS THAN 256 ?
	BNH	ZERO	NO
	LR	4,6	YES - ZERO ONLY THIS AMOUNT
ZERO	SR	6,4	DECREASE AREA REMAINING
	BCTR	4,0	DECREMENT FOR EX
	EX	4,ZEROUT	ZERO
	LA	1,256(1)	UPDATE WORK AREA LOCATION
	LTR	6,6	ANY AREA STILL TO BE DONE ?
	BNZ	REPZERO	YES
	STH	3,WORKLEN	SAVE AREA LENGTH FOR FREEMAIN
	STCK	TIMEIN	REMEMBER TIME OF ENTRY
	DROP	7	
	ST	7,R7SAVE	SAVE REG 7
	ST	8,R8SAVE	SAVE REG 8
	ST	10,R10SAVE	SAVE REG 10
	ST	11,R11SAVE	SAVE REG 11
	EJECT		

* CONSTRUCT IOB, CCW'S, DCB AND DEB

L	4,16	GET ADDRESS OF TCB - START WITH CVT
L	4,0(4)	
L	4,4(4)	
ST	4,TCBADDR	SAVE IN WORK AREA
LA	3,MYECB	BUILD IOB
ST	3,ECBA	ECB ADDRESS
LA	3,CCW	
ST	3,CCWA	COMMAND ADDRESS
MVI	FL1,X'C2'	SET DATA,COMMAND CHAINING,UNRELATED
MVC	CCW(LENCCW),CCWD	INITIALIZE CHANNEL PROGRAM
LA	3,MYSEEK+3	SEEK ADDRESS
STCM	3,7,SEARCH+1	STORE IN SEARCH RO CCW
LA	3,SEARCH	SEARCH CCW ADDRESS
STCM	3,7,TIC+1	STORE IN TIC CCW
LA	3,SDATA	DATA ADDRESS
STCM	3,7,ERASECKD+1	STORE IN ERASE CCW
LA	3,LENSDATA	DATA LENGTH
STH	3,ERASECKD+6	STORE IN ERASE CCW
LA	3,OUTDCB	
ST	3,DCBA	DCB ADDRESS
MVC	OUTDCB(LENDDB),DCBDEB	PLACE DCB AND DEB IN WORK AREA
STC	5,DEBLEN	STORE DEB LENGTH IN PREFIX
LA	3,OUTDEB	ADDRESS OF DEB
ST	3,DCBDEBAD	STORE IN DCB
LA	3,OUTDCB	ADDRESS OF DCB
STCM	3,7,DEBDCBB	STORE IN DEB
LA	3,OUTIOVEC	ADDRESS OF APPENDAGE LIST
STCM	3,7,DEBAPPB	STORE IN DEB
L	4,R8SAVE	UCB ADDRESS
MVC	DCBDEVT,18(4)	EXTRACT DEVICE TYPE FOR DCB
OC	DCBDEVT,19(4)	
L	3,16	CVT
L	3,64(3)	ADDR OF I/O DEVICE CHAR TABLE
SR	1,1	CLEAR 1
IC	1,19(4)	DEVICE CODE
IC	1,0(1,3)	CONSTRUCT ADDRESS OF ENTRY IN ...
LA	3,0(1,3)	DEVICE CHARACTERISTICS TABLE

```

ST      3,DCBDVTBL      STORE IN DCB
USING   DADSM TBL,5
L       5,R7SAVE        ADDRESS OF DADSM EXTENT TABLE
MVC     DEBNMEXT,EXTNUM  NUMBER OF DATA EXTENTS
MVC     DEBTCBAD,TCBADDR MOVE TCB ADDRESS TO DEB
EJECT

