164 MVS update

May 2000

In this issue

MVS Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 33598
From USA: 01144 1635 33598
E-mail: Jaimek@xephon.com

North American office

Xephon/QNA
PO Box 350100,
Westminister, CO 80035-0100
USA
Telephone: 303 410 9344

Contributions

If you have anything original to say about MVS, or any interesting experience to recount, why not spend an hour or two putting it on paper? The article need not be very long – two or three paragraphs could be sufficient. Not only will you be actively helping the free exchange of information, which benefits all MVS users, but you will also gain professional recognition for your expertise, and the expertise of your colleagues, as well as some material reward in the form of a publication fee – we pay at the rate of £170 ($250) per 1000 words for all original material published in *MVS Update*. If you would like to know a bit more before starting on an article, write to us at one of the above addresses, and we'll send you full details, without any obligation on your part.

Editor

Jaime Kaminski

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, EXECs, and other contents of this journal before making any use of it.

MVS Update on-line

Code from *MVS Update* can be downloaded from our Web site at http://www.xephon.com/mvsupdate.html; you will need the user-id shown on your address label.

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Printed in England.

A REXX routine to convert PDS to sequential datasets

INTRODUCTION

I recently had reason to use an easy function for converting a partitioned dataset to a sequential dataset, so that I was able to utilize the following program for file transfer (FT) Host/PC.

Some articles in *MVS Update* have previously described PDS/ download facilities using REXX or CLIST; this simple REXX procedure is my solution. It is functionally the same as these previous utilities, and it is compatible with any method of file transfer (FT).

Usually I work with a simple PC 3270 emulation program that supports FT's function. It accepts a PDS , if the members are requested individually. This makes it difficult to transfer a whole PDS. I wrote this REXX code that aids the data migration process from host to PC and *vice versa*. The code is used in the following way:

- It is executed as a TSO command and it accepts two parameters:

  – parm 1 – hlq.pdsname
  – parm 2 – On/Off for TRACE On/Off;

- The output is a sequential file (userid.seqfile.expds), which will be the input for your FT function. My code writes as output all the members in sequence order.

- If no error conditions exist, the sequential file can be copied from to PC platform to host. If required the JCL below can be used to construct the original PDS.

JCL

```
//........ JOB ...................
//** ———————————————————————— **
//STEPØØ      EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT    DD   SYSOUT=*
//SYSIN       DD   DISP=SHR,DSN=userid.seqfile.expds
//SYSUT2      DD   DISP=SHR,DSN=your.output.pds
```

```
/* ——————————————————— REXX ** —————————————————— */
/* Syntax for expds:                                 */
/*    %expds hlq.pdsname off                         */
/* ———————————————————————————————————————————————— */
  ARG libparm ver
  IF ver = ON THEN
      TRACE ALL
  user = USERID()
  temp_grc = "ØØ"
  tot_mem  = Ø
  CALL read_dir
  CALL all_utl
  DO I = 7  TO libmem.Ø
       MEMBER = STRIP(libmem.I)
       CALL copy_mem
  END

  CALL end_expds
/* ———————————————————————— */

all_utl:

  "LISTDS "user".SEQFILE.EXPDS"
       IF RC = Ø  THEN DO
                " DEL "user".SEQFILE.EXPDS"
                      END

  " ALLOC FI(OUT) DA("user".SEQFILE.EXPDS) MOD CATALOG",
  " RECFM (F B) DSORG(PS) SP(5 5) CYL  ",
  " LRECL(8Ø) BLKSIZE(8ØØØ) UNIT(CKPT) "
       IF RC > Ø THEN
         DO
           temp_rc   =  RC
           temp_ftc  =  "ALLOC  ERROR"
           CALL rou_rc
         END
RETURN

copy_mem:

  tot_mem = tot_mem + 1
  ROUT. = "./       ADD  LEVEL=ØØ,SOURCE=Ø,NAME="MEMBER
  "EXECIO * DISKW OUT (FINIS STEM ROUT."
       IF RC > Ø THEN
         DO
           temp_rc   =  RC
           temp_ftc  =  "EXECIO ERROR"
           CALL rou_rc
```



```
          END

  " REPRO INDATASET("libparm"("MEMBER") )  OUTFILE(OUT) "
       IF RC > Ø THEN
         DO
           temp_rc   =  RC
           temp_ftc  =  "REPRO  ERROR"
           CALL rou_rc
         END
RETURN

read_dir:
  X = OUTTRAP("libmem.")
  "LISTDS "libparm" M"
       IF RC > Ø THEN
         DO
           temp_rc   =  RC
           temp_ftc  =  "LISTDS ERROR"
           CALL rou_rc
         END
  X = OUTTRAP("OFF")
       IF libmem.Ø < 8 THEN
         DO
           temp_ftc = "PDS IS EMPTY"
           temp_rc  = 4
           CALL rou_rc
         END
RETURN

rou_rc:

       IF temp_rc > Ø  THEN
          temp_grc = temp_rc
          SAY " > ERROR         < "
          SAY " > FUNCTION/REXX < " temp_ftc
          SAY " > RETURN CODE   < " temp_rc

  temp_rc  = Ø
  temp_ftc = ""

  CALL end_expds
RETURN

end_expds:

  " FREE DATASET("user".SEQFILE.EXPDS)"

       IF temp_grc = Ø THEN DO
             SAY "MEMBERS SUCCESSFULLY PROCESSED: "tot_mem
                          END
EXIT  temp_grc
```

Assessing the performance of MVS I/O systems

INTRODUCTION

This article reviews a number of installations to determine I/O performance values and average and median statistics for I/O response times, provides a new data filtering rule based on CONN time values, and examines some tuning data to reveal the tuning problems encountered by installations.

In papers, Joe Major[1, 2] compared a considerable number of MVS installations and evaluated performance and capacity planning parameters with a view to establishing relationships between them. This chapter reviews a number of installations with the intention of determining current I/O performance parameter values. Once the range of customary values achieved is determined, they can then be used for capacity planning, design, and setting future objectives. It is also possible that relationships among them can be established. These parameters also yield an idea of how much tuning is still required in the I/O area and where the emphasis should be. With these objectives in mind, measurement data of recent vintage for 12 installations is examined.

The article determines current average and median statistics for I/O response times indicating current usage. The metrics derived, such as access density and I/O content, can be used in capacity planning. The chapter provides a new data filtering rule based on CONN time values and, finally, examination of some tuning data reveals the prevalent tuning problems encountered by installations.

In examining the performance of MVS DASD I/O subsystems, there is one key question to answer: what is the I/O response time, RT, that is achieved? This response time (naturally) is the function of expectations, need, the current level of technology, and various political factors, for example, how much influence the storage manager has on his corporation's IT spending, the depreciation criteria, whether DASD is leased or purchased. Our interest is only the current level of performance, whether it can be improved, and, if so, how? Also, we are interested in the inter-relationship of these parameters for design and capacity planning reasons.

Measurement data was solicited from two sources: from members of the SHARE MVS group and from students participating in the author's performance class. This selection, of course, is not scientific and is somewhat biased towards leading-edge customers. The data solicited was RMF postprocessor output data (listings normally printed), which was reduced using the RMF Spreadsheet Reporter. While the Spreadsheet Reporter does provide very usable output, unfortunately the combination of RMF and the reduction tool suffer from some shortcomings, eg, RMF reports on Logical Control Units (LCUs) and physical control units separately. The LCU number and SSID number are not tied together in the reporting tool.

Also, the identity of the control units (whether from EMC, HDS, IBM, or STK) is not currently identified or reported. Another shortcoming of RMF is that it does not report the actual size of a logical volume (which may be smaller than an emulated 3380 or 3390), it just reports whether the emulated volume is a 3380 or 3390. Because of this limitation the actual volume size is assumed to be that of the reported volume, eg, 2.83GB for a 3390-3. If the logical volume is actually *smaller,* then the access density, which is the ratio of I/O rate to installed space, reported may be too small.

Measurement data originated from 12 separate installations. In many cases, the installation had multiple CPUs sharing DASD and there the load on the DASD was combined. (In other words, the load was viewed from the viewpoint of the logical volume, rather than from the CPU, which is how RMF normally reports.) There were a few installations in which data was available for only one key CPU, not all CPUs.

Highlights of the data are tabulated in Figures 8-13 at the close of this article, together with definitions for each type of data. The data categories are identified by line numbers in Figures 8-10, eg, (16) is response time.

SYSTEM-WIDE RESPONSE TIME DATA, RT

The average, median, maximum, and minimum statistical values for system-wide response times and its components are listed in Figure 1.

	Average	Median	Maximum	Minimum
CONN	3.46	2.28	17.28	1.27
DISC	2.58	1.97	4.95	0.96
PEND	0.93	0.86	2.66	0.17
IOSQ	1.31	1.19	4.60	0.07
RT	6.62	5.71	11.26	2.61

Figure 1: System-wide response time statistics

Since 1998, I have been advocating[3] a guideline that response time, RT, should *not* exceed 7ms *overall* or for a particular logical volume. Examining the data collected, both the average and median response times were indeed around 7ms. A further recommendation, examined later, was that the average (or individual volume) response time to service time ratio, K, should be less than 1.5, ie K = RT / ST < 1.5. The purpose of this rule was to avoid excessive queueing. The combination of the two rules leads to the additional rule that the service time, ST, should not exceed about 4.7ms, ie, ST < 4.7ms, allowing for the wait time, W = IOSQ + PEND a maximum of 2.3ms. These rules were established empirically.

Examining the average and median response time data collected, RTav and RTmd both were indeed approximately 7ms and STav and STmd both were close to 4.7ms. Unfortunately, six installations had system-wide average response times higher than 7ms. In the tuning section we will briefly examine what can be done to improve their responsiveness.

Last year, my recommendation was to achieve HR = 92% (MR = 8%), leading to a DISC = 1.2-2ms. DISC = 2-2.5ms seen here corresponds to an overall hit ratio of approximately HR = 87.5-90%. DISCav = 2.54ms shown here seems to indicate that miss ratios are still somewhat high (or cache sizes are too small). Indeed in the tuning section, it is clearly seen that inadequate cache hit ratios may be a problem.

With the reduction in protocol times for new storage processors, with a concomitant increase in cache sizes, it is perhaps appropriate to reduce the response times, RT, further this year. Some reviewers question the business reason and rationale for this response time reduction. The dramatic revolution in storage processors (largely due to ever-increasing cache sizes) has been producing a steady reduction in response times (in two or three years maybe from 10ms to the current 7ms). Furthermore, the ever larger processing needs for storage and storage accesses demands ever better response times, which, luckily, technology is capable of providing. Hence, in my opinion, a slightly ambitious objective of an (average) response time, RT = 5ms, should now be set. This target can be achieved by obtaining a service time of ST = 3.4ms and the following component times:

```
IOSQ < 1ms, PEND < 0.6ms, CONN < 2ms, DISC < 1.4ms
```

Only installations 2 and 10 have Kav values higher than the recommended maximum value of 1.5 and they are tuning candidates as they also have high response times. Note, however, that Kav by itself is *not* sufficient to say that the installation should be tuned, since, for example, installation 3 has a very low response time of RT = 4.43ms in spite of a fairly high value of Kav = 1.49.

HIT RATIONS, HR

Overall hit ratio data was available only for eight installations. The author has been advocating a hit ratio, HR, of 92% or better.

Six installations out of eight achieved that or better. Hence perhaps the 92% hit ratio is still a reasonable objective for this year with improvements suggested moving towards perhaps the value of 94% (quite often seen today).

ACCESS DENSITY, AD

In their CMG papers, Joe Major[1] and Bruce McNutt described their conclusion that access densities have experienced a steady decline and projected a further such decline for future years (12% per annum initially and 23% per annum since 1994). Well, the decline has indeed occurred. Figure 2 shows the statistical summary for the access density data. The average access density is now 1.12 and the median

Average	Median	Maximum	Minimum
1.12	0.83	2.21	0.45

Figure 2: System-wide access density statistics

0.83. Thus, assuming the continuation of the trend, we can assume a further reduction in average access density – perhaps decreasing to approximately 0.9 for this year. The calculations assume that the logical volumes used are full size, which is not necessarily the case. If in fact they were smaller, then access density would be higher. The logical volume size reduction *may* counteract the reduction trend observed in access density. Note that the storage capacity actually installed in the installations examined ranged from 19TB to 260GB, a very *wide* ratio of 73:1.

AVERAGE I/O ACTIVITY RATE, SAV

Figure 3 shows average logical volume I/O rates statistics. These are remarkably consistent; they range between 1.41 and 6.17 with an average of 3.07 I/Os per second. This value seems to represents an appropriate first-cut design parameter, ie one could assume that each logical volume (in any system) will have approximately 3 I/Os per second on average, but note the impact of the law of diminishing returns (discussed later).

Average	Median	Maximum	Minimum
3.16	3.07	6.17	1.41

Figure 3: Average I/O activity rates statistics

AVERAGE SERVICE TIME INTENSITY, SIAV (19)

Figure 4 shows average service time intensity, SIav, statistics. The maximum value of the service time intensity is 23.29ms/s, or a device utilization of 2.329%, with the *average* value hovering around 1%.

Average	Median	Maximum	Minimum
13.42	11.89	23.29	4.19

Figure 4: Average service time intensity statistics

This means that typically the average logical volume utilization in a system is expected to be around 1%.

Plotting service time intensity, SIav, on Figure 5 against the number of logical volumes, N, (unfortunately there are too few measurement points for this graph) demonstrates the 'law of diminishing returns' identified by Joe Major. It appears that while below 1,000 logical volumes the average service time intensity may vary between 7 and 25ms/s, (or the average device utilization between 0.7 and 2.5%), at approximately 1,000 logical volumes it is reduced to a value of 5-10ms/s, and further reduced when the number of logical volumes reaches 6,000. With more data a more accurate guideline could be established.

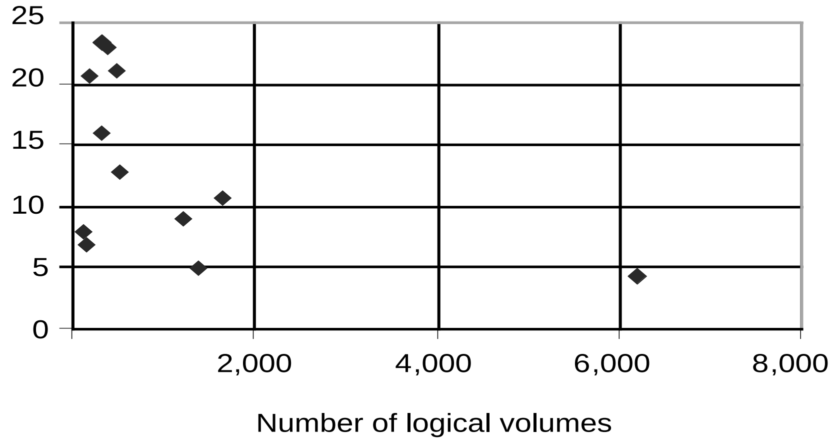

Figure 5: Average service time intensity

SYTEM-WIDE TUNING CONSIDERATIONS

Earlier, I observed that six installations had system-wide average response times higher than 7ms. Installations 6, 9, and 11 have DISC times higher than 4ms, causing the high response times. For these installations it is clear that more cacheing might be an appropriate solution.

For installation 10, CONN time is a high of 3.56ms, which necessitates further and more detailed examination (see below). For installation 4, IOSQ is a somewhat high 2.05ms, leading to a total wait time of 3.05ms. The clear path to response time improvement lies here in reducing the IOSQ time. For installation 2, both IOSQ and DISC times are high. Since it is easier to solve the IOSQ problem than the DISC time problem, probably the IOSQ time should be considered first.

SKEW CONSIDERATIONS

Again, it was Joe Major about ten years ago who established skew as the best guideline for determining whether a DASD installation should be tuned. In his paper he determined that if the observed value of skew (the ratio of maximum and average service time intensities or device utilization) exceeds the average skew determined by experience (depending on the number of logical volumes) then tuning *may* be in order.

According to this indicator, the SKEW value (23), nine out of the 12 installations require tuning. Installations 5, 6, and 12 do not appear to require (much) tuning. In our environment we can conclude that we cannot rely exclusively upon this indicator as a sole indicator showing no need for tuning, eg, we know that installation 6 *does* require tuning. The indicator tells us that there is a fair imbalance as far as the usage of the logical volumes is concerned. However, if the overall response times goals are met, this considerations can be ignored.

I/O CONTENT, R

We designate the total MIPS available in a system by M (27). This is obtained by adding the MIPS for each CPU, Mcpu, ie, M = Sum(Mcpu). MIPS used (28), MU, is determined by multiplying the MIPS for each

CPU, Mcpu, by CPU utilization, Bcpu, for that CPU, and adding the total for each CPU for the complex, ie:

```
MU = Sum (Mcpu x Bcpu)
```

I/O content, R, was defined by Joe Major as a key characteristic of systems. This is defined as I/O rate, S, divided by the MIPS used, MU. Thus:

```
S = MU x R
```

The average value of I/O content and R statistics, shown in Figure 6, is meaningless here. Out of nine values available, four values can be rounded to 3 while four values can be rounded to 5 and there is a single installation with a very unusual value of 53. We can, therefore, assume that I/O content should be between 3 and 5. On the basis of past experience, expectation is that I/O content is decreasing in the long run as more and more processor storage may be used to eliminate I/Os.

Average	Median	Maximum	Minimum
9.21	3.76	53.13	2.47

Figure 6: I/O content, R statistics

I/O content can be used for capacity planning inasmuch that I/O content characterizes a system and its workload.

TUNING CONSIDERATIONS BASED ON LOGICAL VOLUMES WITH THE HIGHEST I/O INTENSITY, I

In order to identify logical volume candidates for tuning, in an earlier paper[3] the author stated that I/O intensity, I, should be evaluated for each logical volume and should be ordered according to the magnitude of I. One exception is that volumes with response times below the installation’s target, whether that is 7, 8, or 10ms, should be excluded from this consideration, since they meet the response time target.

Further, all logical volumes with an I/O rate, S, less than 1 or 2 I/Os per second should also be excluded, since they often include statistically-flawed measurements or they are mostly irrelevant to the overall performance of a system. This exclusion requires filtering of the data.

Using this methodology, the top three volumes with the highest I/O intensity are listed for each installation (see Figures 11-13). For each such volume, the percentage of system-wide I/O intensity represented by that particular volume's I/O intensity is calculated, as is the sum of percentages for the top three volumes. Three volumes represent between 0.05% and 3% of the totality of volumes in the systems, but this represents typically about 10% of the total I/O intensity for the system. (To put it differently, tune three volumes in a system and you impact about 10% of the system's I/O responsiveness.)

LOGICAL VOLUMES WITH THE HIGHEST I/O INTENSITY

In the system in Figure 7, we find that I/O intensity for these volumes exceeds 200ms/s (in all but one of the cases) or exceeds 300ms/s (in nine out of 12 cases). Accordingly one may establish the guideline that if a volume's I/O intensity, I, exceeds 200-300ms/s (assuming that the filtering criteria described earlier are met) then these volumes are worth examining as candidates for tuning.