```

* FILL IN THE EXTENT DESCRIPTIONS IN THE DEB

```

SR      3,3
SR      14,14
IC      3,EXTNUM        NUMBER OF EXTENTS
SR      2,2
IC      2,R8SAVE        NUMBER OF TRACKS PER CYLINDER
LA      4,ENTRIES       POINT AT FIRST EXTENT IN SCRTHWKA
LA      10,DEBDVMOD     POINT AT FIRST EXTENT ENTRY IN DEB
USING   DEBDVMOD,10
EXTFILL EQU *
MVI     DEBDVMOD,X'18'   FILE MASK
MVC     DEBUCBA(3),R8SAVE+1 UCB ADDRESS
LH      7,0(4)          EXTENT START TRACK
LR      11,7            SAVE
SR      6,6
DR      6,2            DIVIDE BY TRACKS PER CYLINDER
STH     7,DEBSTRCC      STORE START CYLINDER IN DEB
STH     6,DEBSTRHH      STORE START TRACK IN DEB
LH      7,2(4)          EXTENT END TRACK +1
LR      8,7            SAVE
SR      8,11           TRACKS IN EXTENT
BCTR    7,0            EXTENT END TRACK
SR      6,6
DR      6,2            DIVIDE BY TRACKS PER CYLINDER
STH     7,DEBENDCC      STORE END CYLINDER IN DEB
STH     6,DEBENDHH      STORE END TRACK IN DEB
CLC     DEBSTRCC(4),=F'0' PROTECT TRACK 0
BE      BADEXT          ERROR
CLC     DEBSTRCC(4),DEBENDCC ENSURE EXTENT IS VALID
BH      BADEXT          ERROR
STH     8,DEBNMTRK      STORE EXTENT SIZE IN DEB
AR      14,8            ACCUMULATE TRACKS ALLOCATED
LA      10,LENDEBEX(10) POINT AT NEXT EXTENT ENTRY IN DEB
LA      4,4(4)          POINT AT NEXT EXTENT IN SCRTHWKA
BCT     3,EXTFILL       GO PROCESS NEXT EXTENT
MVC     0(4,10),=X'00010001' INDICATE 1ST AND ONLY VOLUME
LR      8,14           TRACKS ALLOCATED
DROP    10
DROP    5
EJECT

```

* ADD THE DEB TO THE DEB QUEUE AND CHECK IT

```

L       3,TCBADDR      TCB ADDRESS
OC      DEBPROTG(1),28(3) STORE PROTECTION KEY IN DEB
L       4,8(3)         DEB QUEUE
LR      6,4            SAVE DEB ADDRESS
BZ      NODEB          NO DEB CURRENTLY QUEUED
O       6,DEBDEBB
ST      6,DEBDEBB      POINT TO CURRENT DEB FROM OUR'S
NODEB   LA      5,OUTDEB ADDRESS OF OUR DEB
MODESET EXTKEY=ZERO,SAVEKEY=(2)
ST      5,8(3)         STORE IN TCB
MODESET KEYADDR=(2)
DEBCHK  OUTDCB,TYPE=ADD,AM=EXCP
LTR     15,15          DEB CHECK OK ?
BNZ     BADDEB         NO

```

EJECT

* CHECK THE LAST TTR VALUE FOR SCRATCH REQUESTS

* REG 8 HAS THE NUMBER OF TRACKS ALLOCATED

	CLI	R10SAVE,C'S'	SCRATCH REQUEST ?
	BNE	CHECK2ND	NO
	TM	R11SAVE,X'80'	ISAM ?
	BZ	DSORGOK	NO
	LA	11,0	ERASE ALL TRACKS IF ISAM
	B	CHECKDEQ	CHECK IF CATALOG DEQ IS REQUIRED
DSORGOK	L	11,R11SAVE	GET TTR OF LAST BLOCK
	LA	11,0(11)	ZERO DS1DSORG BYTE
	SLL	8,8	SHIFT TRACKS ALLOCATED FOR COMPARE
	CR	11,8	COMPARE TRACKS USED WITH ALLOCATED
	BL	TTROK	TTR IS VALID
	LA	11,0	ERASE WHOLE DATASET IF TTR INVALID
	B	CHECKDEQ	CHECK IF CATALOG DEQ IS REQUIRED
TTROK	SRL	8,8	SHIFT TRACKS ALLOCATED BACK
	LTR	11,11	IS TTR ZERO ?
	BZ	CHECK1ST	YES - DATASET PROBABLY EMPTY OR VSAM
	SRL	11,8	GET TT ONLY IN REG 11
	LA	11,3(11)	SET UP TO ERASE TT+2 TRACKS (ALLOW 1
	B	COMPSize	EXTRA IN CASE EOF ON NEXT TRACK)
CHECK1ST	LA	11,1	SET UP TO CHECK IF 1ST TRACK EMPTY
	B	COMPSize	GO CHECK DATASET SIZE
CHECK2ND	LA	11,2	CHECK 2ND TRACK (IN CASE EOF ON 1ST)
COMPSize	CR	8,11	COMPARE WITH TRACKS ALLOCATED
	BH	CHECKDEQ	MORE THAN THE ONE TO BE READ
	LA	11,0	DON'T BOTHER TO READ - JUST WRITE

EJECT

* DELETIONS OCCURING AS A RESULT OF A REQUEST TO ACCESS METHOD
 * SERVICES (AMS) ENTER SCRIBBLE WITH THE OS VSAM CATALOG HELD WITH AN
 * EXCLUSIVE ENQ. TO AVOID PROLONGED LOCKOUTS OF THE CATALOG FOR LARGE
 * DELETIONS IT IS DEQ'ED PRIOR TO THE ERASURE AND RE-ENQ'D AFTER.
 * THE CATALOG MANAGEMENT ROUTINES IGGPRPLF AND IGGPRPLM ARE USED TO
 * DEQ AND ENQ THE CATALOG RESPECTIVELY. THEY ALSO CAUSE EXTRA OVERHEAD
 * RELATED TO FREEING AND REACQUIRING BUFFERS ETC.
 * BOTH ROUTINES EXPECT THE ADDRESS OF THE CATALOG COMMUNICATIONS AREA
 * TO BE IN REG 11 AND THE ADDRESS OF THE NEXT AVAILABLE 3 WORD SAVE
 * AREA FROM THE CCA IN REG 13 AND THEY DESTROY ALL REGISTERS EXCEPT
 * 11 TO 14.
 * TO DETERMINE IF THIS IS AN AMS REQUEST WE NEED TO SEE IF SVC 29
 * (DADSM SCRATCH) WAS INVOKED BY SVC 26 (CATALOG MANAGEMENT). IF SO
 * THE REGS REQUIRED (11 AND 13) CAN BE OBTAINED FROM THE SAVE AREA OF
 * THE APPROPRIATE SVRB. TO DO THIS THE RB CHAIN MUST BE TRACED. THE
 * INTERRUPT CODE THAT CAUSED THE CREATION OF THE CURRENT RB IS STORED
 * IN THE NEXT RB IN THE CHAIN, WHILE THE REGISTER CONTENTS WHEN IT
 * RELINQUISHED CONTROL ARE IN THE PREVIOUS RB IN THE CHAIN.
 * THE LINK SVC IS USED TO TRANSFER CONTROL TO IGGPRPLF AND IGGPRPLM
 * AND THIS REQUIRES BOTH TO BE DEFINED AS ALIASES OF IGGOC1A1.
 *

CHECKDEQ	DS	0H	CHECK IF CATALOG DEQ IS NECESSARY
	LR	2,8	SAVE TRACKS ALLOCATED
	CLI	R10SAVE,C'S'	SCRATCH REQUEST ?
	BNE	ERASE	NO - DEQ NOT REQUIRED
	LTR	11,11	ENTIRE DATASET BEING ERASED ?
	BZ	CHECKSIZ	YES
	LR	8,11	INITIAL NO. OF I/O'S TO BE DONE
CHECKSIZ	C	8,=F'5'	MORE THAN 5 I/O'S ?
	BNH	ERASE	NO - DON'T BOTHER WITH DEQ
	BAL	3,DEQCAT	PERFORM DEQ IF AN AMS REQUEST
	B	ERASE	START ERASURE

*
*
* THIS ROUTINE TESTS FOR AN AMS REQUEST AND FREES THE CATALOG IF SO
*

DEQCAT	DS	0H	
	L	14,TCBADDR	ADDRESS OF TCB
	LR	7,14	SAVE
	L	14,0(14)	ADDRESS OF 1ST RB IN CHAIN
TEST29	LR	15,14	
	S	15,=F'2'	ADDRESS OF INTERRUPT CODE
	CLC	0(2,15),=H'29'	LOOK FOR INTERRUPT CODE OF 29
	BNE	NEXTRB	NOT THIS ONE
	TM	10(7),X'CO'	WAS IT SVC 29 (CHAINED SVRB) ?
	BO	FOUND29	YES
NEXTRB	TM	11(14),X'80'	DOES THIS RB POINT BACK TO TCB ?
	BO	LASTRB	YES - NOT AN AMS REQUEST
	LR	7,14	NO - SAVE ADDRESS OF THIS RB
	L	14,28(14)	POINT TO NEXT RB
	B	TEST29	REPEAT SEARCH FOR SVC 29
FOUND29	DS	0H	HAVE FOUND SVC 29
	TM	11(14),X'80'	DOES THIS RB POINT BACK TO TCB ?
	BO	LASTRB	YES - NOT CALLED FROM SVC 26
	L	1,28(14)	GET ADDRESS OF NEXT RB
	S	1,=F'2'	ADDRESS OF INTERRUPT CODE
	CLC	0(2,1),=H'26'	LOOK FOR INTERRUPT CODE OF 26
	BNE	LASTRB	NOT FOUND
	TM	10(14),X'CO'	WAS IT SVC 26 (CHAINED SVRB) ?
	BNO	LASTRB	NO
	L	15,76(7)	CONTENTS OF REG 11 FROM SVRB
	CLC	0(2,15),=X'ACCA'	DOES IT POINT TO THE CCA ?
	BNE	LASTRB	NO
	STM	2,13,SAVE	SAVE REGS
	LR	11,15	ADDRESS OF CCA FOR IGGPRPLF
	L	13,84(7)	ADDRESS OF CCA SAVE AREA
	ST	11,CCA	SAVE CCA ADDRESS FOR IGGPRPLM
	ST	13,CCASAVE	SAVE CCA SAVE AREA ADDRESS
* SIMULATE THE	LINK MACRO	TO INVOKE IGGPRPLF TO FREE CATALOG	
	CNOP	0,4	
	BAL	15,*+20	BRANCH AROUND CONSTANTS
	DC	A(*+8)	ADDRESS OF PARM LIST
	DC	A(0)	DCB ADDRESS PARAMETER
	DC	CL8'IGGPRPLF'	EP PARAMETER
	LR	12,9	SAVE BASE (REG 12 NOT DESTROYED)
	SVC	6	ISSUE LINK SVC
	LR	9,12	RESTORE WORK AREA BASE
	LM	2,13,SAVE	RESTORE REGISTERS
	MVC	DEQCNT,=H'1'	INDICATE DEQ PERFORMED
LASTRB	DS	0H	
	BR	3	RETURN TO CALLER