	Average	Median	Maximum	Minimum
I	775	497	327	453
RT	34.98	15.25	175.09	10.00

Figure 7: Response time and I/O intensity statistics for logical volumes with the highest I/O intensity for any given system

Examining the key (the largest) component of the response time DISC time appears to be the problem for installations 1, 4, 7, and 11. IOSQ appears to be the problem for 2, 3, 8, 9, and 12. Finally, CONN time appears to be the problem for installations 5, 6, and 10.

The latter case of high CONN leads to another interesting conclusion, because high CONN time is typically *not* a problem, unless the installation still has some parallel channels. If that is not the case, then if a volume has I/O intensity, I, (ie high response time, RT) due primarily to high CONN time, then the volume should not be considered a prime tuning candidate. Therefore, perhaps an additional filtering criterion should be introduced. Given that we filter out all volumes with (say) RT < 7ms we should additionally filter out volumes if RT-CONN < 5ms.

```
(Alternatively, DISC + PEND + IOSQ < 5ms)
```

This method implicitly allows 2ms for CONN time and results in ignoring volumes where high response times are due only to high CONN times. If this additional filtering were activated, then the volume with the highest I/O intensity in cases 5, 6, and 10 would not be a tuning candidate. Note that long CONN times usually occur on, for example, DB2 volumes owing to scanning operations and on volumes with sequential access. Usually no tuning action is required for these volumes.

It appears appropriate now to replace the volume with the highest I/O intensity (and high CONN time) for installations 5, 6, and 10 with the next highest I/O intensity volume. Then, for installations 5 and 6 the key component becomes DISC time, while for installation 10 it is IOSQ time.

High DISC time is normally caused by low hit ratios. Let us examine installations 1, 4, 5, 6, 7, and 11 for the cause of high DISC time. In installation 1, volume SMS263 is a volume with 100% read hit ratio and all reads, consequently high DISC time is caused by something else, eg internal path contention in Iceberg/RVA or in Symmetrix.

In installation 4, volume ZINF02 (a write-only volume) appears not to be cached at all, which is certainly a good reason for high DISC times. In installation 5, there appears to be a reporting problem for volume DBAP05 since total hit ratio is reported as 0. Alternatively, this volume is not cached at all.

Volume ARGP9V in installation 6 has a read hit ratio of only 49% – certainly accounting for its high DISC time of 7.3ms. Installation 7 has not provided cacheing data, thus we just have to assume that the high DISC time of 14.4ms is due to low cacheing hit ratio. In installation 11, the high DISC time of 8.5ms for volume BZKR04 is amply explained by a read hit ratio of 39% with 91.5% of all I/Os due to reads.

High IOSQ time is the culprit in installations, 2, 3, 8, 9, 10, and 12. The remedy is to reduce the load on the logical volume. This can be done by removing some dataset from the volume (if there is more than one dataset) or by making the logical volume smaller.

Looking at the high I/O intensity logical volumes (having used the filtering actions described), one can conclude that problems are usually caused by high IOSQ or DISC times. Typically, solving the IOSQ problem is much easier because it involves only data movement. Solving the DISC time problems may involve the need for either increased cache sizes or more extensive data movement.

Looking at the top three volumes with high I/O intensities, one might conclude that solving their problems would yield significant benefits to the installations and minimize the tuning work involved. Note, however, that further tuning analysis most likely requires analysis of the data at the *dataset level*, ie use of SMF 42-type records.

SUMMARY

Having examined performance data for 12 installations, this article has established (capacity) planning guidelines for system-wide response time and its components:

- Overall hit ratio
- Access density
- Average logical volume I/O activity rate
- Average service time intensity (device utilization)
- Response time to service time ratios and I/O content.

It has also illustrated system-wide tuning considerations and specifically addressed installation tuning and its processes using statistics for the logical volumes in a system with the highest I/O intensity.

CONCLUSIONS

Some reviewers questioned whether it is possible to make generalizations on the basis of only 12 installations and, secondly, whether the sample selected (typically one RMF period in a peak period) is adequate for this purpose. In response, of course, it would be *much* better to have several hundred measurement points, ie several hundred installations. But, Joe Major's referenced papers and my experience demonstrate that the metrics (such as relative I/O content and access density) do characterize an installation. More specifically, I/O content is relatively constant (within approximately 10%) in a given time frame (about six weeks or so) and is independent of the actual load observed.

Access density and tuning parameters are load dependent but, having selected *peak* periods, reasonably representative numbers should have been obtained. Finally, the actual range of parameters obtained by themselves are revealing. Without any (conscious) bias of selecting installations, the value range of the characterizing parameters is quite narrow, seemingly indicating that representative numbers can be obtained from a relatively small number of installations.

Examining the response time data for the installations available seemed to confirm the guidelines previously activated and pointed in the direction of further response time reductions.

Please submit your own measurement data to the author, to enrich the sample!

REFERENCES

1 Major, J, *The CPU-Memory Equation*, CMG '90, pp 122-135

2 Major, J, *Resource Usage Metrics and Trends: Host Systems*, CMG '92, pp 76-86

3 Beretvas, T, A paper

EXPLANATION OF ITEMS IN THE TABLES

A parenthetical number in the following refers to a line number in the following tables.

The number of SSIDs (1) is self-explanatory and refers to the number of control unit subsystems recognized by the system. However, this does *not* really represent the number of physical control units. The number of LCUs, L (3), is a better measure of that.

HR (2) is the overall hit ratio and is designated as the ratio of the number of hits to the number of I/O events. A read hit ratio is defined similarly, but only for reads.

N (4) represents the number of logical volumes recognized by MVS.

DG (5) is measured in GB and represents a number calculated by multiplying the number of logical volumes (4) by the associated size for that particular emulated logical volume type (eg 2.83GB for a 3390-3).

AD (6) is access density and is the ratio of DASD gigabytes (5) and number of logical volumes (4).

Activity rate (I/Os per second) (7-9) represent the total, S, the highest, Smx, and average, Sav I/O rates in the system. Sav = S / N

```
Highest / average (10) = Smx / Sav
```

RT (16) is the average DASD response time and is the sum of four quantities:

```
RT = IOSQ(15) + DISC(12)+ CONN(11) + PEND(14)
```

Connect time is part of an I/O event that occurs when a logical volume is connected to the path. Connect time includes data transfer and protocol, times. The average connect time is designated CONN (11) and is the sum of the average protocol and transfer times: CONN = PR + XFER.

In contrast, when the logical volume is busy but *not* connected to the path it encounters disconnect time. The logical volume busy means that an I/O event is in progress and the storage processor is busy internally with this event. A disconnect time, DISCm delay occurs for *each* miss. (In most storage processors only read misses exist.) The *average* disconnect time, DISC (12), is calculated as the *weighted average* of disconnect times for hits (0) and misses.

ST (13) is the average service time and can be defined as the sum of the average disconnect time and connect time:

```
ST = DISC + CONN
```

W, the average wait time, is defined as the sum of the average IOSQ and PEND times:

```
W = IOSQ + PEND
```

PEND (14) is the average pend time. Pend time delay occurs if *all* paths to a logical volume are busy when the I/O request is attempted. It also occurs when a reserve has been issued against the volume on another system.

IOSQ (15) time arises because of an MVS limitation that states that each *logical* volume can have only one outstanding request against it at any time. Any subsequent request must wait for the completion of a previous request and is queued on the UCB queue. The length of time accumulated on this queue is reported as IOSQ time.

The ratio of average response time to average service time, K, or response time to service time ratio is defined as: K (17) = RT (16) / ST (13), is a measure of how much queueing occurs (ie, PEND and IOSQ times). This is applicable system-wide or for a particular logical volume. This value can be considered system-wide for a control unit or for a logical volume. A rule-of-thumb, ROT, has been used for some time indicating that K should *not* exceed 1.5. If it does, that usually indicates excessive queueing.

In what follows it is assumed that all values are averages, ie the average value of that particular parmeter obtained for a measurement interval.

I (18) is the I/O intensity and is defined as the product of the I/O rate and response time. Thus, the system-wide I/O intensity is I(16) = S (7) x RT (16). The I/O intensity for a particular logical volume is analogously the product of the I/O rate pertaining to the particular volume and its response time.

SI (19) is the service time intensity and is the product of the system-wide service time and I/O rate: SI (19) = ST (13) x S (7). Service time intensity can be obtained for a control unit or for a logical volume also. The value obtained for a particular logical volume is also known as device utilization (percentage).

PI (20) is the path intensity and is the product of the system-wide connect time and I/O rate: PI (20) = CONN (11) x S (7). Similarly for a control unit or logical volume.

SIav (21) is the average service time intensity and is the system-wide service time intensity divided by the number of logical volumes:

```
SIav (21) = SI (19) / N (4)
```

SImx (22) is the maximum service time intensity and is the highest service time intensity for a logical volume in a system during the measurement interval.

Max/Avg (Skew) (23), or SKEW, is the ratio of maximum and average service time intensities.

SKEW (23) = SImx (22) / SIav (21). This value gives a measure of how well balanced the system is – ie how well the load is distributed across the logical volumes.

Based on past experience Typical average value (24), Typical low value (25), and typical high value (26) were established by Joe Major as a function of the number of logical volumes, N.

If SKEW is in the range of typical high value, or exceeds it, this is an indication that tuning activities, ie rebalancing logical volume loads should occur.

M (27) is the sum of all MIPS in the complex. M = Sum (Mcpu)

MU (28) is the MIPS used, For each CPU multiply its MIPS, Mcpu, with the CPU utilization, B, and sum it for the complex: MU = Sum (Mcpu x B)

R (29) is the I/O content, which is the ratio of total I/O rate, S, and MIPS used: S = MU x R

Installation number:		**Line**	**1**	**2**	**3**	**4**
Overall	Number of SSIDs	1	9	25	100	20
Values	Hit Ratio, HR	2	0.88	0.92	0.94	1200
	Number of LCUs, L	3	3	20	29	3587.71
	No of DASDs, N	4	495	1616	6204	0.65
	DASD GB, DG:	5	1311	4359	19327	2322.27
	Access Density, AD	6	1.13	0.69	0.45	106.42
Activity	Total, S	7	1499	3003	8731	2322
Rate	Highest, Smx	8	13.17	129.48	116.41	33.43
(I/Os/sec]	Average, Sav	9	2.99	1.86	1.41	1.94
	Highest / Average	10	4.41	69.67	82.72	17.28
Times	CONN Time	11	2.47	2.31	1.54	2.67
[ms]	DISC Time	12	1.82	3.49	1.44	2.07
	Service Time, ST	13	4.30	5.79	2.98	4.75
	PEND Time	14	0.55	0.85	0.93	1.00
	IOSQ Time	15	0.10	4.60	0.52	2.05
	Response Time, RT	16	4.93	11.24	4.43	7.85
	RT to ST Ratio	17	1.15	1.94	1.49	1.65
Intensities	I/O Intensity I	18	7389	33757	38649	18240
[ms/s]	ST Intensity SI	19	6437	17403	25990	11021
	Path Intensity, PI	20	3704	6929	13427	6209
DASD	Avg. ST Intens: SIav	21	13.00	10.77	4.19	9.18
Skew	Max. ST Intens:SImx	22	259.23	723.40	577.51	388.44
	Max / Avg (Skew)	23	19.94	67.17	137.86	42.29
	Typical Avg. Value	24	15.33	25.81	46.64	22.64
	Typical Low Value	25	9.33	14.29	23.19	12.84
	Typical Hi Value	26	25.19	46.60	93.80	39.92
I/O Content	MIPS available, M	27	456	845	173	729
	MIPS used, MU	28	399	621	164	686
	I/O Content, R	29	3.76	4.84	53.13	3.39

Figure 8: The system data for the first four installations

Installation number:			5	6	7	8
		Line				
Overall	Number of SSIDs	1	6	10		24
Values	Hit Ratio, HR	2	0.98	0.72		0.95
	Number of LCUs, L	3	5	7	6	24
	No of DASDs, N	4	155	498	125	1354
	DASD GB, DG:	5	433	1333	322	3832
	Access Density, AD	6	2.21	1.17	1.21	0.83
Activity	Total, S	7	957	1555	388	4611
Rate	Highest, Smx	8	16.63	42.64	14.13	145.81
(I/Os/sec]	Average, Sav	9	6.17	3.12	3.11	3.41
	Highest / Average	10	2.69	13.65	4.55	42.81
Times	CONN Time	11	2.26	1.97	1.27	2.60
[ms]	DISC Time	12	1.10	4.75	0.96	2.46
	Service Time, ST	13	3.36	6.72	2.23	5.06
	PEND Time	14	1.12	0.17	0.31	0.20
	IOSQ Time	15	0.25	1.62	0.07	1.25
	Response Time, RT	16	4.73	8.56	2.61	6.50
	RT to ST Ratio	17	1.41	1.27	1.17	1.28
Intensities	I/O Intensity I	18	4523	13308	1014	29969
[ms/s]	ST Intensity SI	19	3215	10452	867	23346
	Path Intensity, PI	20	2158	3070	494	11993
DASD	Avg. ST Intens: SIav	21	20.74	20.99	6.94	5.06
Skew	Max. ST Intens:SImx	22	147.96	326.17	219.26	356.97
	Max / Avg (Skew)	23	7.13	15.54	31.61	70.51
	Typical Avg. Value	24	8.85	15.37	8.37	23.87
	Typical Low Value	25	5.95	9.35	5.69	13.41
	Typical Hi Value	26	13.16	25.27	12.31	42.51
I/O Content	MIPS available, M	27		1108	331	3023
	MIPS used, MU	28		630	154	931
	I/O Content, R	29		2.47	2.52	4.94

Figure 9: The system data for the first four installations

Installation number			9	10	11	12
		Line				
Overall	Number of SSIDs	1	11			6
Category	Hit Ratio, HR	2	0.88			0.94
	Number of LCUs, L	3	4	6	9	3
	No of DASDs, N	4	339	105.	389	290
	DASD GB, DG	5	624	261	812	821
	Access Density, AD	6	0.64	0.58	1.96	1.91
Activity	Total, S	7	976	152	1590	1569
Rate	Highest, Smx	8	59.58			30.33
[IO's/sec]	Average, Sav	9	2.88	1.45	4.09	5.41
	Highest / Average	10	20.70			5.61
Times	CONN Time	11	3.14	3.56	1.46	1.63
[ms]	DISC Time	12	4.95	1.97	4.12	1.35
	Service Time, ST	13	8.09	5.52	5.58	2.98
	PEND Time	14	0.96	1.88	0.87	0.63
	IOSQ Time	15	2.14	1.14	1.62	0.48
	Response Time, RT	16	11.26	8.54	8.07	4.09
	RT to ST Ratio	17	1.39	1.55	1.45	1.37
Intensities	I/O Intensity, I	18	10986	1301	12831	6410
[ms/s]	ST Intensity, SI	19	7897	842	8873	4673
	Path Intensity, PI	20	3067	542	2326	2554
DASD	Avg ST Intensity	21	23.29	8.02	22.81	16.11
Skew	Max. ST Intensity	22	458.74	202.72	559.97	172.86
	Max / Avg (Skew)	23	19.69	25.28	24.55	10.73
	Typical Avg. Value	24	12.67	7.75	13.79	12.12
	Typical Low Value	25	7.99	5.34	8.56	7.70
	Typical High Value	26	20.11	11.25	22.22	19.07
I/O Content	MIPS available, M	27	362			331
	MIPS used, MU	28	314			331
	I/O Content, R	29	3.11			4.74

Figure 10: The system data for the last four installations

Installation	**1**	**2**	**3**	**4**
Volume serial	SMS263	DBP527	DB2PG3	ZINF02
I/O Intensity	277.85	3273.79	1898.66	484.99
ST Intensity	71.46	723.40	577.51	313.67
Path Intensity	10.92	73.92	166.13	30.15
Activity Rate	9.93	52.80	79.11	2.77
Response Time	27.99	62.00	24.00	175.09
Service Time	7.20	13.70	7.30	113.24
IOSQ Time	2.00	48.00	15.00	47.82
PEND Time	0.40	0.80	2.10	14.46
DISC Time	6.10	12.30	5.20	102.35
CONN Time	1.10	1.40	2.10	10.89
RT/ST	3.89	4.53	3.29	1.55
%I/O Intensity	3.76%	9.70%	4.91%	2.64%
Read H/R	1.00	0.30	0.89	0.00
% Read	100.00	99.10	100.00	0.00
Volume serial	SMS228	IMSS45	DB2PF8	SYSTS1
I/O Intensity	175.02	1754.43	1396.91	1537.50
ST Intensity	60.85	60.85	523.84	25.90
Path Intensity	106.68	15.84	186.25	24.76
Activity Rate	8.34	8.34	116.41	29.54
Response Time	21.00	32.00	12.00	52.05
Service Time	7.3	7.3	4.50	0.88
IOSQ Time	0.00	26.00	6.00	49.05
PEND Time	0.40	4.40	1.80	1.81
DISC Time	8.00	0.10	2.90	0.04
CONN Time	12.80	1.90	1.60	0.84
RT/ST	2.88	4.38	2.67	59.37
%I/O Intensity	2.37%	5.20%	3.61%	8.38%
Read H/R	1.00	0.99	0.94	1.00
% Read	60.80	99.20	100.00	99.60
Volume serial	SMS191	DBP458	SYSS11	WSF210
I/O Intensity	131.05	1560.21	434.18	961.96
ST Intensity	129.96	382.69	7.89	388.44
Path Intensity	54.61	147.19	7.89	116.23
Activity Rate	10.92	29.44	13.16	36.97
Response Time	12.00	53.00	33.00	26.02
Service Time	11.90	13.00	0.60	10.51
IOSQ Time	0.00	39.00	5.00	10.86
PEND Time	0.30	0.90	26.70	5.00
DISC Time	6.90	8.00	0.00	7.36
CONN Time	5.00	5.00	0.60	3.14
RT/ST	1.01	4.08	55.00	2.48
%I/O Intensity	1.77%	4.62%	1.12%	5.24%
Read H/R	0.53	0.91	1.00	0.94
% Read	55.60	98.30	96.30	84.60
Top3%IO Int	7.90%	19.52%	9.65%	16.26%

Figure 11: The three most tunable logical volumes (first four)

	5	6	7	8
Volume serial	SYSS08	ARGP2E	TYGPG2	SY1300
I/O Intensity	222.55	509.34	53.26	542.75
ST Intensity	158.83	289.73	51.19	108.55
Path Intensity	132.25	232.29	8.58	74.01
Activity Rate	19.92	50.93	2.96	49.34
Response Time	11.17	10.00	17.50	11.00
Service Time	7.97	5.69	17.30	2.20
IOSQ Time	1.83	2.67	0.00	9.00
PEND Time	1.26	1.59	0.20	0.20
DISC Time	1.33	1.13	14.40	0.70
CONN Time	6.64	4.56	2.90	1.50
RT/ST	1.40	1.76	1.01	5.00
%I/O Intensity	4.92%	3.83%	5.25%	1.81%
Read H/R	0.97	0.96		0.00
% Read	96.30	96.90		0.00

Volume serial	DBAP05	ARGP9V	TYGPG3	SY1301
I/O Intensity	123.43	471.59	50.07	508.71
ST Intensity	106.60	326.18	49.77	117.00
Path Intensity	16.83	39.30	8.54	76.31
Activity Rate	11.22	39.30	2.95	50.87
Response Time	11.00	12.00	17.10	10.00
Service Time	9.50	8.30	16.90	2.30
IOSQ Time	1.00	3.00	0.00	7.00
PEND Time	0.30	0.30	0.20	0.20
DISC Time	8.00	7.30	14.00	0.80
CONN Time	1.50	1.00	2.90	1.50
RT/ST	1.16	1.45	1.01	4.35
%I/O Intensity	2.73%	3.54%	4.94%	1.70%
Read H/R	0.00	0.49		0.00
% Read	0.00	94.90		0.00

Volume serial	DBAP30	ARGPAQ		DB2025
I/O Intensity	119.31	431.60		381.38
ST Intensity	92.70	221.19		356.97
Path Intensity	25.70	13.49		295.95
Activity Rate	9.18	26.98		15.26
Response Time	13.00	16.00		25.00
Service Time	10.10	8.20		23.40
IOSQ Time	2.00	7.00		2.00
PEND Time	0.60	0.20		0.20
DISC Time	7.30	7.70		4.00
CONN Time	2.80	0.50		19.40
RT/ST	1.29	1.95		1.07
%I/O Intensity	2.64%	3.24%		1.27%
Read H/R	0.00	0.63		0.91
% Read	0.00	92.30		100.00
Top3%IO Int	10.29%	10.61%	10.19%	4.78%

Figure 12: The three most tunable logical volumes (second four)

	9	10	11	12
Volume serial	CME244	M90YBF	BZKR04	DSN002
I/O Intensity	673.12	273.90	728.00	363.91
ST Intensity	210.72	202.69	560.00	172.86
Path Intensity	51.22	156.12	84.00	100.08
Activity Rate	14.63	27.39	56.00	30.33
Response Time	46.00	10.00	13.00	12.00
Service Time	14.40	7.40	10.00	5.70
IOSQ Time	31.00	1.00	2.00	6.00
PEND Time	0.50	1.60	0.90	0.30
DISC Time	10.90	1.70	8.50	2.40
CONN Time	3.50	5.70	1.50	3.30
RT/ST	3.19	1.35	1.30	2.11
%I/O Intensity	6.13%	21.05%	5.67%	0.06
Read H/R	0.64		0.39	0.90
% Read	69.40		91.50	64.50
Volume serial	CME105	M9BE33	BZKR47	DSN004
I/O Intensity	655.35	156.35	537.60	311.47
ST Intensity	458.74	50.63	295.68	163.52
Path Intensity	125.11	67.20	73.92	46.72
Activity Rate	59.58	14.89	67.20	25.96
Response Time	11.00	10.50	8.00	12.00
Service Time	7.70	3.40	4.40	6.30
IOSQ Time	2.00	5.00	3.00	5.00
PEND Time	0.60	2.10	0.90	0.30
DISC Time	5.60	1.50	3.30	4.50
CONN Time	2.10	1.90	1.10	1.80
RT/ST	1.43	3.09	1.82	1.90
%I/O Intensity	5.97%	12.01%	4.19%	0.05
Read H/R	0.94		0.86	0.83
% Read	3.70		94.10	98.40
Volume serial	CME109	M9BE2F	BZKR47	PROD10
I/O Intensity	451.58	97.47	283.40	166.35
ST Intensity	412.07	44.60	163.50	152.38
Path Intensity	95.96	31.39	30.52	40.59
Activity Rate	56.45	16.52	21.80	6.65
Response Time	8.00	5.90	13.00	25.00
Service Time	7.30	2.70	7.50	22.90
IOSQ Time	0.00	2.00	5.00	2.00
PEND Time	0.60	1.20	1.00	0.30
DISC Time	5.60	0.80	6.10	16.80
CONN Time	1.70	1.90	1.40	6.10
RT/ST	1.10	2.19	1.73	1.09
%I/O Intensity	4.11%	7.49%	2.21%	0.03
Read H/R	0.00		0.67	0.75
% Read	0.00		91.60	99.60
Top3%IO Int	16.20%	40.55%	12.07%	13.13

Figure 13: The three most tunable logical volumes (last four)

CALL FOR PERFORMANCE DATA

This article has examined performance data from 12 MVS installations to draw its conclusions. In order to draw more meaningful conclusions the number of installations should be much higher.

It would be desirable if every reader could send his own performance data so that the article may be updated. If interested, please send your data to the author, Thomas Beretvas, at e-mail address: beretvas@iname.com. Your contribution will remain anonymous. If requested, a subsequent report could be sent to you.

FORMAT OF DATA TO BE SENT

The format should be RMF postprocessed data (133 characters in length) in ASCII. The following reports have to be included: CPU, channel, IOQ, DASD, and cache.

Not to be included: tape, workload activity, unit record. The interval to be covered should be perhaps fifteen minutes to one hour peak time. If multiple systems share DASD, data from all sharing systems with significant load should be included. In addition, please include in a separate file the following information: your name, telephone number, and e-mail address.

The following information is *not* in RMF (or not correctly) so please include this if you can:

- CPU types used (eg IBM 9672-RD6) with your understanding of their MIPS (eg, 408).
- Control units used (eg EMC 5430, 8 paths, 2GB cache) with address if known.
- Logical volume sizes (if known and other than the default 3390-1 or 3390-3 as reported on RMF).

Thomas Beretvas
Beretvas Performance Consultants (USA)

Determining who is delaying the system

INTRODUCTION

The worst thing about a 'hanging system' is normally that it is really hard to get signed on to see what exactly is causing the problem. It would be useful if we could have a simple command we could enter from the console to display a summary of CPU and storage usage per job name. Well, although MVS does not support such a command directly, there is always the possibility of writing one ourselves. In some of the previous editions of *MVS Update* we have already seen examples of how to write an MVS command exit. Refer to the article *Selectively blocking commands* in *MVS Update*, Issue 157, October 1999, for a discussion on how to do this. This exit allows us to intercept commands before MVS sees them and, in the event that we wish to add support for a new command to the system, we can process the command ourselves and then instruct MVS to ignore it. This will suppress MVS from giving an 'Invalid command' message.

Hidden in a somewhat obscure manual is a really great interface you can use to obtain RMF data inside your Assembler program. The manual is the *RMF Programmer's Guide* and it describes RMF API called ERBSMFI. This interface allows users to obtain RMF data directly from storage, rather than through SMF. It supports SMF record type 79 and the Monitor II header information for system CPU utilization ,and the system demand paging rate. When the call is made, a single record that contains all the data is returned and the record is not split up into 32-byte segments.

So how can we put this to good use? Well, amongst other subtypes of record type 79 that are supported is also support for subtype 2. This allows us to retrieve all performance statistics per job name. Combine this with the ability to define and support a new command and we have a command with the format DISPLAY SYSUSE (or whatever you may want to call your new command), to show us who is using the most system resources.

The information in the SMF 79/2 record is of a point-in-time nature. It contains for each job/ STC, amongst others, the amount of CPU used at that point, the number of storage frames in use, and also the total number of SRM service units consumed. So, what we do is this: accept the command from MVS, call the RMF API to get point-in-time statistics, wait for a while (maybe 0.5 seconds or so), call the RMF API again and subtract the values. Let's say a particular job has used 1 CPU second when we made the first call and 1.2 CPU seconds when we made the second call. Subtracting the two values gives a total CPU usage of 0.2 seconds for the interval. Say we have four processors on-line. In the 0.5 seconds that we waited, the total CPU capacity was 0.5 seconds times 4 CPUs, thus 2 CPU seconds. As the job has used 0.2 CPU seconds, it has used 10% of the available CPU for the period. Work this out for every job, sort them in descending order, and print the highest one and we have caught the culprit red-handed. (You can have a look at the source code for SDSF; it uses the same method to calculate CPU percentages for jobs). So, to recap:

- Accept the DISPLAY SYSUSE command through the command exit.
- Call ERBSMFI to get the figures for all jobs.
- 'Sleep' for 0.5 seconds.
- Call ERBSMFI again to get the updated figures.
- Determine how much CPU/resources each job has used by subtracting the two readings.
- Find the highest user.
- Display the job name of the highest user.

The same principle can be applied for SRM service usage as well as central and auxiliary storage usages: take two snapshots, subtract the first from the second, sort, and then pick the highest one. Job A may show up as the highest CPU user, yet job B may be the highest user of fixed page frames below the 16MB line. The following is an example of such a program and it is currently used in production. The program displays the following on the operator's console:

- Highest CPU user
- Highest user of fixed frames below the 16MB line
- Highest user of virtual storage frames
- Highest user of total SRM service.

Note that the program has one ‘feature’, which has to do with a little laziness on the side of the developer. If the list of active job names changes in the 0.5 seconds that the program ‘sleeps’ between the two calls, the program gives up and asks to be invoked again. This is because the tables containing the job names will be slightly different, which complicates the subtractions. This can easily be fixed: if a job has ended in this 0.5 seconds or if a new one has entered, simply delete it from the equation. This is not a major problem: it is only used by systems programmers, who as we all know, have a high-level of tolerance for certain questionable ‘features’. Fixing it should take no longer than an hour or so and will give you a good insight into the program. Just a reminder: the program was specifically written to be called from the command exit and expects to receive the CMDX (command buffer) as an input buffer. If you plan to call this from REXX or any other environment, you will obviously have to make some minor changes.

A final hint: if the most resource-intensive job shows up as being your – own TSO user-id from which you issued the new command (as sometimes happens), your system is probably not hanging because one job is too resource or CPU intensive and you may want to investigate other possibilities for example like deadlocks, etc.

This routine lists the following highest job/user-ids/stc:

- Highest CPU user
- Highest user of fixed frames below the 16MB line
- Highest user of virtual STORAGE frames
- Highest user of total SRM service
- It also lists the system’s CPU usage and demand page rate.

(CPU usage is a percentage of usage of all logical processors assigned to this system, not a percentage of the CPU time actually received by this LPAR.) It is called from the command exit and receives a pointer to the CMDX as parameter.

```
SYSUSERS CSECT
```



```
SYSUSERS AMODE 31
SYSUSERS RMODE ANY
         BAKR  R14,Ø                .Save Caller's Status
         BALR  R12,Ø
         USING *,12
*********************************************************************
*        Main driver routine
*********************************************************************
Load     L     R4,Ø(R1)             .Pointer to routine parms (none)
         L     R5,4(R1)             .Address of CMDX
         USING CMDX,R5              .Addressability to CMDX
STORAGE  L     R3,STORSIZE          .Size of storage to get and clear
         STORAGE OBTAIN,LENGTH=(3),LOC=ANY,SP=229
         LR    R2,R1                .Point to getmained area
         L     R3,STORSIZE          .Length of storage to clear
         XR    R9,R9                .Fill with binary zeroes
         MVCL  R2,R8                .Propagate binary zeroes
         USING GETMAREA,R1
         ST    R13,SAVEAREA+4       .Backchain
         DROP  R1
         LR    R13,R1
         USING GETMAREA,R13         .Addressability to getmained area
         STORAGE OBTAIN,LENGTH=BUFSIZE,LOC=BELOW,SP=229
         LR    R8,R1
         USING BUFFER,R8
         MVC   CART,CMDXCART        .Pick up the CART
         MVC   FROMCNID,CMDXC4ID    .Where the command originated (Id)
         MVC   FROMSYS,CMDXISYN     .Where the command originated (Sys)
         BAS   R14,DORMF            .Obtain information from RMF
         LTR   R15,R15              .Did we get it?
         BNZ   CLEANUP              .No, get out
         BAS   R14,SEEKHIGH         .Determine all the high users
         BAS   R14,WTOHIGH          .Display all the high users
CLEANUP  STORAGE RELEASE,LENGTH=BUFSIZE,ADDR=(8),SP=229
         L     R4,RCODE             .Pick up retrun code
         LR    R2,R13               .Pointer to storage area
         L     R3,STORSIZE          .Size of storage to free
         STORAGE RELEASE,LENGTH=(3),ADDR=(2),SP=229
         LR    R15,R4               .Reload return code
         PR                         .Back to our caller
**********************************************************************
*        This routine calls RMF
**********************************************************************
DORMF    BAKR  R14,Ø
         MVC   EYECATCH,=C'EYE:'
         LA    R1,REQTYPE
         ST    R1,PARMLIST          .Type of request
         LA    R1,RECTYPE
         ST    R1,PARMLIST+4        .SMF record type
         LA    R1,SUBTYPE
```

```
         ST    R1,PARMLIST+8       .SMF record SUBTYPE
         LA    R1,BUFFER
         ST    R1,PARMLIST+12      .Address of buffer
         LA    R1,BUFLENG
         ST    R1,PARMLIST+16      .Length of buffer
         LA    R1,CPUUTIL
         ST    R1,PARMLIST+2Ø      .Address to contain %CPU busy
         LA    R1,DMNDPAGE
         ST    R1,PARMLIST+24      .Demand page rate
         OI    FIRSTIME,X'Ø1'      .This is the first call to RMF
CALLRMF  LA    R1,PARMLIST
         STCK  LASTCLOK            .Store current time
         LINK  EP=ERBSMFI          .Call RMF
         ST    R15,RCODE
         LTR   R15,R15             .OK from RMF?
         BNZ   DORMFX              .No, get out immediately
         LA    R2,BUFFER           .Where RMF put the data
         USING SMF79HDR,R2         .Addressability to type 79/2
         LH    R3,SMF79ASL         .Length of each data section
         LH    R4,SMF79ASN         .Number of data sections
         STH   R4,NUMASID          .Store this value
         CH    R4,=AL2(NUMENTRY)   .More than we have space for?
         BNH   NUMOK               .No
         WTO   'SYSUSERS(W): -Only first 999 ASCBs scanned',           X
               ROUTCDE=13,CONSID=FROMCNID,SYSNAME=FROMSYS,CART=CART
         LH    R4,=AL2(NUMENTRY)   .Set to what we have space for
         STH   R4,NUMASID          .Update number of ASIDs scanned
NUMOK    EQU   *
         A     R2,SMF79ASS         .Add offset to data section
         DROP  R2
         USING R792ELEM,R2         .Addressability to data section
         LA    R5,TABLE
         USING TABDSECT,R5         .Addressability to TABLE
LOOP     TM    FIRSTIME,X'Ø1'      .First iteration?
         BNO   SUBTRACT            .No, SUBTRACT the value
         MVC   JOBNAME,R792JBN     .Put jobname into the TABLE
         MVC   FIRSTCPU,R792EJST   .Put TCB+SRB time into TABLE
         B     BUMPUP
SUBTRACT CLC   JOBNAME,R792JBN     .Still matching?
         BNE   CANTDO              .No
         L     R1,FIRSTCPU         .Load previous service
         ICM   RØ,15,R792EJST
         ST    RØ,LASTCPU          .Keep for debugging purposes
         SLR   RØ,R1               .SUBTRACT current value
         ST    RØ,DIFFERNC         .Store back
         MVC   JOBABS,R792SVAR     .Get the job's absorbtion rate
         ICM   RØ,15,R792PNV       .Non-VIO pages in use
         ICM   R1,15,R792PVIO      .VIO pages
         AR    RØ,R1               .Add it
         ICM   R1,15,R792PHSP      .Hiperspace pages
```

```
         AR     RØ,R1                .Add it
         STCM   RØ,15,JOBRGN         .The total picture
         MVC    JOBF16,R792FXBL      .Get the job's fixed frames below 16
         B      BUMPUP
CANTDO   WTO    'SYSUSERS(W): -Cannot calculate individual CPU%, re-issuX
               e comand',ROUTCDE=13,CONSID=FROMCNID,SYSNAME=FROMSYS,   X
               CART=CART
         LA     R15,8                .Set return code to 8
         ST     R15,RCODE
         B      DORMFX               .Get out
BUMPUP   LA     R5,TABSIZE(R5)       .Bump up TABLE pointer
         AR     R2,R3
         BCT    R4,LOOP              .Do for each jobname in the TABLE
CHKFIRST TM     FIRSTIME,X'Ø1'       .Is this the first iteration?
         BNO    DORMFX               .No, get out
         NI     FIRSTIME,X'ØØ'       .Turn flag off
         LA      R6,WAITPARM         .Required to work out user's %CPU
         STIMER WAIT,DINTVL=(6)      .Sleep for a while to get an offset
         MVC    OLDCLOK,LASTCLOK     .Preserve the previous store clock
         B      CALLRMF              .Call RMF for the second time
DORMFX   PR                          .Back to our caller
***********************************************************************
*        This routine determines the highest users of resources
***********************************************************************
SEEKHIGH BAKR   R14,Ø                .Find ASIDs with high usage
         LH     R4,NUMASID           .Number of jobnames in TABLE
         LA     R5,TABLE             .Start of jobname TABLE
         XR     R9,R9                .Set highest CPU to Ø to start off
         XR     R3,R3                .Set highest fixed frames to Ø
         XR     RØ,RØ                .Set highest region to Ø
         XR     R6,R6                .Set highest service ration to Ø
COMPCPU  C      R9,DIFFERNC          .Higher than what we got?
         BNL    COMPFFRM             .No
         ST     R5,HIGHCPU@          .Adr of job with highest CPU usge
         L      R9,DIFFERNC          .This one now becomes highest
COMPFFRM C      R3,JOBF16
         BNL    COMPRGN
         ST     R5,HIGHF16@          .Adr of job with highest <16 fixed
         L      R3,JOBF16
COMPRGN  C      RØ,JOBRGN
         BNL    COMPABS
         ST     R5,HIGHRGN@          .Adr of job with highest reg size
         L      RØ,JOBRGN
COMPABS  CLM    R6,15,JOBABS
         BNL    SCANNEXT
         ST     R5,HIGHABS@          .Adr of job with highest absorption
         L      R6,JOBABS
SCANNEXT LA     R5,TABSIZE(R5)
         BCT    R4,COMPCPU           .Do for each entry in the TABLE
SEEKHIGX PR                          .Return to our caller
***********************************************************************
```



```
*        This routine prints out the highest users.
***********************************************************************
WTOHIGH  BAKR  R14,Ø
CALCTIME LM    RØ,R1,LASTCLOK      .Pick latest STCK value
         LM    R2,R3,OLDCLOK       .Pick up previous STCK value
         SLR   RØ,R2               .SUBTRACT high-order word (unsigned)
         SLR   R1,R3               .SUBTRACT low-order word (unsigned)
         BM    CARRY               .Negative, carry 1 over
         B     MAKEMILI            .Store the values back
CARRY    BCTR  RØ,Ø                .SUBTRACT 1 for carry-over
MAKEMILI SRDL  RØ,12               .Make micro seconds, result in R1
         LR    R2,R1               .Preserve R1
         BAS   R14,GET#PROC        .Find out how many online processors
         LR    R1,R2               .Reload R1
         MH    R1,NUMPROCS         .Number of processors
         XR    R2,R2
         L     R5,HIGHCPU@         .Address of highest user in TABLE
         L     R3,DIFFERNC
         M     R2,=F'1ØØØØØØØ'     .Make micro seconds x 1ØØ% + 2 dgts
         DR    R2,R1               .Calc %CPU for the highest user
         AH    R3,=H'5'            .Rounding factor
         CVD   R3,DOUBLE
         MVC   WORK8,PATTERN1
         ED    WORK8(8),DOUBLE+4   .Make printable
         MVC   MAXWTOA(MAXCPUL),MAXCPU
         MVC   MAXWTOA+38(2),WORK8+4 CPU usage
         MVC   MAXWTOA+41(1),WORK8+6 CPU usage, decimal value
         MVC   MAXWTOA+47(8),Ø(R5) Move the jobname into the WTO
         LA    R1,MAXWTOA
         WTO   MF=(E,(1)),CONSID=FROMCNID,SYSNAME=FROMSYS,CART=CART
         L     R5,HIGHF16@         .Addr of job with highest fixed
         L     R1,JOBF16           .Get the number of frames
         CVD   R1,DOUBLE
         MVC   WORK8,PATTERN2
         ED    WORK8(8),DOUBLE+4   .Make printable
         MVC   MAXWTOA(MAXF16L),MAXF16
         MVC   MAXWTOA+47(6),WORK8+2
         MVC   MAXWTOA+65(8),JOBNAME
         LA    R1,MAXWTOA
         WTO   MF=(E,(1)),CONSID=FROMCNID,SYSNAME=FROMSYS,CART=CART
         L     R5,HIGHRGN@         .Address of job with max virt stor
         MVC   MAXWTOA(MAXFRMSL),MAXFRMS
         L     R1,JOBRGN           .Pick up the number of frames
         CVD   R1,DOUBLE
         MVC   WORK8,PATTERN2
         ED    WORK8(8),DOUBLE+4   .Make printable
         MVC   MAXWTOA+41(7),WORK8+1
         MVC   MAXWTOA+6Ø(8),JOBNAME
         LA    R1,MAXWTOA
         WTO   MF=(E,(1)),CONSID=FROMCNID,SYSNAME=FROMSYS,CART=CART
         L     R5,HIGHABS@         .Address of job with max absorbtion
```



```
         MVC   MAXWTOA(MAXABSL),MAXABS
         MVC   MAXWTOA+55(8),JOBNAME
         LA    R1,MAXWTOA
         WTO   MF=(E,(1)),CONSID=FROMCNID,SYSNAME=FROMSYS,CART=CART
SHOWCPU  L     R5,CPUUTIL          .System wide %CPU
         CVD   R5,DOUBLE
         MVC   WORK8,PATTERN2
         ED    WORK8(8),DOUBLE+4   .Make printable
         MVC   CPUPERC,WORK8+5
         L     R5,DMNDPAGE         .System's demand page rate
         CVD   R5,DOUBLE
         UNPK  DOUBLE(5),DOUBLE+5(3)
         OI    DOUBLE+4,X'FØ'      .Make printable
         MVC   DEMANDPG,DOUBLE
         MVC   WTOAREA(WTOLENG),SYSWIDE
         MVC   WTOAREA+35(3),CPUPERC
         MVC   WTOAREA+58(5),DEMANDPG
         LA    R1,WTOAREA
         WTO   MF=(E,(1)),CONSID=FROMCNID,SYSNAME=FROMSYS,CART=CART
WTOHIGHX PR                        .Return to our caller
**********************************************************************
*        Determine the number of online processors
**********************************************************************
GET#PROC BAKR  R14,Ø
         L     R5,16               .CVT address
         DROP  R5
         USING CVT,R5              .CVT addressing
         L     R8,CVTLCCAT         .Address of LCCAVT
         LA    R2,6Ø(R8)           .Last entry address
         LA    R3,16               .Maximum possible processors
LCCALOOP DS    ØH
         ICM   R1,15,Ø(R2)         .Get LCCA address
         BNZ   LCCAOUT1            .Does not exist
         SH    R2,=H'4'            .Go to prior entry
         BCT   R3,LCCALOOP         .Do for each entry
         ABEND ØØØ3
LCCAOUT1 DS    ØH
         STH   R3,NUMPROCS         .Maximum number of CPUs
         L     R8,CVTPCCAT         .PCCAVT address
         LA    R2,6Ø(R8)           .Last entry address
         LA    R3,16               .Maximum possible processors
PCCALOOP DS    ØH
         ICM   R1,15,Ø(R2)         .Is PCCA address specified?
         BNZ   PCCAOUT1            .No
         SH    R2,=H'4'            .Go to prior entry
         BCT   R3,PCCALOOP         .LOOP for all processors
         ABEND ØØØ4                .Serious problem
PCCAOUT1 DS    ØH
         CH    R3,NUMPROCS         .Higher?
         BNH   GET#PROX            .No
         STH   R3,NUMPROCS         .Update CPU max
```

```
GET#PROX PR
***********************************************************************
*        Constants follow
***********************************************************************
REQTYPE  DC    F'1'
RECTYPE  DC    F'79'
SUBTYPE  DC    F'2'
BUFLENG  DC    F'500000'
SYSWIDE  WTO   'SYSUSERS(I): -System CPU usage=xxx%, Demand page rate=xX
               xxxx pages/sec',ROUTCDE=13,CONSID=,SYSNAME=,CART=,MF=L
WTOLENG  EQU   *-SYSWIDE
         DS    0F
STORSIZE DC    AL4(GETMSIZE+(TABSIZE*NUMENTRY))
WAITPARM DC    C'00000150'          .Sleep for 1.5 second
MAXCPU   WTO   'SYSUSERS(I): -Highest CPU user at xx.x% is xxxxxxxx',  X
               ROUTCDE=13,CONSID=,SYSNAME=,CART=,MF=L
MAXCPUL  EQU   *-MAXCPU
MAXF16   WTO   'SYSUSERS(I): -Max #pages fixed below 16MB (xxxxxx frameX
               s) is xxxxxxxx',CONSID=,SYSNAME=,CART=,ROUTCDE=13,      X
               MF=L
MAXF16L  EQU   *-MAXF16
MAXFRMS  WTO   'SYSUSERS(I): -Largest user of VSTOR (xxxxxxx frames) isX
                xxxxxxxx',ROUTCDE=13,CONSID=,SYSNAME=,CART=,MF=L
MAXFRMSL EQU   *-MAXFRMS
MAXABS   WTO   'SYSUSERS(I): -Highest user of total SRM service is xxxxX
               xxxx',ROUTCDE=13,CONSID=,SYSNAME=,CART=,MF=L
MAXABSL  EQU   *-MAXABS
PATTERN1 DC    X'4020202020212020'
PATTERN2 DC    X'4020202020202021'
         LTORG
GETMAREA DSECT
SAVEAREA DS    18F
PARMLIST DS    7F
CPUPERC  DS    CL3
DEMANDPG DS    CL5
CART     DS    D                     .Command and response token
FROMCNID DS    F                     .Console the cmnd was issued from
FROMSYS  DS    D                     .System the cmnd was issued from
DOUBLE   DS    D                     .General workarea
RCODE    DS    F                     .Our return code
WTOAREA  DS    CL(WTOLENG)           .Area to contain WTO
CPUUTIL  DS    F                     .System's CPU utilisation
DMNDPAGE DS    F                     .System's demand page rate
WORK8    DS    D                     .General workarea
OLDCLOK  DS    D                     .Storeclock value first time
LASTCLOK DS    D                     .Storeclock value second time
FIRSTIME DS    C                     .Flag
HIGHCPU@ DS    F                     .Address of highest CPU user
HIGHRGN@ DS    F                     .Address of largest region user
HIGHABS@ DS    F                     .Address of highest SRM rate user
HIGHF16@ DS    F                     .Address of highest fixed fram
NUMASID  DS    H                     .Number of ASIDs on ASCB chain
```



```
NUMPROCS DS    H                     .Number of online processors
MAXWTOA  DS    CL(MAXF16L)           .Workarea for highest CPU user WTO
EYECATCH DS    CL4
GETMSIZE EQU   *-GETMAREA
*
         DS    ØD
TABLE    DS    CL(TABSIZE*NUMENTRY) .TABLE has space for 999 entries
NUMENTRY EQU   999                   .Support up to 999 address spaces
*
BUFFER   DSECT
BUFSIZE  EQU   5ØØØØØ

TABDSECT DSECT
JOBNAME  DS    D
FIRSTCPU DS    F                     .Service at start of interval
LASTCPU  DS    F                     .Service at end of interval
DIFFERNC DS    F                     .Difference between the 2
JOBABS   DS    F                     .Job's SRM absorbtion rate
JOBF16   DS    F                     .Job's fixed frames below 16 MB line
JOBRGN   DS    F                     .Total # of pages in use by job
TABSIZE  EQU   *-TABDSECT
*
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*
         IEZVX1Ø1                    .Command exit parameter mapping
*
         ERBSMF79                    .SMF type 79/2 mapping
*
         IHALCCA                     .LCCA mapping
*
         CVT DSECT=YES
         END
```