*
*
* THIS ROUTINE INVOKES IGGPRPLM TO RESERVE THE CATALOG
*

ENQCAT	DS	0H	
	STM	2,13,SAVE	SAVE REGS
	L	11,CCA	CCA ADDRESS
	L	13,CCASAVE	CCA SAVE AREA ADDRESS
* SIMULATE THE	LINK MACRO	TO INVOKE IGGPRPLM TO RESERVE CATALOG	
	CNOP	0,4	
	BAL	15,*+20	BRANCH AROUND CONSTANTS
	DC	A(*+8)	ADDRESS OF PARM LIST

DC	A(0)	DCB ADDRESS PARAMETER
DC	CL8'IGGPRPLM'	EP PARAMETER
LR	12,9	SAVE BASE (REG 12 NOT DESTROYED)
SVC	6	ISSUE LINK SVC
LR	9,12	RESTORE WORK AREA BASE
LM	2,13,SAVE	RESTORE REGS
BR	3	RETURN TO CALLER
EJECT		

* ERASE DATA
 * REG 2 CONTAINS THE NUMBER OF TRACKS ALLOCATED.
 * REG 11 CONTAINS THE NUMBER OF TRACKS+1 TO BE ERASED INITIALLY. WHEN
 * THIS HAS BEEN DONE THE NEXT TRACK IS READ TO SEE IF IT IS ALREADY
 * ERASED. IF SO THE REMAINDER OF THE DATASET IS ASSUMED TO BE CLEAR
 * AND WILL NOT BE ERASED. HOWEVER IF THE TRACK READ IS NOT EMPTY A
 * FURTHER 30 TRACKS WILL BE ERASED AND THE NEXT READ ETC.
 *

ERASE	MVC	SDATA(LENSDATA),SDATAD	
	L	8,=X'00000000'	INITIAL TTRN
	SR	10,10	NUMBER OF TRACKS READ
EXCP	L	1,DCBDEBAD	DEB ADDR
	LR	0,8	
	LR	7,9	SAVE BASE (7 NOT DSTRYD)
	STM	2,13,SAVE	SAVE REGS
	LA	2,MYSEEK	
	L	15,16	CVT
	L	15,28(15)	TTR CONVERT ROUTINE
	BALR	14,15	
	LR	9,7	
	LM	2,13,SAVE	RESTORE REGS
	LTR	15,15	
	BNZ	CLOSE	END OF ALLOCATED EXTENTS
	XR	3,3	
	ST	3,MYECB	CLEAR ECB
	BCT	11,REISSUE	ERASE THE TRACK IF NOT DUE FOR READ