Tommy Afbeen
Senior Consultant
Brandenberg Consulting (South Africa)

Conversion from AutoMedia to DFSMSrmm

INTRODUCTION

Our management decided to migrate from AutoMedia(ZARA) to DFSMSrmm. Our task was to convert the existing tape library, which contained 7,000 tapes.

The first part of the process was to install and customize DFSMSrmm. The principal conclusion from this process was that ZARA and DFSMSrmm can work in parallel.

In the second stage we extracted data from the ZARA files and prepared ADDVOLUME and ADDVRS control statements for DFSMSrmm. Because we did not find procedures for that process in the IBM literature or anywhere in the Web, we wrote them ourselves.

EXTRACTING INFORMATION FROM ZARA

The first job shown below lists all the ZARA information and calls the program ZARARMM,which generates ADDVOLUME statements, and REXX procedure ZARARMM, to generate ADDVRS statements.

```
//*************************************************************
//***  DELETE ALL WORK FILES
//*************************************************************
//DEL EXEC PGM=IDCAMS,REGION=0M
//SYSPRINT  DD SYSOUT=X
//SYSIN     DD *
    DELETE userid.#ZAS.LIST
    DELETE userid.#ZR.LIST
    DELETE userid.#ZDMG.LIST
    DELETE userid.#ZV.LIST
    SET MAXCC=0
/*
//*************************************************************
//***  LIST  ALL ACTIV AND SCRATCH ZARA TAPES
//*************************************************************
//LISTACT  EXEC ZARAUTL
//SYSUDUMP DD *
//ZARAUTL.ZARAWK    DD DSN=userid.#ZAS.LIST,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,
//         SPACE=(TRK,(5,5),RLSE)
//SYSIN    DD *
  REPORT ALLDSNPRT VSTAT=(ACTV,SCR)
         SORTSEQ=(VSTAT,FRSTVOL,VSEQ,FSEQ) OUTFILE $$
```

```
/*
//***********************************************************
//***  LIST  ALL DAMAGED ZARA TAPES
//***********************************************************
//LISTDMG  EXEC ZARAUTL
//SYSUDUMP DD *
//ZARAUTL.ZARAWK    DD DSN=userid.#ZDMG.LIST,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,
//         SPACE=(TRK,(5,5),RLSE)
//SYSIN    DD *
  REPORT ALLDSNPRT VSTAT=DMGE
         SORTSEQ=(VSTAT,FRSTVOL,VSEQ,FSEQ) OUTFILE $$
/*
//***********************************************************
//***  LIST  ALL UNUSED TAPES
//***********************************************************
//LISTRNGI  EXEC ZARAUTL
//SYSUDUMP DD *
//ZARAUTL.ZARAWK    DD DSN=userid.#ZR.LIST,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,
//         SPACE=(TRK,(5,5),RLSE)
//SYSIN    DD *
  REPORT ALLDSNPRT VSTAT=RNGI OUTFILE $$
/*
//***********************************************************
//***  LIST  ALL FILTERS FOR KEEP TAPES
//***********************************************************
//LISTAUTO  EXEC ZARAUTL
//SYSUDUMP DD *
//ZARAUTL.SYSPRINT   DD DSN=userid.#ZV.LIST,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,
//         SPACE=(TRK,(5,5),RLSE)
//SYSIN    DD *
  LIST NOSORT AUTO ALL OUTFILE $$
/*
//***********************************************************
//***  SORT RECORDS IN FILE BY (VSTAT,FRSTVOL,VSEQ,FSEQ)
//***********************************************************
//SORTA     EXEC PGM=ICEMAN
//SYSPRINT  DD SYSOUT=X
//SYSOUT    DD SYSOUT=X
//SORTIN    DD DSN=userid.#ZAS.LIST,DISP=SHR
//SORTOUT   DD DSN=userid.#ZAS.LIST,DISP=SHR
//SYSIN     DD *
 RECORD TYPE=V
 SORT FIELDS=(106,1,A,13,6,A,11,2,A,168,2,A),FORMAT=CH,WORK=1
 END
/*
//***********************************************************
//***  SECOND SEGMENT
//***  GENERATE STATEMENTS FOR DFSMSRMM
//***********************************************************
//***  DELETE ALL WORK FILES
```

```
//**************************************************************
//DEL      EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
   DELETE  userid.#RMMDEF.LIST
   DELETE  userid.#RMMDEFV.LIST
   SET MAXCC=Ø
/*
//********************************************************************
//***  PROGRAM ZARARMM GENERATE ADDVOLUME DFSMSRMM STATEMENTS
//***  FOR ACTIVE AND SCRATCH VOLUMES
//*********************************************************************
//ZARAA    EXEC PGM=ZARARMM,REGION=1M
//STEPLIB  DD DSN=userid.USER.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=X
//IN      DD DSN=userid.#ZAS.LIST,DISP=SHR
//        DD DSN=userid.#ZR.LIST,DISP=SHR
//        DD DSN=userid.#ZDMG.LIST,DISP=SHR   THIS FILE CAN BE EXCLUDED
//OUT     DD DSN=userid.#RMMDEF.LIST,DISP=(NEW,CATLG,DELETE),
//        UNIT=SYSDA,DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=Ø),
//        SPACE=(TRK,(5Ø,3Ø),RLSE)
/*
//********************************************************************
//***  REXX  PROCEDURE  GENERATE ADDVRS DFSMSRMM STATEMENTS
//*********************************************************************
//ZARAV     EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø
//SYSPROC   DD DSN=userid.USER.CLIST,DISP=SHR
//SYSTERM   DD SYSOUT=X
//SYSPRINT  DD SYSOUT=X
//SYSTSPRT  DD SYSOUT=X
//IN        DD DSN=userid.#ZV.LIST,DISP=SHR
//OUT       DD DSN=userid.#RMMDEFV.LIST,DISP=(NEW,CATLG,DELETE),
//          UNIT=SYSDA,DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=Ø),
//          SPACE=(TRK,(5,3),RLSE)
//SYSTSIN   DD *
  %ZARARMM
/*
```

INITIAL LOADING OF DFSMSRMM CONTROL INFORMATION

The next job defines basic information for DFSMSrmm. It executes all the statements generated in the previous job and updates the control parameters.

```
//DEF   EXEC PGM=IKJEFTØ1
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
    /*change xxxxxx to your hostid */
    RMM ADDOWNER xxxxxx -
    DEPARTMENT('company name')
```

```
    /*ADDRACK are based on information from userid.#ZV.LIST under */
    /*title Auto Range Definition Data *****/
    RMM ADDRACK  ØØØØØ1 COUNT(7ØØØ)
    RMM ADDRACK  99ØØØ1 COUNT(5ØØ)
    RMM ADDRACK  PSMØØ1 COUNT(999)

/*
//DEF       EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø
//SYSTERM   DD SYSOUT=X
//SYSPRINT  DD SYSOUT=X
//SYSTSPRT  DD SYSOUT=X
//SYSTSIN   DD DSN=userid.#RMMDEF.LIST,DISP=SHR
//          DD DSN=userid.#RMMDEFV.LIST,DISP=SHR
/*
//HSKP     EXEC PGM=EDGHSKP,
//     PARM='VRSEL'
//MESSAGE  DD DISP=SHR,DSN=RMM.MESSAGES
//REPORT   DD DISP=SHR,DSN=RMM.REPORT
```

This ends the process of extracting data.

PARALLEL RUNNING AND VALIDATION

DFSMSrmm works in RECORDING or WARNING mode, and ZARA is still our production tape manager. We submit jobs for scratch processing periodically on both products. The following job compares discrepancies between ZARA and DFSMSrmm scratch pools.

```
//**************************************************************
//***  DELETE ALL WORK FILES
//**************************************************************
//DEL EXEC PGM=IDCAMS,REGION=ØM
//SYSPRINT  DD SYSOUT=X
//SYSIN     DD *
    DELETE userid.#LSZ.LIST
    DELETE userid.#LSRMM.LIST
    DELETE userid.#LSALL.LIST
    DELETE userid.#LSNODUP.LIST
    SET MAXCC=Ø
/*
//**************************************************************
//***  LIST ALL SCRATCH ZARA TAPES
//**************************************************************
//LISTACT  EXEC ZARAUTL
//SYSUDUMP DD *
//ZARAUTL.SYSPRINT  DD DSN=userid.#LSZ.LIST,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,DCB=(RECFM=FB,LRECL=133,BLKSIZE=),
//         SPACE=(TRK,(5,5),RLSE)
```

```
//SYSIN    DD *
  LIST SCRATCH ALL $$
/*
//***************************************************************
//***  LIST ALL SCRATCH DFSMSRMM TAPES
//***************************************************************
// EXEC PGM=IKJEFTØ1
//SYSPRINT DD SYSOUT=X
//SYSTSPRT DD DSN=userid.#LSRMM.LIST,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,DCB=(RECFM=FB,LRECL=133,BLKSIZE=),
//         SPACE=(TRK,(5,5),RLSE)
//SYSTSIN DD *
    RMM SEARCHVOLUME OWNER(*) STATUS(SCRATCH) LIMIT(*)
/*
//***************************************************************
//***  PREPARATION FOR FINDING DIFFERENCES
//***  SERIAL NUMBERS MUST BE IN THE SAME COLUMNS (1-6)
//***************************************************************
//ICETOOL EXEC PGM=ICETOOL,REGION=1M
//TOOLMSG  DD SYSOUT=X
//DFSMSG   DD SYSOUT=X
//INZ      DD DSN=userid.#LSZ.LIST,DISP=SHR
//INRMM    DD DSN=userid.#LSRMM.LIST,DISP=SHR
//OUT      DD DSN=userid.#LSALL.LIST,DISP=(MOD,CATLG,DELETE),
//         UNIT=SYSDA,
//         SPACE=(TRK,(5,5),RLSE)
//TOOLIN   DD *
 COPY FROM(INZ)   TO(OUT) USING(SELZ)
 COPY FROM(INRMM) TO(OUT) USING(SELR)
/*
//SELZCNTL DD *
   INCLUDE COND=(118,7,CH,EQ,C'SCRATCH',OR,118,4,CH,EQ,C'RNGI')
   OUTREC FIELDS=(4,125)
/*
//SELRCNTL DD *
   INCLUDE COND=(6Ø,3,CH,EQ,C' S ')
   OUTREC FIELDS=(1,125)
/*
//*
//***************************************************************
//***  LIST OF DIFFERENCES
//***************************************************************
//ICETOOL EXEC PGM=ICETOOL,REGION=1M
//TOOLMSG  DD SYSOUT=X
//DFSMSG   DD SYSOUT=X
//IN       DD DSN=userid.#LSALL.LIST,DISP=SHR
//OUT      DD DSN=userid.#LSNODUP.LIST,DISP=(MOD,CATLG,DELETE),
//         UNIT=SYSDA,
//         SPACE=(TRK,(5,5),RLSE)
//TOOLIN   DD *
 SELECT FROM(IN) TO(OUT) ON(1,6,CH) NODUPS
/*
//
```