*
 * NOW PERFORM THE READ TO SEE IF THE REST OF THE DATASET IS CLEAR

	LA	10,1(10)	INCREMENT TRACKS READ
	MVI	ERASECKD,X'1E'	READ CKD CHANNEL COMMAND
	EXCP	MYIOB	READ THE TRACK
	LA	3,MYECB	
	WAIT	1,ECB=(3)	WAIT FOR READ TO COMPLETE
	CLI	MYECB,X'41'	EXPECT ERROR IF TRACK EMPTY
	BNE	ERMORE	NO ERROR - MUST CONTAIN DATA
	CLC	CSW+4(2),=X'0E00'	EXPECT UNIT CHECK ALSO
	BNE	ERMORE	NO - PROBABLY CONTAINS EOF
	CLC	SENSE,=H'8'	MUST BE NO RECORD FOUND CONDITION
	BNE	ERMORE	NO
	B	CLOSE	TRACK IS EMPTY - END ERASE
ERMORE	MVI	ERASECKD,X'11'	RESET ERASE CCW
	XC	MYECB,MYECB	CLEAR ECB
	LA	11,30	SET TO ERASE 30 MORE TRACKS
	CLI	R10SAVE,C'S'	SCRATCH REQUEST ?
	BNE	REISSUE	NO - CATALOG DEQ NOT REQUIRED
	CLC	DEQCNT,=H'0'	CATALOG ALREADY DEQUED ?
	BNE	REISSUE	YES
	SLL	2,16	SHIFT TRACKS ALLOCATED
	SR	2,8	NUMBER OF TRACKS REMAINING
	SRL	2,16	SHIFT BACK
	C	2,=F'5'	MORE THAN 5 STILL TO DO ?
	BNH	REISSUE	NO
	BAL	3,DEQCAT	YES - GO DEQ CAT BEFORE ERASING MORE

* END OF READ LOGIC

*

```

REISSUE  DS    OH
          MVC   CCHH,MYSEEK+3      MOVE SEEK ADDRESS TO COUNT FIELD
          EXCP  MYIOB              WRITE CRAP ON DATASET
          LA    3,MYECB
          WAIT  1,ECB=(3)
          CLI   MYECB,X'44'
          BE    REISSUE
          CLI   MYECB,X'7F'
          BNE   BADEXCP
          A     8,=X'00010000'      INCREMENT RELATIVE TRACK
          B     EXCP
CLOSE     DS    OH                SPACE ERASED SUCCESSFULLY
          SR    2,2                ZERO RETURN CODE
          B     PURGEDEB           GO REMOVE DEB
          EJECT
BADPARM   WTO   'SCRIBBLE - ERROR IN INPUT, SPACE NOT ERASED',      X
          ROUTCDE=(9),DESC=(3)
          LA    15,13              ERROR CODE
          B     RETURN
          SPACE 4
GETERROR  WTO   'SCRIBBLE - ERROR IN GETMAIN, SPACE NOT ERASED',    X
          ROUTCDE=(9),DESC=(3)
          LA    15,12              ERROR CODE
          B     RETURN
          SPACE 4
BADEXT    WTO   'SCRIBBLE - ERROR IN EXTENT LIST, SPACE NOT ERASED', X
          ROUTCDE=(9),DESC=(3)
          LA    15,14              ERROR CODE
          B     FREE
          SPACE 4
BADDEB    WTO   'SCRIBBLE - DEB CHECK FAILED, SPACE NOT ERASED',    X
          ROUTCDE=(9),DESC=(3)
          LA    2,15              RETURN CODE
          B     UNCHAIN            REMOVE FROM TCB DEB QUEUE
          SPACE 4
BADEXCP   WTO   'SCRIBBLE - ERROR IN CHANNEL PROGRAM, SPACE MAY NOT HAVEX
          BEEN ERASED',ROUTCDE=(9),DESC=(3)
          LA    2,8              RETURN CODE
          SPACE 4
PURGEDEB  DS    OH
          DEBCHK OUTDEB,TYPE=PURGE
          LTR   15,15              ERROR ?
          BZ    UNCHAIN            NO
          WTO   'SCRIBBLE - DEB PURGE FAILED, BUT SPACE ERASED',      X
          ROUTCDE=(9),DESC=(3)
          LA    2,1              RETURN CODE
          SPACE 4
UNCHAIN   EQU   *
          SR    4,4
          ICM   4,7,DEBDEBB+1      GET NEXT DEB ADDRESS
          L     3,TCBADDR          TCB ADDRESS
          LR    5,2              SAVE REG 2
          MODESET EXTKEY=ZERO,SAVEKEY=(2)
          ST    4,8(3)            STORE NEXT DEB ADDRESS ON TCB QUEUE
          MODESET KEYADDR=(2)
          LR    2,5              RESTORE REG 2
FREE      BAL   3,GTWRITE          WRITE GTF RECORD
          CLC   DEQCNT,=H'0'      WAS CATALOG DEQUED ?
          BE    WORKFREE          NO