CONVERSION PROGRAM

```
ZARARMM: PROCEDURE OPTIONS(MAIN) REORDER;
/*****************************************************************/
/** PROGRAM GENERATE DFSMSRMM ADDVOLUME STATEMENT FROM ZARA RECORDS**/
/*****************************************************************/
/** CHANGE XXXXXX TO YOUR HOSTID                                  */
/*****************************************************************/
DCL IN   FILE RECORD  INPUT;
DCL OUT  FILE STREAM  OUTPUT;

DCL REC_IN  CHAR(STG(ZARA_RECORD)) VAR;
DCL 1 ZARA_RECORD UNALIGNED BASED(ADDR(REC_IN)),
 2 LEN_REC    BIN FIXED(15),
 2 VOLUME,
3 VOLSER      CHAR(6),       /* VOLUME SERIAL NUMBER OF TAPE         */
3 VOLSEQ      BIN FIXED(15), /* VOLUME SEQUENCE NUMBER               */
3 VOLFIRST    CHAR(6),       /* FIRST VOLUME IN CHAIN                */
3 VOLPREV     CHAR(6),       /* PREVIOUS VOLSER IN CHAIN             */
3 VOLNEXT     CHAR(6),       /* NEXT VOLSER IN CHAIN                 */
3 VOLVOFF     CHAR(6),       /* VOL OF DSN CONTROL OFFSITE MOVE      */
3 VOLOFSEQ    BIN FIXED(15), /* FILESEQ OF FILE CNTRLING OFFSIT MOVE */
3 VOLLOCK     CHAR(1),       /* Y - VOLUME IS IN USE OR ENQUED       */
3 VOLCBLVL    CHAR(1),       /* VOLDATA FORMAT VERSION               */
/*  VALID VALUE
  VOLVER1 =   1                   1.Ø-1.2
  VOLVER2 =   2                   1.3-?
  VOLVCUR =   2                   WHAT CURR LEVEL IS
*/
3 VOLFILE@    BIN FIXED(31), /*ADDRESS OF FILEDATA INFORMATION       */
3 VOLOSNAM    CHAR(8),       /* OFF-SITE LOCATION NAME               */
3 VOLOSPNM    CHAR(8),       /* PREVIOUS OFF-SITE LOCATION           */
3 VOLCKDAT    DEC FIXED(7),  /*DATE CHECKIN EXPIRES (RTN TO VAULT)   */
3 VOLOSDAT    DEC FIXED(7),  /*DAT VOL                               */
3 VOLOSLOT    BIN FIXED(31), /*OFFSITE SLOT NUMBER                   */
3 VOLOPEN     BIN FIXED(31), /*# TIMES TAPE OPENED                   */
3 VOLCLN#     BIN FIXED(31), /*# TIMES USED SINCE CHAR(EANED         */
3 VOLERRW     BIN FIXED(15), /* NUMBER OF TEMPORARY WRITE ERRORS     */
3 VOLERRR     BIN FIXED(15), /* NUMBER OF TEMPORARY READ ERRORS      */
3 VOLCLEN     BIN FIXED(15), /* # TIMES TAPE CHAR(EANED              */
3 VOLFILES    BIN FIXED(15), /* NUMBER OF FILES ON THE VOLUME        */
3 VOLCLEAN    DEC FIXED(7), /*DATE TAPE LAST CHAR(EANED (YYYYDDDC)   */
3 VOLDATE1    DEC FIXED(7), /*DATE USED FOR 1ST TIME (YYYYDDDC)      */
3 VOLWIPED    DEC FIXED(7), /*DATE VOLUME DATA WIPED (YYYYDDDC)      */
3 VOLLABEL    CHAR(3), /* TAPE LABEL TYPE (SL,NSL...)                */
3 VOLDEN      CHAR(1), /* TAPE TYPE/DENSITY                          */
/*
  VOLD556 = X'43'                 7 TRACK 556 BPI
  VOLD8ØØ = X'83'                 7 & 9 TRACK 8ØØ BPI
  VOLD16ØØ = X'C3'                9 TRACK 16ØØ BPI
  VOLD625Ø = X'D3'                9 TRACK 625Ø BPI
```



```
  VOLD348Ø = X'Ø1'                3480 (CARTRIDGE)
  VOLD349Ø = X'Ø2'                3490 (CARTRIDGE)
  VOLD359Ø = X'Ø3'                3590 (CARTRIDGE)
*/
3 VOLTRTCH    CHAR(1), /* RECORDING TECHNIQUE                 */
/*
  VOLTEVEN = X'23'                EVEN PARITY, NO TRANSLATION
  VOLTTRAN = X'3B'                ODD PARITY, WITH TRANSLATION
  VOLTCONV = X'13'                ODD PARITY, WITH CONVERSION
  VOLTTREV = X'2B'                EVEN PARITY, WITH TRANSLATION
  VOLTCOMP = X'Ø8'                348Ø COMPRESSED MODE
  VOLTNOCM = X'Ø4'                348Ø NON COMPRESSED MODE
  VOLT1TRK = X'42'                349Ø? "FUTURE DEVELOPMENT"
  VOLT2TRK = X'82'                349Ø? "FUTURE DEVELOPMENT"
  VOLT4TRK = X'C2'                349Ø? "FUTURE DEVELOPMENT"
*/
3 VOLCREAT CHAR(1), /*    VOLUME CREATOR INFORMATION           */
/*
  VOLCNORM = C'N'         VOLUME CREATED BY OPEN, CHAR(OSE, EOV
  VOLCINIT = C'I'         VOLUME RANGE-DEFINED AND INITIALIZEDVØ12ØØØ3
  VOLCRNGI = C'R'         VOLUME RANGE-DEFINED, BUT UNUSED    VØ12ØØØ3
  VOLCCONV = C'C'         VOLUME CREATED BY CONVERSION
  VOLCNLNK = C'L'         VOLUME HAS NO INTER-VOLUME LINKAGE (HSM)
  VOLC1FIL = C'F'         ONLY FIRST FILE'S INFO IS KEPT (RPRT DIST)
  VOLC1FNL = C'B'         ONLY FIRST FILE'S INFO, NO INTER-VOL LINKAGE
*/
3 VOLSTAT1 CHAR(1), /*    VOLUME STATUS                 */
/*
  VOLSACTV = C'A'         VOLUME IS ACTIVE
  VOLSSCR = C'S'          VOLUME CAN BE USED AS SCRATCH
  VOLSDEL = C'D'          VOLUME IS DELETED (CAN BE RE-INITED)
  VOLSDMGE = C'E'         VOLUME IS DAMAGED (HAS PERMANENT I/O ERROR)
*/
3 VOLSTAT2 CHAR(1), /*    VOLUME INFORMATION STATUS      */
/*
  VOLSDSNS = C'D'         DSNS EITHER IN DB OR ON LABELS UNKNOWN
*/
3 VOLCHECK CHAR(1), /*    VOLUME CHECK IN/OUT INFO       */
/*
  VOLCHKAU = C'A'         VOLUME WAS AUTO CHECKED IN (FROM OFFSITE)
  VOLCHKIN = C'I'         VOLUME IS MANUALLY CHECKED IN (FROM OFFSITE)
  VOLCHKOT = C'O'         VOLUME IS CHECKED OUT (TO OFFSITE)
  VOLCHKPR = C'P'         VOLUME IS CHECKED IN PERMANENTLY
*/
3 VOLSFUTR    CHAR(3),       /* FUTURE STATUS BYTES                 */
3 VOLACCT     CHAR(44),      /* ACCOUNTING DATA                     */
3 VOLATWER    BIN FIXED(15), /* ACCUMULATED TEMPORARY WRITE ERRORS  */
3 VOLATRER    BIN FIXED(15), /* ACCUMULATED TEMPORARY READ ERRORS   */
3 VOLAPERR    BIN FIXED(15), /* ACCUMULATED PERM. ERR. (READ/WRITE) */
3 VOLMAXFL    BIN FIXED(15), /* MAX NUMBER OF FILES ON VOLUME CHAIN */
```

```
3 VOLMAXVL    BIN FIXED(15), /* MAX NUMBER OF VOLUMES IN CHAIN       */
3 VOLMEDIA    CHAR(1), /* VOLUME MEDIA TYPE                          */
/*
  VOLMVIRT = C'V'         VIRTUAL VOLUME
*/
 2 FILE,
3 FILSEQ   BIN FIXED(15), /* FILE SEQUENCE                           */
3 FILDSNLN BIN FIXED(15), /* LENGTH OF THE DATASET NAME              */
3 FILDSN   CHAR(44),      /* DATASET NAME                            */
3 FILBLK#  BIN FIXED(31), /* BLOCK COUNT                             */
3 FILBLKSZ BIN FIXED(31), /* BLOCK SIZE                              */
3 FILLRECL BIN FIXED(31), /* LOGICAL RECORD LENGTH                   */
3 FILDATEX DEC FIXED(7),  /* EXPIRATION DATE                         */
3 FILCNTX  BIN FIXED(15), /* NUMBER OF COPIES/DAYS SINCE LAST USE    */
3 FILXFLAG CHAR(1),       /* EXPIRATION INDICATOR                    */
/*
  FILXDATE = 1            EXPIRATION IS A DATE (SEE FILDATEX)
  FILXPERM = 2            NEVER EXPIRE
  FILXCATL = 3            CATALOG CONTROLLED EXPIRATION
  FILXCYCL = 4            CYCLICAL EXPIRATION (SEE FILCNTX)
  FILXLUSE = 5            DAYS SINCE LAST USE (SEE FILCNTX)
  FILXUSER = 6            USER DEFINED EXPDTE (88UUU)
  FILXRETN = 7            RETPD, ALREADY CONVERTED TO A DATE
  FILXCTCR = 8            CAT CRTL, THEN USE DATEX
  FILXCTUN = 9            CAT CRTL, THEN CALC DATEX USING CNTX
  FILXIMMD = 1Ø           IMMEDIATE EXPIRE
  FILXCYRT = 11           CYCLE(XCNT)+RETN(DATEX)
*/
3 FILRECFM CHAR(3),       /* RECORD FORMAT (EG F, VBA, DS, U)        */
3 FILABEND CHAR(1),       /* Y - DATASET CHAR(OSED ABNORMALLY        */
3 FILCONT  CHAR(1),       /* Y - FILE CONTINUES ON VOLNEXT TAPE      */
3 FILEXPIR CHAR(1),       /* Y - FILE EXPIRED                        */
3 FILTPSTK CHAR(1),       /* FOR USE BY TAPE STACKING SOFTWARE       */
3 FILENEXT BIN FIXED(15), /* OFFSET TO NEXT FILEDATA (FROM THIS ONE) */
3 FILERRW  BIN FIXED(15), /* NUMBER OF WRITE ERRORS                  */
3 FILERRR  BIN FIXED(15), /* NUMBER OF READ ERRORS                   */
3 FILJOBNC CHAR(8),       /* CREATING JOBNAME                        */
3 FILSTEPC CHAR(8),       /* CREATING STEP NAME                      */
3 FILPROGC CHAR(8),       /* CREATING PROGRAM NAME                   */
3 FILDDNMC CHAR(8),      /* CREATING DDNAME                          */
3 FILUNUTC CHAR(4),      /* 4 DIGIT CREATING UNIT                    */
3 FILDATEC DEC FIXED(7), /* CREATING DATE (YYYYDDDC)                 */
3 FILTIMEC DEC FIXED(7), /* CREATING TIME (ØHHMMSSC)                 */
3 FILJOBNL CHAR(8),      /* JOBNAME LAST USED BY                     */
3 FILSTEPL CHAR(8),      /* STEPNAME LAST USED                       */
3 FILPROGL CHAR(8),      /* PROGRAM LAST USED                        */
3 FILDDNML CHAR(8),      /* DDNAME LAST USED BY                      */
3 FILUNUTL CHAR(4),      /* 4 DIGIT UNIT ON WHICH LAST USED          */
3 FILDATEL DEC FIXED(7), /* DATE LAST USED (YYYYDDDC)                */
3 FILTIMEL DEC FIXED(7), /* TIME LAST USED (ØHHMMSSC)                */
```



```
3 FILCRSID CHAR(4),      /* JES ID OF CREATING SYSTEM, OR BLANKS    */
3 FILOPEN  BIN FIXED(15);/* NO TIMES FILE OPENED (1=JUST CREATED    */

DCL RECOUT CHAR(8Ø) VAR; /* OUT RECORD */
DCL VOLSERS CHAR(6) INIT('');
DCL INDNSCR  BIT;        /* VOLUME STATUS SCRATCH ?                 */
DCL FSEQ  BIN FIXED(15);

ON ERROR SNAP SYSTEM;

DCL NEOF INIT('1'B) BIT;
ON ENDFILE(IN) NEOF='Ø'B;

   CALL PUTOUT(' PROFILE NOPREF');
   CALL PUTOUT('');

   READ FILE(IN) INTO(REC_IN);
   DO WHILE(NEOF);
      IF VOLSERS ¬= VOLSER
      THEN DO;
           INDNSCR='Ø'B;
           VOLSERS=VOLSER;
           FSEQ=1;
           RECOUT=' RMM ADDVOLUME';
           CALL PUTOUT(VOLSER);
           CALL PUTSTR('RACK',VOLSER);
           CALL PUTSTR('USE','MVS');
           CALL PUTSTR('LABEL','SL');

           IF SUBSTR(VOLSER,1,3) ¬= 'PSM' & VOLDEN ¬= 'Ø1'X
           THEN VOLDEN='Ø1'X;
           IF SUBSTR(VOLSER,1,3) = 'PSM' & VOLDEN ¬= 'Ø3'X
           THEN VOLDEN='Ø3'X;

           SELECT(VOLDEN);
           WHEN('Ø1'X)
                DO;
                CALL PUTSTR('DENSITY','348Ø');
                CALL PUTSTR('MEDIANAME','348Ø');
                CALL PUTSTR('MEDIATYPE','CST');
                CALL PUTSTR('COMPACTION','NONE');
                CALL PUTSTR('RECORDINGFORMAT','18TRACK');
                END;
           WHEN('Ø2'X)
                DO;
                CALL PUTSTR('DENSITY','*');
                CALL PUTSTR('MEDIANAME','349Ø');
                CALL PUTSTR('COMPACTION','*');
                END;
           WHEN('Ø3'X)
```

```
          DO;
          CALL PUTSTR('DENSITY','*');
          CALL PUTSTR('MEDIANAME','359Ø');
          CALL PUTSTR('MEDIATYPE','HPCT');
          CALL PUTSTR('COMPACTION','YES');
          CALL PUTSTR('RECORDINGFORMAT','128TRACK');
          END;
     OTHERWISE ;
     END;
     SELECT(VOLSTAT1);
     WHEN('A')   /* ACTIVE */
          DO;
          CALL PUTSTR('STATUS','MASTER');
          CALL PUTSTR('INITIALIZE','N');
          CALL PUTSTR('RELEASEACTION','SCRATCH');
          CALL NONSCRATCH();
          END;
     WHEN('S')  /* SCRATCH */
          DO;
          CALL PUTSTR('STATUS','SCRATCH');
          CALL PUTSTR('INITIALIZE','N');
          END;
     WHEN('D')  /* DELETED */
          DO;
          CALL PUTSTR('STATUS','SCRATCH');
          CALL PUTSTR('INITIALIZE','Y');
          CALL PUTSTR('DESCRIPTION','OBRISANA');
          END;
     WHEN('E')  /* DAMAGED */
          DO;
          CALL PUTSTR('STATUS','MASTER');
          CALL PUTSTR('INITIALIZE','Y');
          CALL PUTSTR('DESCRIPTION','OSTECENA');
          CALL PUTSTR('RELEASEACTION','REPLACE');
          IF LENGTH(REC_IN) > STG(ZARA_RECORD.VOLUME)
          THEN CALL NONSCRATCH();
          ELSE CALL PUTSTR('OWNER','xxxxxx');
          END;
     END;
     CALL PUTOUT('');
     END;
ELSE FSEQ=FSEQ+1;
IF VOLATWER > Ø | VOLATRRR > Ø | VOLAPERR > Ø
THEN PUT SKIP EDIT(VOLSER,
                   '>> WR ERR >>',VOLATWER,
                   '>> TE ERR >>',VOLATRER,
                   '>> PE ERR >>',VOLAPERR) (A);
IF INDNSCR
THEN CALL ADDDATASET();
READ FILE(IN) INTO(REC_IN);
```



```
   END;
NONSCRATCH: PROC ;
   CALL PUTDATE('ASDATE',FILDATEC);
   CALL PUTNUM('ASTIME',FILTIMEC,6);
   CALL PUTDATE('EXPDT',FILDATEX);
   CALL PUTSTR('DSNAME',FILDSN);
   CALL PUTSTR('JOBNAME',FILJOBNC);
   CALL PUTDATE('READDATE',FILDATEL);
   CALL PUTSTR('OWNER','xxxxxx');
   CALL PUTSTR('PREVVOL',VOLPREV);
   IF VOLACCT ¬= ' '
   THEN CALL PUTSTR('ACCOUNT',''''||SUBSTR(VOLACCT,1,4Ø)||'''');
   INDNSCR='1'B;
END NONSCRATCH;
/*******************************************************************/
ADDDATASET: PROC;
      RECOUT=' RMM ADDDATASET';
      CALL PUTOUT(FILDSN);
      CALL PUTSTR('VOLUME',VOLSER);
      CALL PUTNUM('BLKCOUNT',FILBLK#,7);
      IF FILBLKSZ > 3276Ø
      THEN FILBLKSZ=3276Ø;
      CALL PUTNUM('BLKSIZE',FILBLKSZ,5);
      CALL PUTDATE('CRDATE',FILDATEC);
      CALL PUTNUM('CRTIME',FILTIMEC,6);
      CALL PUTSTR('DEVNUM',FILUNUTC);
      CALL PUTNUM('FILESEQ',FSEQ,4);
      CALL PUTSTR('JOBNAME',FILJOBNC);
      CALL PUTNUM('LABELNUMBER',FILSEQ,4);
      IF FILLRECL > 3276Ø
      THEN FILLRECL=3276Ø;
      CALL PUTNUM('LRECL',FILLRECL,5);
      CALL PUTSTR('RECFM',FILRECFM);
      CALL PUTSTR('SYSID','PSHOST');
      CALL PUTDATE('WRITEDATE',FILDATEC);
      CALL PUTOUT('');
 END ADDDATASET;
/*******************************************************************/
/** PUT STRING PARAMETER INSIDE THE KEYWORD                       **/
/*******************************************************************/
PUTSTR: PROC(KEYWORD,STRING) ;
DCL KEYWORD CHAR(*);
DCL STRING  CHAR(*);
   IF STRING  ¬= ' '
   THEN CALL PUTOUT(KEYWORD!!'('||STRING||')');
END PUTSTR;
/*******************************************************************/
/** PUT NUMERIC PARAMETER INSIDE THE KEYWORD                      **/
/*******************************************************************/
PUTNUM: PROC(KEYWORD,NUM,LEN);
```

```
DCL KEYWORD CHAR(*);
DCL NUM     PIC'9999999';
   IF NUM > Ø
   THEN CALL PUTOUT(KEYWORD||'('!!SUBSTR(NUM,8-LEN)||')');
END PUTNUM;
/*******************************************************************/
/** PUT DATE PARAMETER INSIDE THE KEYWORD                        **/
/*******************************************************************/
PUTDATE:PROC(KEYWORD,DAT) ;
DCL KEYWORD CHAR(*);
DCL DAT     DEC FIXED(7);
DCL DATEP   PIC'9999999';
   IF DAT > Ø
   THEN DO;
        DATEP=DAT;
        IF SUBSTR(DATEP,1,4) ¬= 'ØØØØ' & SUBSTR(DATEP,5,3) ¬= 'ØØØ'
        THEN CALL PUTOUT(KEYWORD!!'('||SUBSTR(DATEP,1,4)||'/'||
                                       SUBSTR(DATEP,5,3)||')');
        END;
END PUTDATE;
/*******************************************************************/
/** PUT ROW OF STATEMENT                                          **/
/*******************************************************************/
PUTOUT: PROC(STRING) ;
DCL STRING CHAR(*);
DCL L BIN FIXED;
L=LENGTH(STRING)+1;
IF LENGTH(RECOUT) + L >  71 | L=1
THEN DO;
     IF L > 1
     THEN RECOUT=RECOUT!!' -';
     PUT FILE(OUT) SKIP EDIT(RECOUT) (A);
     RECOUT='   ';
     END;
RECOUT=RECOUT||' '||STRING;
END PUTOUT ;
END ZARARMM;
```

REXX PROCEDURE FOR VRS CONVERSION

```
"EXECIO * DISKR in (STEM recin. FINIS"
do i=1 to recin.Ø
   if index(recin.i,'Auto Expiration Date Candidate') > Ø ,
   then leave
end
k=Ø
do i=i+2 to recin.Ø
   if index(recin.i,'NUMBER OF RECORDS REPORTED ON') > Ø ,
   then leave
   if index(recin.i,'1AutoMedia') = 1 ,
```

```
      then i=i+5
      zaraauto= substr(recin.i,2)
      PARSE VAR zaraauto name nul1 period nul2 gdg nul3
      perrmm=' '
      Select
        when  substr(period,1,2) = 'CY' THEN
         perrmm= 'CYCLE COUNT('||substr(period,3,3)||')'
        when  substr(period,1,2) = 'RT' THEN
         perrmm= 'DAYS COUNT('||substr(period,3,3)||')'
        when  substr(period,1,2) = 'YR' THEN
         perrmm= 'DAYS COUNT('||substr(period,3,3)*365||')'
        when  period = 'CATLG' THEN
         perrmm= 'WHILECATALOG'
        when  period = 'NEVER' THEN
         perrmm= 'DAYS COUNT(99999)'
        Otherwise perrmm=period
      end
      if index(name,'.*') > Ø & index(name,'.*')=length(name)-1,
      then name=name||'*'
      else do
           name1=name!!' '
           if index(name1,'* ') > Ø then name=name||'.**'
           end
      if gdg='N' ,
      then gdgrmm='NOGDG '
      else gdgrmm='GDG '
      if perrmm ¬= ' ' ,
      then do
           k=k+1
           recout.k = " RMM ADDVRS DSN('"||name||,
                      "') OWNER(HOSTID) ",
                      ||gdgrmm||perrmm
           end
   end
   recout.Ø = k
   "EXECIO * DISKW out (STEM recout. FINIS"
  return
```

The results of the control jobs proved that there is no difference between ZARA and DFSMSrmm scratch pools and we finished the process of validation successfully. We are now ready to cut over to production.

Dragan Nikolic and Emina Spasic
System Programmers

Sorting stem variables using REXX

INTRODUCTION

This is the documentation for the REXX function REXSORT. The function should be invoked from within an MVS REXX EXEC. This function sorts the content of a stem variable on a single key into either ascending or descending sequence.

This REXX function accepts four arguments, the fourth being optional. The first argument is a stem variable (with the terminating period) holding the data records to be sorted, with the total number in stem.0. The maximum record length is 256 bytes; all records must be the same length. The stem name may be a maximum of 32 bytes in length. The second argument is the key length in bytes, a maximum of 256 bytes. The third argument is the start byte (not the offset/displacement) of the key within the data record. The fourth argument is the sequence – ascending or descending (EBCDIC collating); ascending is the default. The sorted output is returned in the stem variable.

The syntax of the function is:

```
                                                 Ascending
REXSORT(-stemname,—key_length,—key_start,
                                                 Descending
```

In keeping with standard REXX practices, the Ascending / Descending requires only the first character to be provided, and that character may be in upper or lower case.

The function returns an integer. This integer will indicate success or failure. An example of the function being invoked:

```
RC = REXSORT(SORTDATA., 1, 1Ø, ‘A’);
```

The different values that may be returned are as follows:

```
-2   IRXEXCOM - LACK OF STORAGE
-1   IRXEXCOM - ERROR CONDITION
Ø    NORMAL
8    STEM NAME SPECIFIED > 32 BYTES
12   NO PERIOD AT END OF STEM NAME
16   STEM NAME CONTAINS INVALID CHARACTERS
2Ø   INVALID NUMBER OF ARGUMENTS (MUST BE THREE or FOUR)
```

```
24   VARIABLE LENGTH STEM RECORDS
28   KEY LENGTH / KEY START > 3 BYTES LONG
32   KEY LENGTH / KEY START NOT NUMERIC
36   KEY LENGTH / KEY START EXTEND BEOND 256 BYTES
4Ø   SEQUENCE FIELD > 1Ø BYTES LONG
44   SEQUENCE VALUE INVALID
48   NON-EXISTENT STEM
52   STEM DATA > 256 BYTESA full example of the function being used:

/* REXX ****************************************************/
"ALLOC F(INPUT) DA(my.input.file) SH REU";
"EXECIO * DISKR INPUT (STEM SORTDATA. FINIS";
say "Ascending or Descending";
parse pull SEQ;
say "REXSORT ended with RC:" REXSORT(SORTDATA., 2, 3, SEQ);
do J = 1 to SORTDATA.0;
     say strip(SORTDATA.J, t, ' ');
end;
exit;
```