```

```

        BAL      3,ENQCAT          YES - ENQ ON THE CATALOG AGAIN
WORKFREE LH      3,WORKLEN        GET WORK AREA LENGTH
        FREEMAIN RC,LV=(3),SP=230,A=(9),RELATED=WORK
        LTR      15,15           ERROR ?
        BZ       GETCODE         NO
        WTO      'SCRIBBLE - ERROR IN FREEMAIN, BUT SPACE ERASED',      X
        ROUTCDE=(9),DESC=(3)
        LA       2,2             RETURN CODE
GETCODE LR      15,2             SET RETURN CODE IN REG 15
*
* THE POSSIBLE RETURN CODES ARE
* 0 - SPACE ERASED SUCCESSFULLY
* 1 - SPACE ERASED BUT DEB PURGE FAILED
* 2 - SPACE ERASED BUT FREEMAIN FAILED
* 8 - ERROR IN CHANNEL PROGRAM AND SOME SPACE POSSIBLY NOT ERASED
* 12 - ERROR IN GETMAIN AND SPACE NOT ERASED
* 13 - ERROR IN PARAMETER INPUT AND SPACE NOT ERASED
* 14 - ERROR IN EXTENT LIST AND SPACE NOT ERASED
* 15 - DEB CHECK FAILED AND SPACE NOT ERASED
        SPACE 4
RETURN  LM      0,14,0(13)        RESTORE REGISTERS
        BR       14              AND RETURN
        SPACE 4
APPEND  BR      14              APPENDAGE ROUTINES
        EJECT
GTWRITE DS      0H              ROUTINE TO FORMAT AND WRITE GTF
        MVC      GTIMEIN,TIMEIN   PLACE TIME OF ENTRY IN GTF RECORD
        STCK     GTIMEOUT         PLACE TIME OF EXIT IN GTF RECORD
        STCM     8,12,GTNERASE    PLACE TRACKS ERASED IN GTF RECORD
        STCM     10,3,GTNREAD     PLACE TRACKS READ IN GTF RECORD
        MVC      GTNDEQ,DEQCNT    PLACE CAT DEQ/ENQ COUNT IN GTF RECORD
        SR       7,7
        IC       7,DEBNMEXT       NUMBER OF DATA EXTENTS
        STH      7,GTNMEXT        SAVE IN GTF RECORD
        MVC      GTCALLER,R10SAVE SET CALLER CODE
        STC      2,GTCOMP         SET COMPLETION CODE
        L        10,R10SAVE       ADDRESS OF DSNAME
        MVC      GTDSN,0(10)      MOVE DSN TO GTF RECORD
        L        8,R8SAVE         ADDRESS OF UCB
        MVC      GTVOL,28(8)      MOVE VOLUME TO GTF RECORD
        LR       10,7            NUMBER OF EXTENTS
        LA       4,GTEXTS        ADDRESS OF 1ST EXTENT IN GTF RECORD
        LA       8,DEBSTRCC      ADDRESS OF 1ST EXTENT IN DEB
MOVEXT  MVC      0(10,4),0(8)     MOVE 10-BYTE EXTENT FROM DEB TO GTF
        LA       4,10(4)         NEXT GTF EXTENT DESCRIPTION
        LA       8,16(8)         NEXT DEB EXTENT DESCRIPTION
        BCT      10,MOVEXT        MOVE NEXT EXTENT
        LA       4,10            LENGTH OF EACH GTF EXTENT
        MR       6,4            TOTAL LENGTH OF GTF EXTENTS
        LA       7,GTEXTS-GTREC(7) TOTAL LENGTH OF GTF RECORD
        LA       8,GTREC         ADDRESS OF GTF RECORD
        MVC      GTF(LENGTMAC),GTFMAC INITIALIZE LIST FORM OF MACRO
        GTRACE MF=(E,GTF),ID=100,DATA=(8),LNG=(7),PAGEIN=YES WRITE GTF
        BR       3              RETURN
GTFMAC  GTRACE MF=L
LENGTMAC EQU    *-GTFMAC
        EJECT
SECTOR  DC      X'00'
CCWD    CCW     X'23',SECTOR,X'60',1 SET SECTOR FOR HA
        CCW     X'31',0,X'40',5   SEARCH FOR R0
        CCW     X'08',0,0,0       TIC*-8

```

	CCW	X'11',0,X'60',0	ERASE
	CCW	X'03',0,X'20',5	NO-OP
LENCCW	EQU	*-CCWD	
SDATAD	DS	0H	
	DS	XL4 SAME AS IOBCCHH	
	DC	X'0100' R=1, KL=0	
LEN	DC	AL2(L'DATA)	
DATA	DC	C'SCRIBBLE'	
LENSDATA	EQU	*-SDATAD	
ZEROUT	XC	0(0,1),0(1)	
	EJECT		
DCBDEB	DS	0F	DCB FOR DATA BEING ERASED
	DS	17X'00'	
	DC	X'00'	
	DC	2X'00'	
	DC	F'1'	
	DC	H'0'	
	DC	X'4000'	PS
	DC	F'1'	
	DC	X'06000001'	
	DC	X'C0000000'	
	DC	H'0'	
	DC	BL2'1101000000001000'	
	DC	A(0)	
	DC	X'9200'	
	DC	BL2'1101000000001000'	
	DC	5F'0'	
	DS	0H	DEB PREFIX
	DC	A(APPEND)	
	DC	A(APPEND)	
	DC	A(APPEND)	
	DC	A(APPEND)	
	DC	A(APPEND)	
	DC	3F'0'	
	DC	X'00000000'	LENGTH OF DEB IN DOUBLE WORDS
	DS	0F	
	DC	F'0'	TCB ADDRESS
	DC	X'10000000'	NEXT DEB ADDRESS
	DC	X'60000000'	OLD DATASET
	DC	X'0F001000'	OUTPUT PROCESSING
	DC	X'00'	NUMBER OF DASD EXTENTS
	DC	3X'00'	
	DC	X'FF000000'	PRIORITY
	DC	X'0F'	THIS IS A DEB
	DC	AL3(0)	DCB ADDRESS
	DC	X'04'	DASD DEB
	DC	AL3(0)	
LEND CBDB	EQU	*-DCBDEB	
	EJECT		
WORK	DSECT		
WORKLEN	DS	H	LENGTH OF WORK AREA
DEQCNT	DS	H	NUMBER OF DEQ/ENQ'S ON CATALOG
TCBADDR	DS	F	TCB ADDRESS
R7SAVE	DS	F	REG 7 SAVE AREA
R8SAVE	DS	F	REG 8 SAVE AREA
R10SAVE	DS	F	REG 10 SAVE AREA
R11SAVE	DS	F	REG 11 SAVE AREA
TIMEIN	DS	D	TIME OF ENTRY
CCA	DS	F	CATALOG COMMUNICATIONS AREA ADDRESS
CCASAVE	DS	F	ADDRESS OF CURRENT SAVE AREA IN CCA
MYECB	DS	F	

CCW	CCW	X'23',SECTOR,X'60',1	SET SECTOR
SEARCH	CCW	X'31',0,X'40',5	SEARCH FOR R0
TIC	CCW	X'08',0,0,0	TIC*-8
ERASECKD	CCW	X'11',0,X'60',0	ERASE
	CCW	X'03',0,X'20',5	NO-OP
MYIOB	DS	0F	
FL1	DS	C	
FL2	DS	C	
SENSE	DS	H	
ECBA	DS	F	
CSW	DS	2F	
CCWA	DS	F	
DCBA	DS	F	
RESTR	DS	F	
INC	DS	F	
MYSEEK	DS	2F	
*			
SDATA	DS	0D	
CCHH	DS	XL4	SAME AS IOBCCHH
	DC	X'0100'	R=1, KL=0
	DC	AL2(0)	
	DC	C'SCRIBBLE'	
*			
SAVE	DS	12F	
GTF	GTRACE	MF=L	
OUTDCB	DS	0F	DCB FOR DATA BEING ERASED
	DS	12X'00'	
DCBDVTBL	DC	F'0'	ADDR OF ENTRY IN I/O DEV CHAR TAB
	DC	X'00'	
DCBDEVT	DC	X'00'	
	DC	2X'00'	
	DC	F'1'	
	DC	H'0'	
	DC	X'4000'	PS
	DC	F'1'	
	DC	X'060000001'	
	DC	X'C0000000'	
	DC	H'0'	
	DC	BL2'1101000000001000'	
DCBDEBAD	DC	A(0)	
	DC	X'9200'	
	DC	BL2'1101000000001000'	
	DC	5F'0'	
OUTIOVEC	DS	0H	DEB PREFIX
	DC	A(APPEND)	
	DC	A(APPEND)	
	DC	A(APPEND)	
	DC	A(APPEND)	
	DC	A(APPEND)	
	DC	3F'0'	
DEBLEN	DC	X'00000000'	LENGTH OF DEB IN DOUBLE WORDS
OUTDEB	DS	0F	
DEBTCBAD	DC	F'0'	TCB ADDRESS
DEBDEBB	DC	X'10000000'	NEXT DEB ADDRESS
	DC	X'60000000'	OLD DATASET
	DC	X'0F001000'	OUTPUT PROCESSING
DEBNMEXT	DC	X'00'	NUMBER OF DASD EXTENTS
	DC	3X'00'	
	DC	X'FF000000'	PRIORITY
DEBPROTG	DC	X'0F'	THIS IS A DEB
DEBDCBB	DC	AL3(0)	DCB ADDRESS