This example reads the content of the file assigned to DDname INPUT into the stem variable SORTDATA. The records are then sorted into either ascending or descending sequence. The key is three bytes long, starting in the second byte of the record.

```
TITLE 'REXX FUNCTION TO SORT CONTENT OF STEM VARIABLE'
         PRINT NOGEN
*
*        PROGRAM:     REXSORT
*                     SORT ALL VARIABLES IN STEM ACCORDING TO
*                     USER SPECIFIED SEQUENCE
*
*        ATTRIBUTES:
*                     REENTRANT
*                     AMODE: 31
*                     RMODE: ANY
*
*        ABSTRACT:
*        REXX FUNCTION THAT SORTS DATA STORED WITHIN A STEM VARIABLE
*        INTO THE USER-SPECIFIED SEQUENCE.  THE DATA IN THE STEM MUST
*        MEET THE FOLLOWING CRITERIA:
*           STEM.Ø WILL CONTAIN THE NUMBER OF STEM VALUES
*           ALL OTHER STEM ELEMENTS MUST BE THE SAME LENGTH
*           THE MAXIMUM LENGTH OF A STEM ELEMENT IS 256 BYTES
*           SINGLE KEY WITH CONSTANT DISPLACEMENT WITHIN DATA
*
*        USAGE:
*        RET_CODE = REXSORT(STEM., KEY_START, KEY_LENGTH, SEQUENCE);
*
```

```
*        RET_CODE VALUES:
*        -2                             . IRXEXCOM - LACK OF STORAGE
*        -1                             . IRXEXCOM - ERROR CONDITION
*        Ø                              . NORMAL
*        8                              . STEM NAME SPECIFIED > 32 BYTES
*        12                             . NO PERIOD AT END OF STEM NAME
*        16                             . STEM NAME INVALID CHARACTERS
*        2Ø                             . INVALID NUMBER OF ARGUMENTS
*        24                             . VARIABLE LENGTH STEM RECORDS
*        28                             . KEY LENGTH/START > 3 BYTES
*        32                             . KEY LENGTH/START NOT NUMERIC
*        36                             . KEY LENGTH/START OUTSIDE BOUND
*        4Ø                             . SEQUENCE > 1Ø BYTES
*        44                             . SEQUENCE VALUE INVALID
*        48                             . STEM NON-EXISTENT
*        52                             . STEM DATA > 256 BYTES
         TITLE 'EQUATES, MACROS && CONTROL BLOCKS USED'
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø                       . BAS RETURN REGISTER
R11      EQU   11
R12      EQU   12                       . CSECT BASE REGISTER
R13      EQU   13                       . -> DYNAMIC AREA
R14      EQU   14                       . -> RETURN
R15      EQU   15                       . -> ENTRY POINT
*                                       . RETURN CODE
*
*        MACROS USED:
*              IRXARGTB                 . MAP ARGUMENT TABLE
*              IRXEFPL                  . MAP EXTERNAL FUNCTIONS PLIST
*              IRXEVALB                 . MAP EVALUATION BLOCK
*              IRXSHVB                  . MAP SHARED VARIABLE BLOCK
*              STORAGE                  . STORAGE ACQUIRE AND RELEASE
         TITLE 'MAIN CSECT PROCESS'
REXSORT  CSECT
REXSORT  AMODE 31
REXSORT  RMODE ANY
         LA    R14,Ø(,R14)              . VALIDITY OF R14
         BSM   R14,RØ                   . CURRENT ADDRESSING MODE
         BAKR  R14,RØ                   . ESTABLISH LINKAGE
         LR    R12,R15                  . 12 -> EPA
         USING REXSORT,R12              . CSECT ADDRESSABILITY
```



```
         STORAGE OBTAIN,              . ACQUIRE DYNAMIC AREA         *
               ADDR=(R13),                                           *
               LENGTH=DYNLEN,                                        *
               SP=Ø
         MVC   4(4,R13),=C'F1SA'      . INDICATE FORMAT OF SAVE AREA
         USING DYNAREA,R13            . DSECT ADDRESSABILITY
         XC    @IRXEXCOM,@IRXEXCOM    . INDICATE IRXEXCOM NOT LOADED
         BAS   R1Ø,REXXVECT           . REXX VECTOR PROCESSING
         BAS   R1Ø,ARGUMENT           . PROCESS ARGUMENTS
         CLC   RETCODE,=F'Ø'          . Q. ARGUMENTS VALID?
         BNE   AØØØ1                  . A. NO
         BAS   R1Ø,SORTSTEM           . SORT CONTENT OF STEM
*
AØØØ1    EQU   *
*
         BAS   R1Ø,TERMINAT           . TERMINATION
         STORAGE RELEASE,             . RELEASE DYNAMIC STORAGE      *
               ADDR=(R13),                                           *
               LENGTH=DYNLEN,                                        *
               SP=Ø
         SLR   R15,R15                . 15 - RETURN CODE
         PR                           . ADIOS
         TITLE 'REXX VECTOR PROCESSING AND LOAD IRXEXCOM'
*        PROCESS THE TWO ARGUMENTS PASSED TO REXX FUNCTIONS
*        THE ADDRESS OF THE REXX ENVIRONMENT BLOCK (OPTIONAL)
*        THE ADDRESS OF THE EXTERNAL FUNCTION PARAMETER LIST
*        LOAD THE REXX SERVICE ROUTINE IRXEXCOM
*        FORMAT IRXEXCOM PARAMETER LIST
*
*        REGISTER USAGE
*        Ø                            . -> ENVIONMENT BLOCK
*        1                            . -> EXTERNAL FUNCTION PLIST
*        2                            . -> PARSED PARAMETER LIST
*
REXXVECT EQU   *
*
         EREG  RØ,R1                  . EXTRACT CALLER'S REGISTERS
         ST    RØ,@REXX               . SAVE REXX ENVIRONMENT BLOCK ->
         ST    R1,@EFPL               . SAVE EXTERNAL FUNCTION PLIST
         USING EFPL,R1                . IRXEFPL DSECT ADDRESSABILITY
         L     R2,EFPLARG             . 2 -> PARSED ARGUMENT LIST
         ST    R2,@ARGTAB             . SAVE
         L     R2,EFPLEVAL            . 2 -> EVALUATION BLOCK VECTOR
         L     R2,Ø(,R2)              . 2 -> EVALUATION BLOCK
         ST    R2,@EVALBLK            . SAVE
         DROP  R1                     . DSECT NOT REQUIRED
*
         LOAD  EP=IRXEXCOM            . LOAD IRXECOM
         ST    RØ,@IRXEXCOM           . SAVE EPA
*
```

```
         LA    R1,CIRXEXCOM            . 1 -> IRXEXCOM STRING
         ST    R1,@CSTR                . SAVE IN PARAMETER LIST
         XC    @DUMMY1(L'@DUMMY1+L'@DUMMY2),@DUMMY1
         LA    R1,SHVARBLK             . 1 -> SHARED VARIABLE BLOCK
         ST    R1,@SHVB                . SAVE IN PARAMETER LIST
         OI    @SHVB,X'8Ø'             . FLAG END OF ARGUMENTS
*
         BR    R1Ø                     . RETURN
         TITLE 'PROCESS INPUT ARGUMENTS'
*        PROCESS ARGUMENTS - VALIDATE ETC.
*        THREE MANDATORY ARGUMENTS, ONE OPTIONAL
*        1. STEM VARIABLE - MUST END IN PERIOD
*           NAME MUST BE VALID FORMAT
*        2. KEY START - MUST BE NUMERIC
*        3. KEY LENGTH - MUST BE NUMERIC
*           LENGTH + OFFSET MUST BE < 257
*        4. SEQUENCE - DEFAULT (A)SCENDING
*                      OPTIONALLY (D)ESCENDING
*
*        REGISTER USAGE
*        1                             . ARGUMENT COUNT
*        2                             . -> CURRENT ARG TABLE ENTRY
*        3                             . WORK
*        4                             . -> CURRENT ARGUMENT VALUE
*        5                             . CURRENT ARGUMENT LENGTH
*        6                             . -> SAVED VALUE
*                                      . WORK
*        7                             . LENGTH OF STEM NAME
*        8                             . ERROR VALUE
*        1Ø                            . RETURN
*
ARGUMENT EQU   *
*
         L     R2,@ARGTAB              . 2 -> ARGUMENT TABLE
         USING ARGTABLE_ENTRY,R2       . DSECT ADDRESSABILITY
         SLR   R1,R1                   . 1 - ZERO (ARGUMENT COUNT)
         MVI   SEQ,C'A'                . DEFAULT SEQUENCE
*
CØØØ1    EQU   *
*                                      . 4 -> ARGUMENT STRING
*                                      . 5 - ARGUMENT STRING LENGTH
         LM    R4,R5,ARGTABLE_ARGSTRING_PTR
         LTR   R5,R5                   . Q. LENGTH NEGATIVE?
         BM    CØØØ9                   . A. YES - LAST ARGUMENT
         LA    R1,1(,R1)               . INCREMENT ARGUMENT COUNT
         CH    R1,=H'1'                . Q. ARGUMENT ONE?
         BNE   CØØØ3                   . A. NO
         LA    R8,8                    . SET ERROR CODE
         CH    R5,=Y(L'STEM)           . Q. VARIABLE NAME TOO GREAT?
         BH    CØØ1Ø                   . A. YES - ERROR
```

```
         LA    R6,Ø(R5,R4)              . 6 -> AFTER LAST BYTE OF NAME
         LA    R8,12                    . SET ERROR CODE
         BCTR  R6,RØ                    . 6 -> LAST BYTE OF STEM NAME
         CLI   Ø(R6),C'.'               . Q. PERIOD PRESENT?
         BNE   CØØ1Ø                    . A. NO - ERROR
         LA    R8,16                    . SET ERROR CODE
         MVC   STEM,SPACES              . INITIALIZE SAVED STEM VALUE
         LA    R6,STEM                  . 6 -> SAVED STEM NAME VALUE
         SLR   R7,R7                    . LENGTH OF STEM NAME
*
CØØØ2    EQU   *
*
         SLR   R3,R3                    . 3 - ZERO
         IC    R3,Ø(,R4)                . 3 - BYTE OF STEM VARIABLE
         LA    R3,TRTABLE(R3)           . 3 - CHARACTER FROM TABLE
         CLI   Ø(R3),X'ØØ'              . Q. VALID CHARACTER?
         BE    CØØ1Ø                    . A. NO
         MVC   Ø(1,R6),Ø(R4)            . MOVE BYTE TO SAVE STEM
         LA    R4,1(,R4)                . 4 -> NEXT BYTE OF STEM NAME
         LA    R6,1(,R6)                . 6 -> NEXT BYTE OF SAVED NAME
         LA    R7,1(,R7)                . INCREMENT BYTES IN STEM NAME
         BCT   R5,CØØØ2                 . LOOP THROUGH STEM NAME
         ST    R7,#STEM                 . SAVE LENGTH
*                                       . 2 -> NEXT ARGUMENT DATA
         LA    R2,ARGTABLE_NEXT-ARGTABLE_ENTRY(,R2)
         B     CØØØ1                    . PROCESS NEXT ARGUMENT
*
CØØØ3    EQU   *
*
         CH    R1,=H'3'                 . Q. SECOND OR THIRD ARGUMENT?
         BH    CØØØ6                    . A. NO
         LA    R8,28                    . SET ERROR CODE
         CH    R5,=H'3'                 . Q. ARGUMENT LENGTH > THREE?
         BH    CØØ1Ø                    . A. YES - ERROR
         MVC   TEST,ZEROS               . PERFORM NUMERIC VALIDATION
         BCTR  R5,RØ                    . DECREMENT FOR EXECUTE
         EX    R5,MVZNUM                . MOVE THE ZONES
         LA    R8,32                    . SET ERROR CODE
         CLC   TEST,ZEROS               . Q. ARGUMENT NUMERIC?
         BNE   CØØ1Ø                    . A. NO - ERROR
         AH    R5,=H'112'               . GET READY FOR PACK
*                                       . 112 - 7 SHIFTED LEFT 4 BITS
         EX    R5,PACKNUM               . PACK THE VALUE
         CVB   R6,DWORD                 . CONVERT TO BINARY
         CH    R1,=H'2'                 . Q. SECOND ARGUMENT?
         BNE   CØØØ4                    . A. NO
         BCTR  R6,RØ                    . DECREMENT START FOR OFFSET
         STH   R6,KEYDISP               . SAVE KEY DISPLACEMENT
         B     CØØØ5                    . CONTINUE
*
```

```
C0004    EQU   *
*
         STH   R6,KEYLEN                .  SAVE KEY LENGTH
         LA    R8,36                    .  SET ERROR CODE
         AH    R6,KEYDISP               .  ADD LENGTH TO DISPLACEMENT
         CH    R6,=H'256'               .  Q. WITHIN BOUNDS?
         BH    C0010                    .  A. NO - ERROR
*
C0005    EQU   *
*                                       .  2 -> NEXT ARGUMENT DATA
         LA    R2,ARGTABLE_NEXT-ARGTABLE_ENTRY(,R2)
         SLR   R8,R8
         B     C0001                    .  PROCESS NEXT ARGUMENT
*
C0006    EQU   *
*
         LA    R8,40                    .  SET ERROR CODE
         CH    R5,=H'10'                .  Q. ARGUMENT LENGTH > TEN?
         BH    C0010                    .  A. YES - ERROR
         BCTR  R5,R0                    .  DECREMENT FOR EXECUTE
         EX    R5,OCUP                  .  ENSURE UPPER-CASE
         CLI   0(R4),C'A'               .  Q. FIRST CHARACTER AN 'A'?
         BNE   C0007                    .  A. NO
         LA    R6,ASCEND                .  6 -> ASCENDING STRING
         B     C0008
*
C0007    EQU   *
*
         LA    R6,DESCEND               .  6 -> DESCENDING STRING
*
C0008    EQU   *
*
         LA    R8,44                    .  SET ERROR CODE
         EX    R5,CLCAORD               .  VALIDATE SEQUENCE
         BNE   C0010
         MVC   SEQ,0(R4)                .  SAVE SEQUENCE
*                                       .  2 -> NEXT ARGUMENT
         LA    R2,ARGTABLE_NEXT-ARGTABLE_ENTRY(,R2)
*                                       .  4 -> ARGUMENT STRING
*                                       .  5 - ARGUMENT STRING LENGTH
         LM    R4,R5,ARGTABLE_ARGSTRING_PTR
         SLR   R8,R8                    .  VALID RETURN
         LTR   R5,R5                    .  Q. LENGTH NEGATIVE?
         BM    C0010                    .  A. YES
         LA    R8,20                    .  SET ERROR CODE
         B     C0010                    .  EXCESS ARGUMENTS
*
C0009    EQU   *
*
         CH    R1,=H'3'                 .  Q. VALID NUMBER OF ARGUMENTS?
```



```
         BE    CØØ1Ø                    . A. YES
         CH    R1,=H'4'                 . Q. VALID NUMBER OF ARGUMENTS?
         BE    CØØ1Ø                    . A. YES
         LA    R8,2Ø                    . SET ERROR CODE
*
CØØ1Ø    EQU   *
*
         DROP  R2                       . DSECT NOT REQUIRED
         ST    R8,RETCODE               . SAVE RETURN CODE
         BR    R1Ø
CLCAORD  CLC   *-*(*-*,R6),*-*(R4)      . EXECUTED CHECK FOR ASC/DESC
MVZNUM   MVZ   TEST(*-*),*-*(R4)        . EXECUTED NUMERIC TEST
OCUP     OC    *-*(*-*,R4),SPACES       . EXECUTED UPPER CASE CONV
PACKNUM  PACK  DWORD(*-*),*-*(*-*,R4)   . EXECUTED PACK
         TITLE 'SORT DATA IN STEM VARIABLE'
*        DETERMINE NUMBER OF ENTRIES IN STEM
*            FROM STEM.Ø
*        IF > 1 ENTRY
*            DETERMINE LENGTH OF STEM ENTRY
*                FROM STEM.1
*            ACQUIRE STORAGE FOR ALL VARIABLES: STEM.N
*            LOAD STEM.1, STEM.2, ... INTO ARRAY
*            DO SORT PROCESS
*            LOAD DATA BACK INTO STEM VARIABLE
*        FI
*
*        REGISTER USAGE
*        1                              . WORK
*        2                              . -> SHARED VARIABLE BLOCK
*        3                              . WORK
*                                       . STEM COUNT
*        4                              . -> CURRENT ARRAY ENTRY
*        5                              . WORK
*
SORTSTEM EQU   *
*
         ST    R1Ø,DSAVE                . SAVE RETURN ADDRESS
*
         BAS   R1Ø,NUMENTS              . GET THE NUMBER OF STEM ENTRIES
         CLC   RETCODE,=F'Ø'            . Q. ANY ERRORS?
         BNE   DØØØ2                    . A. YES
         L     R3,#STEMVAR              . 3 - NUMBER OF STEM VARIABLES
         CH    R3,=H'2'                 . Q. ARE WE TALKING SORT?
         BL    DØØØ2                    . A. NO
*
         BAS   R1Ø,GETSTEML             . GET THE STEM LENGTH
*
         L     R3,#STEMVAR              . 3 - NUMBER OF STEM VARIABLES
         LH    R5,#DATALEN              . 5 - STEM LENGTH
         MR    R4,R3                    . 5 - BYTES REQUIRED FOR ARRAY
```

```
         ST    R5,#ARRAY                . SAVE IT
         STORAGE OBTAIN,                . GET THE ARRAY                *
               ADDR=(R4),                                              *
               LENGTH=(R5),                                            *
               SP=Ø
         ST    R4,@ARRAY                . SAVE ITS ADDRESS
*
         BAS   R1Ø,POPARRAY             . POPULATE THE ARRAY
         CLC   RETCODE,=F'Ø'            . Q. ANY PROBLEMS?
         BNE   DØØØ1                    . A. YES
*
         BAS   R1Ø,SORTDATA             . SET UP AND DO SORT
*
         BAS   R1Ø,RELOADST             . PUT STEM DATA INTO SEQUENCE
*
DØØØ1    EQU   *
*
         L     R4,@ARRAY                . 4 -> ARRAY
         L     R5,#ARRAY                . 5 - LENGTH OF ARRAY
         STORAGE RELEASE,               . FREE UP ARRAY                *
               ADDR=(R4),                                              *
               LENGTH=(R5),                                            *
               SP=Ø
*
DØØØ2    EQU   *
*
         L     R1Ø,DSAVE                . RESTORE RETURN ADDRESS
         BR    R1Ø
         TITLE 'TERMINATION ROUTINE'
*        DELETE IRXEXCOM IF LOADED
*        SET UP REXX FUNCTION RETURN CODE
*        PUT RETURN VALUE INTO REXX EVALUATION BLOCK
*
*        REGISTER USAGE
*        1                              . LENGTH OF RETURN VALUE
*        2                              . -> RETURN VALUE
*                                       . -> EVAL BLOCK
*        3                              . BINARY RETURN VALUE
*                                       . EVAL BLOCK SIZE
*        4                              . LENGTH OF EDITED REURN VALUE
*
TERMINAT EQU   *
*
         ICM   R8,B'1111',@IRXEXCOM     . Q. IRXEXCOM LOADED?
         BZ    EØØØ1                    . A. YES
         DELETE EP=IRXEXCOM             . DECREMENT RESPONSIBILITY
*
EØØØ1    EQU   *
*
         SLR   R1,R1                    . 1 - ZERO
```

```
         LA    R2,RCDATA               . 2 -> OUTPUT DATA
         MVC   RCDATA,SPACES           . INITIALIZE OUTPUT
         L     R3,RETCODE              . 3 - RETURN CODE
         LTR   R3,R3                   . Q. RETURN CODE NEGATIVE?
         BNM   EØØØ2                   . A. NO
         MVI   Ø(R2),C'-'              . OUTPUT NEGATIVE SIGN
         LA    R1,1(,R1)               . INCREMENT BYTES OUTPUT
         LA    R2,1(,R2)               . 2 -> NEXT OUTPUT BYTE
*
EØØØ2    EQU   *
*
         CVD   R3,DWORD                . PACK IT
         MVC   VARWORK,MASK8           . MOVE EDIT MASK TO WORK AREA
         ED    VARWORK,DWORD+4         . EDIT THE DATA
         LA    R3,VARWORK              . 3 -> EDITED DATA
         LA    R4,L'VARWORK            . 4 - LENGTH OF EDITED DATA
*
EØØØ3    EQU   *
*
         CLI   Ø(R3),C' '              . Q. SIGNIFICANT?
         BNE   EØØØ4                   . A. YES
         LA    R3,1(,R3)               . 3 -> NEXT BYTE
         BCT   R4,EØØØ3                . LOOP
*
EØØØ4    EQU   *
*
         MVC   Ø(1,R2),Ø(R3)           . MOVE OUT BYTE
         LA    R1,1(,R1)               . INCREMENT BYTES OUTPUT
         LA    R2,1(,R2)               . 2 -> NEXT OUTPUT BYTE
         LA    R3,1(,R3)               . 3 -> NEXT INPUT BYTE
         BCT   R4,EØØØ4                . LOOP
         ST    R1,#RCDATA              . NUMBER OF BYTES
*
         L     R2,@EVALBLK             . 2 -> EVAL BLOCK
         USING EVALBLOCK,R2            . DSECT ADDRESSABILITY
         L     R3,EVALBLOCK_EVSIZE     . 3 - LENGTH
         CH    R3,=H'3'                . Q. AT LEAST THREE DOUBLES?
         BL    EØØØ5                   . A. NO
         MVC   EVALBLOCK_EVDATA(4),RCDATA . SET RESULT
         MVC   EVALBLOCK_EVLEN(4),#RCDATA
         DROP  R2
*
EØØØ5    EQU   *
*
         BR    R1Ø
         TITLE 'DETERMINE THE NUMBER OF STEM ENTRIES'
*        SET UP PARAMETER LIST FOR IRXEXCOM TO FETCH
*        THE STEM.Ø VALUE
*        BUILD THE VARIABLE NAME
*        GET THE DATA AND SAVE IT FOR FUTURE GENERATIONS
```

```
*
*         REGISTER USAGE
*         1                               . WORK
*         2                               . -> SHARED VARIABLE BLOCK
*         3                               . WORK
*
NUMENTS   EQU   *
*
          ST    R1Ø,FSAVE                 . SAVE RETURN ADDRESS
*
          LA    R2,SHVARBLK               . 2 -> SHARED VARIABLE BLOCK
          XC    Ø(L'SHVARBLK,R2),Ø(R2)    . INITIALIZE IT
          USING SHVBLOCK,R2               . DSECT ADDRESSABILITY
          MVI   SHVCODE,SHVSYFET          . SPECIFY ACTION
          LA    R1,NEWSTEM                . 1 -> NEW STEM NAME
          ST    R1,SHVNAMA                . SAVE IN DSECT
          LA    R1,L'BUFFER               . 1 - LENGTH OF BUFFER
          ST    R1,SHVBUFL                . SAVE IN DSECT
          LA    R1,BUFFER                 . 1 -> BUFFER
          ST    R1,SHVVALA                . SAVE IN DSECT
*
          ZAP   #VARS,=P'+Ø'              . INITIALIZE NUMBER OF VARIABLES
          BAS   R1Ø,BLDVARNM              . DEVELOP STEM NAME
          MVC   SHVNAML,#NEWSTEM          . LENGTH OF VARIABLE NAME
          L     RØ,@REXX                  . Ø -> REXX ENVIRONMENT BLOCK
          LA    R1,PIRXEXCOM              . 1 -> PARAMETER LIST
          L     R15,@IRXEXCOM             . 15 -> EPA IRXEXCOM
          BASSM R14,R15                   . INVOKE IRXEXCOM
          LTR   R15,R15                   . Q. RETURN CODE < ZERO?
          BM    FØØØ2                     . A. YES - ERROR
          CLI   SHVRET,SHVNEWV            . Q. NON-EXISTENT STEM?
          BNE   FØØØ1                     . A. NO
          LA    R15,48                    . A. YES - SET ERROR
          B     FØØØ2                     . OUT OF HERE
*
FØØØ1     EQU   *
*
          L     R1,SHVVALL                . 1 - LENGTH OF VALUE
          BCTR  R1,RØ                     . DECREMENT FOR EXECUTE
          AH    R1,=H'112'                . PREPARE FOR EXECUTE
*                                         . 112 - 7 SHIFTED LEFT 4 BITS
          L     R3,SHVVALA                . 3 -> VALUE
          EX    R1,PACKVAL                . PACK THE VALUE
          CVB   R3,DWORD                  . CONVERT TO BINARY
          ST    R3,#STEMVAR               . SAVE NUMBER OF STEM VARIABLES
          B     FØØØ3
*
FØØØ2     EQU   *
*
          ST    R15,RETCODE
```

```
*
FØØØ3    EQU   *
*
         L     R1Ø,FSAVE               . RESTORE RETURN ADDRESS
         BR    R1Ø
         DROP  R2
PACKVAL  PACK  DWORD(*-*),*-*(*-*,R3)  . EXECUTED PACK
         TITLE 'DETERMINE LENGTH OF STEM DATA'
*        RETRIEVE FIRST STEM VARIABLE STEM.1.
*        SAVE THIS LENGTH FOR FUTURE GENERATIONS.
*        CODE REQUIRES ALL STEM DATA (EXCEPT STEM.Ø) TO BE THE SAME
*        LENGTH
*
*        REGISTER USAGE
*        1                             . WORK
*        2                             . -> SHVARBLK
*
GETSTEML EQU   *
*
         ST    R1Ø,GSAVE               . SAVE RETURN ADDRESS
*
         LA    R2,SHVARBLK             . 2 -> SHARED VARIABLE BLOCK
         USING SHVBLOCK,R2             . DSECT ADDRESSABILITY
*
         ZAP   #VARS,=P'+1'            . GET STEM.1
         BAS   R1Ø,BLDVARNM            . DEVELOP STEM NAME
         MVC   SHVNAML,#NEWSTEM        . LENGTH OF VARIABLE NAME
         L     RØ,@REXX                . Ø -> REXX ENVIRONMENT BLOCK
         LA    R1,PIRXEXCOM            . 1 -> PARAMETER LIST
         L     R15,@IRXEXCOM           . 15 -> EPA IRXEXCOM
         BASSM R14,R15                 . INVOKE IRXEXCOM
         LTR   R15,R15                 . Q. RETURN CODE ZERO?
         BZ    GØØØ1                   . A. YES
         CH    R15,=H'1'               . Q. RETURN CODE ONE?
         BNE   GØØØ2                   . A. NO -ERROR
*
GØØØ1    EQU   *
*
         L     R1,SHVVALL              . 1 - LENGTH OF VALUE
         STH   R1,#DATALEN             . SAVE VALUE LENGTH
         CH    R1,=H'256'              . Q. LENGTH ACCEPTABLE?
         BNH   GØØØ3
         LA    R15,52                  . SET RETURN CODE
*
GØØØ2    EQU   *
*
         ST    R15,RETCODE             . SAVE RETURN CODE
*
GØØØ3    EQU   *
*
```



```
         L     R1Ø,GSAVE                 . RESTORE RETURN ADDRESS
         BR    R1Ø
         DROP  R2
         TITLE 'MOVE THE DATA FROM THE STEM VARIABLE INTO THE ARRAY'
*        MOVE THE FIRST STEM INTO ITS LOCATION IN THE ARRAY
*        FETCH THE REMAINING STEM VARIABLES AND PUT THEM IN THE ARRAY
*
*        REGISTER USAGE
*        1                               . WORK
*        2                               . -> SHARED VARIABLE BLOCK
*        3                               . NUMBER OF STEM VARIABLES
*        4                               . -> ARRAY
*        5                               . - ELEMENT LENGTH
*
POPARRAY EQU   *
*
         ST    R1Ø,HSAVE                 . SAVE RETURN ADDRESS
*
         LA    R2,SHVARBLK               . 2 -> SHARED VARIABLE BLOCK
         USING SHVBLOCK,R2               . DSECT ADDRESSABILITY
         L     R3,#STEMVAR               . 3 - NUMBER OF STEM VARIABLES
         BCTR  R3,RØ                     . ALREADY GOT FIRST
         L     R4,@ARRAY                 . 4 -> ARRAY
         LH    R5,#DATALEN               . 5 - LENGTH OF DATA
         BCTR  R5,RØ                     . DECREMENT FOR EXECUTE
*
HØØØ1    EQU   *
*
         L     R1,SHVVALA                . 1 -> VALUE
         EX    R5,MVCDATA                . MOVE DATA INTO ARRAY
         LA    R4,1(R5,R4)               . 4 -> NEXT ENTRY IN ARRAY
*
         AP    #VARS,=P'+1'              . GET STEM. + 1
         BAS   R1Ø,BLDVARNM              . DEVELOP STEM NAME
         MVC   SHVNAML,#NEWSTEM          . LENGTH OF VARIABLE NAME
         L     RØ,@REXX                  . Ø -> REXX ENVIRONMENT BLOCK
         LA    R1,PIRXEXCOM              . 1 -> PARAMETER LIST
         L     R15,@IRXEXCOM             . 15 -> EPA IRXEXCOM
         BASSM R14,R15                   . INVOKE IRXEXCOM
         LTR   R15,R15                   . Q. RETURN CODE < ZERO?
         BM    HØØØ3                     . A. YES - ERROR
         BZ    HØØØ2                     . A. ZERO
         CH    R15,=H'1'                 . Q. RETURN CODE ONE?
         BNE   HØØØ3                     . A. NO - ERROR
*
HØØØ2    EQU   *
*
         LA    R15,24                    . SET ERROR CODE
         CLC   #DATALEN,SHVVALL+2        . Q. CHANGE IN VARIABLE LENGTH?
         BNE   HØØØ3                     . A. YES - ERROR
```



```
         BCT   R3,HØØØ1                  . LOOP
*                                        . MOVE THE LAST ONE
         L     R1,SHVVALA                . 1 -> VALUE
         EX    R5,MVCDATA                . MOVE DATA INTO ARRAY
         SLR   R15,R15                   . ZEROIZE 15
         B     HØØØ4
*
HØØØ3    EQU   *
*
         ST    R15,RETCODE               . SAVE RETURN VALUE
*
HØØØ4    EQU   *
*
         L     R1Ø,HSAVE                 . RESTORE RETURN ADDRESS
         BR    R1Ø
         DROP  R2
*
MVCDATA  MVC   *-*(*-*,R4),*-*(R1)       . EXECUTED MOVE
         TITLE 'SET UP SORT CODE AND PERFORM SORT'
*        FOR THE SAKE OF REENTRANCY
*            GET STORAGE FOR SORT CODE
*            MODIFY CODE IN STORAGE
*        DO THE SORT
*        RELEASE STORAGE
*
*        REGISTER USAGE
*        Ø                               . -> ARRAY TO BE SORTED
*        1                               . WORK
*        4                               . BINARY HALVING CONTROL
*        5                               . WORK
*        6                               . BINARY HALVING CONTROL
*        9                               . ONE
*        11                              . -> STORAGE FOR CODE
*
SORTDATA EQU   *
*
         ST    R1Ø,ISAVE                 . SAVE RETURN ADDRESS
*
         STORAGE OBTAIN,                 . GET STORAGE FOR CODE          *
               ADDR=(R11),                                               *
               LENGTH=CODELEN,                                           *
               SP=Ø
         ST    R11,@CODE                 . SAVE ADDRESS
         LA    R5,CODELEN-1              . LENGTH OF SORT CODE
         EX    R5,MVCCODE                . MOVE THE CODE
*
         L     RØ,@ARRAY                 . Ø -> ARRAY
         LH    R5,#DATALEN               . 5 - LENGTH OF ARRAY ELEMENT
         SR    RØ,R5                     . Ø -> ARRAY ELEMENT #Ø
         BCTR  R5,RØ                     . DECREMENT BY ONE
```



```
         LA    R1,XCDISP                . 1 - DISPLACEMENT OF XC
*                                       .     INSTRUCTIONS IN SORT CODE
         STC   R5,1(R1,R11)             . ZAP LENGTH DATA WITHIN
         STC   R5,7(R1,R11)             .     EXCLUSIVE OR CHARACTER
         STC   R5,13(R1,R11)            .     INSTRUCTIONS
*
         LH    R5,KEYLEN                . 5 - KEY LENGTH
         BCTR  R5,RØ                    . DECREMENT BY ONE
         LA    R1,CLCDISP               . 1 - DISPLACEMENT OF CLC
*                                       .     INSTRUCTION IN SORT CODE
         STC   R5,1(R1,R11)             . ZAP COMPARE INSTRUCTION
*                                       .     LENGTH
         LH    R5,KEYDISP               . 5 - KEY DISPLACEMENT
         STC   R5,3(R1,R11)             . ZAP COMPARE INSTRUCTION
         STC   R5,5(R1,R11)             .     DISPLACEMENTS
*
         CLI   SEQ,C'A'                 . Q. ASCENDING SEQUENCE?
         BE    IØØØ1                    . A. YES
         MVI   1+BNLDISP(R11),X'DØ'     . ZAP BNL TO BNH
*
IØØØ1    EQU   *
*
         L     R5,#STEMVAR              . 5 - NUMBER OF ELEMENTS
*
         LA    R9,1                     . 9 - ONE
         LR    R4,R9                    . 4 - ONE
         BXLE  R4,R4,*                  . VALUE OF 2**N > # OF ELEMENTS
         LR    R6,R4                    . 6 - PARTITION SIZE
         BCTR  R6,RØ                    . DECREMENT BY ONE
         BASR  R1Ø,R11                  . PERFORM THE SORT
*
         STORAGE RELEASE,               . RELEASE SORT CODE            *
               ADDR=(R11),                                             *
               LENGTH=CODELEN
*
         L     R1Ø,ISAVE                . RESTORE RETURN ADDRESS
         BR    R1Ø
MVCCODE  MVC   *-*(*-*,R11),SORTCODE     . EXECUTED MOVE
         TITLE 'RELOAD STEM DATA FROM SORTED ARRAY'
*        HAVING SORTED THE DATA IN AN ARRAY
*        MOVE ARRAY DATA BACK INTO STEM VARIABLES
*
*        REGISTER USAGE
*        2                              . -> SHARED VARIABLE BLOCK
*
RELOADST EQU   *
*
         ST    R1Ø,JSAVE                . SAVE RETURN ADDRESS
*
         LA    R2,SHVARBLK              . 2 -> SHARED VARIABLE BLOCK
```

```
         USING SHVBLOCK,R2              . DSECT ADDRESSABILITY
         XC    Ø(L'SHVARBLK,R2),Ø(R2)   . INITIALIZE IT
         MVI   SHVCODE,SHVSYSET         . SPECIFY ACTION
         LA    R1,NEWSTEM               . 1 -> NEW STEM NAME
         ST    R1,SHVNAMA               . SAVE IN DSECT
         LH    R1,#DATALEN              . 1 - LENGTH OF DATA
         ST    R1,SHVVALL               . SAVE IN DSECT
*
         L     R3,#STEMVAR              . 3 - NUMBER OF STEM VARIABLES
         ZAP   #VARS,=P'+1'             . VARIABLE COUNTER
         L     R4,@ARRAY                . 4 -> ARRAY
         LH    R5,#DATALEN              . ELEMENT LENGTH
*
JØØØ1    EQU   *
*
         ST    R4,SHVVALA               . -> VALUE IN DSECT
         BAS   R1Ø,BLDVARNM             . BUILD VARIABLE NAME
         MVC   SHVNAML,#NEWSTEM         . LENGTH OF VARIABLE NAME
         L     RØ,@REXX                 . Ø -> REXX ENVIRONMENT BLOCK
         LA    R1,PIRXEXCOM             . 1 -> PARAMETER LIST
         L     R15,@IRXEXCOM            . 15 -> EPA IRXEXCOM
         BASSM R14,R15                  . GO FOR IT
         LTR   R15,R15                  . Q. RETURN CODE < ZERO?
         BM    JØØØ3                    . A. YES
         BZ    JØØØ2                    . A. ZERO
         CH    R15,=H'1'                . Q. RETURN CODE ONE?
         BNE   JØØØ3                    . A. NO
*
JØØØ2    EQU   *
*
         LA    R4,Ø(R5,R4)              . 4 -> NEXT ARRAY ENTRY
         AP    #VARS,=P'+1'             . INCREMENT VARIABLE NUMBER
         BCT   R3,JØØØ1                 . LOOP
         B     JØØØ4                    . WE ARE DONE
*
JØØØ3    EQU   *
*
         ST    R15,RETCODE              . SAVE RETURN CODE
*
JØØØ4    EQU   *
*
         L     R1Ø,JSAVE                . RESTORE RETURN ADDRESS
         BR    R1Ø
         DROP  R2
         TITLE 'DEVELOP STEM NAME'
*        CREATE STEM NAME FOR VARIABLE
*        TAKE SPECIFIED STEM AND APPEND THE OCCURRENCE NUMBER
*
*        REGISTER USAGE
*        1                              . -> INSTANCE NUMBER
```

```
*        6                              . LENGTH OF STEM
*                                       . -> NEW STEM (COMPOUND)
*        7                              . LENGTH OF NEW STEM
*        8                              . LENGTH OF INSTANCE NUMBER
*
BLDVARNM EQU   *
*
         MVC   STEMQUAL,MASK8           . MOVE EDIT MASK TO WORK AREA
         ED    STEMQUAL,#VARS           . EDIT THE DATA
         LA    R1,STEMQUAL              . 1 -> EDITED DATA
         LA    R8,L'STEMQUAL            . 8 - LENGTH OF EDITED DATA
*
KØØØ1    EQU   *
*
         CLI   Ø(R1),C' '               . Q. SIGNIFICANT?
         BNE   KØØØ2                    . A. YES
         LA    R1,1(,R1)                . 1 -> NEXT BYTE
         BCT   R8,KØØØ1                 . LOOP
*
KØØØ2    EQU   *
*
         MVC   NEWSTEM,SPACES           . INITIALIZE NEW STEM
         L     R6,#STEM                 . NUMBER OF BYTES IN STEM
         LR    R7,R6                    . 7 - SAME
         BCTR  R6,RØ                    . DECREMENT FOR EXECUTE
         EX    R6,MVCSTEM               . MOVE STEM INTO NEW STEM
         LA    R6,NEWSTEM               . 6 -> NEW STEM
         LA    R6,Ø(R7,R6)              . 6 -> AFTER STEM IN NEW STEM
*
KØØØ3    EQU   *
*
         MVC   Ø(1,R6),Ø(R1)            . MOVE COUNT BYTE BY BYTE
         LA    R1,1(,R1)                . 1 -> NEXT BYTE OF COUNT
         LA    R6,1(,R6)                . 6 -> NEXT BYTE OF NEW STEM
         LA    R7,1(,R7)                . INCREMENT LENGTH
         BCT   R8,KØØØ3                 . LOOP
         ST    R7,#NEWSTEM              . SAVE LENGTH
         BR    R1Ø
MVCSTEM  MVC   NEWSTEM(*-*),STEM
         TITLE '"FORM CODE" PERFORMING THE ACTUAL SORT'
SORTCODE EQU   *                        . ACTUAL SORT PROCESS
*                                       . SPLIT TABLE INTO PARTITIONS
*                                       .   BY CONTINUOUS HALVING
*
         USING *,R11                    . ESTABLISH ADDRESSABILITY
*
SORT1    EQU   *
*
         SRA   R6,1                     . HALF THE CURRENT PARTITION
         BZR   R1Ø                      . IF ZERO - EXIT SORT
```

```
         LR    R8,R5                      . 8 - NUMBER OF ELEMENTS
         SR    R8,R6                      . SUBTRACT PARTITION SIZE
         LR    R7,R9                      . 7 - ONE
*
SORT2    EQU   *
*
         CR    R7,R8                      . Q. WITHIN PARTITION?
         BH    SORT1                      . A. OUTSIDE - RESET & RESTART
         LR    R4,R7                      . 4 = 7 (POINTER)
*
SORT3    EQU   *
*        SPACE 1
         CR    R4,R9                      . Q. POINTER < ONE?
         BL    SORT4                      . A. YES - INCREMENT POINTER
         LA    R1,Ø(R4,R6)                . GET RELATIVE POSITION OF PTR
         MH    R1,#DATALEN                . ASSOCIATE WITH ELEMENT
         AR    R1,RØ                      . 1 -> CURRENT ELEMENT
         LR    R2,R4                      . GET POSITION OF COMPARAND
         MH    R2,#DATALEN                . ASSOCIATE WITH ELEMENT
         AR    R2,RØ                      . 2 -> COMPARE ELEMENT
*
CLCDISP  EQU   *-SORTCODE                 . THE FOLLOWING COMPARE IS
*                                         .    MODIFIED IN THE PROLOG CODE
*
         CLC   Ø(Ø,R1),Ø(R2)              . Q. ELEMENTS IN SEQUENCE?
*
BNLDISP  EQU   *-SORTCODE                 . THE FOLLOWING BRANCH IS
*                                         .    MODIFIED IN THE PROLOG CODE
*
         BNL   SORT4                      . A. YES - DO NOT SWAP
*
XCDISP   EQU   *-SORTCODE                 . THE FOLLOWING EXCLUSIVE OR
*                                         .    INSTRUCTIONS ARE MODIFIED
*                                         .    IN THE PROLOG CODE
*
         XC    Ø(Ø,R1),Ø(R2)              . INTERCHANGE ELEMENTS - MUST
         XC    Ø(Ø,R2),Ø(R1)              .    NOT BE EQUAL
         XC    Ø(Ø,R1),Ø(R2)
*
         SR    R4,R6                      . MODIFY PARTITION LOW BOUND
         B     SORT3                      . CHECK REST OF PARTITION
*
SORT4    EQU   *
*
         AR    R7,R9                      . INCREMENT HIGH POINTER
         B     SORT2
*
CODELEN  EQU   *-SORTCODE                 . LENGTH OF "FORM CODE"
         DROP  R11
         DROP  R13
```



```
         TITLE 'DYNAMIC AREA'
DYNAREA  DSECT
         DS    18F
DWORD    DS    D                        . FOR CVD
DSAVE    DS    F                        . REGISTER SAVE AREA
FSAVE    DS    F                        . REGISTER SAVE AREA
GSAVE    DS    F                        . REGISTER SAVE AREA
HSAVE    DS    F                        . REGISTER SAVE AREA
ISAVE    DS    F                        . REGISTER SAVE AREA
JSAVE    DS    F                        . REGISTER SAVE AREA
@ARGTAB  DS    F                        . -> ARGUMENT TABLE
@ARRAY   DS    F                        . -> ARRAY
@CODE    DS    F                        . -> SORT CODE
@EFPL    DS    F                        . -> REXX EXT FUNCTION PLIST
@EPAREA  DS    F                        . -> EXTERNAL PARAMETER AREA
@EVALBLK DS    F                        . -> EVAL BLOCK
@IRXEXCOM DS   F                        . -> ENTRY POINT IRXEXCOM
@REXX    DS    F                        . -> REXX ENVIRONMENT BLOCK
#ARRAY   DS    F                        . LENTGH OF ARRAY
#NEWSTEM DS    F                        . LENGTH OF NEW STEM NAME
#RCDATA  DS    F                        . LENGTH OF RETURNED DATA
#STEM    DS    F                        . LENGTH OF STEM VARIABLE NAME
#STEMVAR DS    F                        . NUMBER OF STEM VARIABLES
RETCODE  DS    F                        . RETURN CODE
*
PIRXEXCOM DS   ØF                       . IRXEXCOM PARAMETER LIST
@CSTR    DS    F                        . -> CHARACTER STRING IRXEXCOM
@DUMMY1  DS    F                        . -> DUMMY ARGUMENT
@DUMMY2  DS    F                        . -> DUMMY ARGUMENT
@SHVB    DS    F                        . -> FIRST SHARED VARIABLE BLOCK
*
#DATALEN DS    H                        . DATA LENGTH
KEYLEN   DS    H                        . KEY LENGTH
KEYDISP  DS    H                        . KEY DISPLACEMENT
SEQ      DS    C                        . SORT SEQUENCE
TEST     DS    CL3                      . NUMERIC TEST
*
NEWSTEM  DS    CL44                     . NEW STEM NAME
STEM     DS    CL32                     . STEM NAME ARGUMENT VALUE
BUFFER   DS    CL256                    . BUFFER
#VARS    DS    PL4                      . NUMBER OF INSTANCES OF STEM
RCDATA   DS    CL8                      . RETURN DATA
STEMQUAL DS    CL8                      . STEM QUALIFIER WORK
VARWORK  DS    CL8                      . VARIABLE NUMBER WORK
         DS    ØF
SHVARBLK DS    CL(SHVBLEN)              . SHARED VARIABLE BLOCK AREA
         DS    ØF
EPAREA   DS    CL28                     . EXTERNAL PARAMETER AREA
DYNLEN   EQU   *-DYNAREA
         TITLE 'IBM SUPPLIED DSECTS'
```



```
         IRXARGTB                         . ARGUMENT TABLE
         IRXEFPL                          . EXTERNAL FUNCTION PARAM LIST
         IRXEVALB                         . EVALUATION BLOCK
         IRXSHVB                          . SHARED VARIABLE REQUEST BLOCK
         TITLE 'LIST FORM MACROS, CONSTANTS'
REXSORT  CSECT
ASCEND   DC    C'ASCENDING',X'FE'       . SORT SEQUENCE
DESCEND  DC    C'DESCENDING'
*
MASK8    DC    X'4020202020202120'      . EDIT MASK
SPACES   DC    32C' '                   . SPACES FOR INITIALIZATION
ZEROS    DC    C'000'                   . NUMERIC TEST
CIRXEXCOM DC   C'IRXEXCOM'              . NAME OF REXX SERVICE ROUTINE
TRTABLE  DC    256X'00'                 . TRANSLATE TABLE
         ORG   TRTABLE+X'4B'            . VALIDATE CONTENT OF STEM NAME
         DC    X'4B'
         ORG   TRTABLE+X'5B'
         DC    X'5B'
         ORG   TRTABLE+X'6D'
         DC    X'6D'
         ORG   TRTABLE+X'7B'
         DC    X'7B7C'
         ORG   TRTABLE+X'81'
         DC    X'C1C2C3C4C5C6C7C8C9'
         ORG   TRTABLE+X'91'
         DC    X'D1D2D3D4D5D6D7D8D9'
         ORG   TRTABLE+X'A2'
         DC    X'E2E3E4E5E6E7E8E9'
         ORG   TRTABLE+X'C1'
         DC    X'C1C2C3C4C5C6C7C8C9'
         ORG   TRTABLE+X'D1'
         DC    X'D1D2D3D4D5D6D7D8D9'
         ORG   TRTABLE+X'E2'
         DC    X'E2E3E4E5E6E7E8E9'
         ORG   TRTABLE+X'F0'
         DC    X'F0F1F2F3F4F5F6F7F8F9'
         ORG
         LTORG
         END   REXSORT
```

Dave Loveluck
System Programmer (USA)

DYNAM/NODYNAM

INTRODUCTION

DYNAM/NODYNAM is a compiler option that determines when a program's subroutines are link edited.

If NODYNAM is specified, all statically-called subroutines (CALL 'PROGNAME' USING X, Y, Z) to a program will be link edited into the executable object program during the link-edit phase that follows actual compilation.

If DYNAM is specified, the program's subroutines will be linked on-the-fly when the program is executed. Dynamically-called subroutines (CALL SUB-PROG USING X, Y, Z) will always be linked dynamically.

THE ADVANTAGES

NODYNAM freezes your configuration at the time you link edit the program. You always know which version of a subroutine you are using. It also loads faster at run time, since everything is in one package. If a statically-called subroutine is changed, your shop will have to find every program that calls it and re-link those programs.

DYNAM takes longer to load, since it has to search the executable libraries for each subroutine. However, with this method, you always get the latest version of each subroutine. There is no necessity to re-link main programs when a statically-called subroutine is changed. The only requirement is to replace the old version of the subroutine in its executable library.

Alan Kalar
System Programmer (USA)

MVS news

William Data Systems has announced FTPalert Version 1.1, an OS/390 application that interfaces to FTP and enables reporting of all FTP activity, showing both successful and failed file transfers, the users' ID and IP addresses, and the transfer rates achieved. This information can be passed to an operations system for further action to be taken, such as the submission of a job to process the newly-arrived files or to advise support staff that a file transfer has failed.

The reporting and control provided is further extended to the security of FTP. Interactive users of FTP are only compared with their TSO signon to validate their access rights to FTP. This means that all users of FTP must have access to TSO but that gives all Open Edition TSO users access to all FTP facilities.

With FTPalert, all FTP activities can be defined to RACF and other security systems as secure resources, making FTP as secure as all other mainframe services. FTPalert comes with a 3270 application that enables users to manage activity and search for user IDs, file names, and transfer failures etc.

William Data Systems, Suite 290, 2034 Eisenhower Avenue, Alexandria, Virginia 22314-4678, USA.
Tel: (703) 299 0008
Fax: (703) 299 9776

William Data Systems, Arch House, 5 High Street, Old Oxted, Surrey, RH8 9LN, UK
Tel: (01883) 723 999
Fax: (01883) 723 888
http:// www.willdata.com

* * *

Blockade Systems has announced a partnership with enCommerce, a provider of software and services for managing secure access to e-business portals.

The Blockade/enCommerce partnership means Blockade's OS/390 security products for authentication, authorization, and auditing will be integrated with the enCommerce getAccess portal management software through a specialized getAccess pluggable authentication and authorization module (PAAM). enCommerce customers that have an investment in OS/390 security – those with RACF, ACF2, or Top Secret – can use PAAM to exploit their existing IT infrastructure whilst extending software applications to the Internet and on-line e-commerce transactions.

Blockade Systems Inc, 7500 Brooktree Drive, Suite 204, Wexford, PA, 15090, USA.
Tel: (412) 577 2487
Fax: (724) 327 5825
http://www.blockade.com

* * *

Xephon will be holding its annual *MVS 2000* conference at the Mountbatten Hotel in London, 7-8 June 2000. *MVS 2000* is designed specifically for technical managers, systems programmers, strategic planners, and other system specialists at MVS/ESA and OS/390 installations.

The attendance fee for *MVS Update* subscribers is £570.00 plus £66.50 VAT. For further information, please telephone the registrar, Angela Scott, on (01635) 33823.

* * *