DC	X'04'	DASD DEB
DEBAPPB DC	AL3(0)	
DEBDVMOD DC	X'00'	
DEBUCBA DC	X'000000'	UCB ADDRESS
DEBBINUM DC	X'0000'	BIN NUMBER
DEBSTRCC DC	X'0000'	START CYLINDER
DEBSTRHH DC	X'0000'	START TRACK
DEBENDCC DC	X'0000'	END CYLINDER
DEBENDHH DC	X'0000'	END TRACK
DEBNMTRK DC	X'0000'	NUMBER OF TRACKS
LENDEBEX EQU	*-DEBDVMOD	LENGTH OF EXTENT DESCRIPTION
DC	11F'0'	
ENDGET EQU	*	
SPACE	4	
ORG	OUTDCB	
GTREC DS	0D	GTF RECORD FORMAT
GTIMEIN DS	D	TIME OF ENTRY TO SCRIBBLE
GTIMEOUT DS	D	TIME OF EXIT
GTCALLER DS	C	SCRIBBLE CALLER CODE (S OR R)
GTCOMP DS	C	SCRIBBLE COMPLETION CODE
GTDSN DS	CL44	DSNAME
GTVOL DS	CL6	VOLUME SERIAL
GTNERASE DS	CL2	NUMBER OF TRACKS ERASED
GTNREAD DS	CL2	NUMBER OF TRACKS READ
GTNDEQ DS	CL2	NUMBER OF DEQ/ENQ'S ON CATALOG
GTNMEXT DS	CL2	NUMBER OF EXTENTS RELEASED
GTEXTS DS	0C	UP TO 16 10-BYTE EXTENT DESCRIPTS
SPACE	10	
DADSMTBL DSECT		DADSM EXTENT TABLE
DS	C	
EXTNUM DS	C	NUMBER OF EXTENTS IN TABLE
DS	2C	
ENTRIES DS	16F	UP TO 16 EXTENTS
EJECT		
END		

DOCUMENT CONTROL DATA SHEET

Security classification of this page

UNCLASSIFIED

1 DOCUMENT NUMBERS

AR
Number: AR-002-005Report
Number: ERL-0143-TROther
Numbers:

2 SECURITY CLASSIFICATION

a. Complete
Document: Unclassifiedb. Title in
Isolation: Unclassifiedc. Summary in
Isolation: Unclassified

3 TITLE

AUTOMATIC ERASURE OF RELEASED DISK SPACE ON AN IBM 370 COMPUTER WITH THE
MVS OPERATING SYSTEM

4 PERSONAL AUTHOR(S):

J.C. Gwatking

5 DOCUMENT DATE:

June 1980

6 6.1 TOTAL NUMBER
OF PAGES 376.2 NUMBER OF
REFERENCES: 13

7 7.1 CORPORATE AUTHOR(S):

Electronics Research Laboratory

7.2 DOCUMENT SERIES
AND NUMBERElectronics Research Laboratory
0143-TR

8 REFERENCE NUMBERS

a. Task: DST78/44

b. Sponsoring
Agency:

9 COST CODE:

228801

10 IMPRINT (Publishing organisation)

Defence Research Centre Salisbury

11 COMPUTER PROGRAM(S)
(Title(s) and language(s))

12 RELEASE LIMITATIONS (of the document):

Approved for public release

12.0	OVERSEAS	NO		P.R.	1	A		B		C		D		E	
------	----------	----	--	------	---	---	--	---	--	---	--	---	--	---	--

Security classification of this page:

UNCLASSIFIED

13 ANNOUNCEMENT LIMITATIONS (of the information on these pages):

No limitation

14 DESCRIPTORS:

a. EJC Thesaurus
TermsComputer storage devices
Magnetic disks

15 COSATI CODES:

0902

b. Non-Thesaurus
Terms

Automatic erasure

16 LIBRARY LOCATION CODES (for libraries listed in the distribution):

17 SUMMARY OR ABSTRACT:

(if this is security classified, the announcement of this report will be similarly classified)

The standard MVS operating system supplied for use on IBM 370 computers does not erase or protect data residing on areas of disk that have been released for reuse. This report discusses the problem and describes efficient techniques that have been developed and installed at the DRCS which automatically erase all data when disk space is released by its owner. This has involved modifying the disk space management software and included some restructuring to eliminate contention for system resources while data is erased.