168 MVS update

September 2000

In this issue

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,
Westminster, CO 80035-0100
USA
Telephone: (303) 410 9344
Fax: (303) 438 0290

Contributions
Articles published in *MVS Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*, or you can download a copy from www.xephon.com/contnote.html.

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.

Using the RMF Spreadsheet Reporter

INTRODUCTION

In my article 'Internet Resources for Systems Programmers' in *MVS Update* Issue 165, June 2000, page 6, I mentioned the tools available from the RMF group, including the RMF Spreadsheet Reporter. These tools are free to download from IBM (http://www.s390.ibm.com/rmf/rmfhtmls/rmftools.htm).

The RMF Spreadsheet Reporter is available for the Microsoft Windows operating system (including NT, 98, and 95) and the Lotus 1-2-3 and Microsoft Excel spreadsheets. The Spreadsheet Reporter provides a front-end to the RMF postprocessor to take advantage of the graphical features of Windows-based spreadsheet programs to present RMF performance data.

USING THE RMF SPREADSHEET REPORTER

Getting up and running with the RMF Spreadsheet Reporter is relatively straight forward. The downloadable module called rmfppv4.exe is currently (August 2000) at Release 4.7.2, and is about 8.5 MB.

Rmfppv4.exe is a self-extracting archive, that installs into directory C:\Rmfpp by default. There is very little for the user to do during the install dialog, but it is necessary to specify the custom installation because you are then required to specify which spreadsheet program you will be using. There are two types of Microsoft Excel macros, one type for pre-Excel 97 and the other for Excel 97 and later, so Excel users must choose the correct set of macros.

The IBM documention states that all that is necessary to use the product is a TCP/IP connection between your Windows workstation and the OS/390 host mainframe, but, as is explained below, even this is not really necessary to enable you to use the spreadsheets. Starting the RMF Spreadsheet Reporter opens a window on your workstation with eight options in the form of clickable icons.

The first is the Collector, and this is the only part of the product which actually requires a TCP/IP connection to the host. The Collector extracts data from your OS/390 host's RMF history and brings it to the Windows workstation for processing. This process consists of customizing a deck of OS/390 JCL (which is stored locally on the workstation in C:\Rmfpp\Progs\rmfpp.jcl), submitting this JCL to run on the host via the TCP/IP connection, and then FTPing the files created by the job back to the workstation in directory C:\Rmfpp\Listing.

The JCL deck is an RMF post processor job, and it creates files in normal report format, which are stored in temporary disk files and then FTPed back to the workstation.

The Collector can be replaced by a manual process if you do not have the necessary TCP/IP connection. Simply take the JCL deck from C:\Rmfpp\Progs\rmfpp.jcl and copy it to a library on the host. Considerable cosmetic modification necessary, but the essential logic of the job is clear and does not need changing at all. Note that the MFPINPUT DD statement in the ERBRMFPP step is not present; it is intended to be added by the Collector immediately prior to submitting the job, so this has to be added manually.

Submitting the job will create the necessary sequential files containing the RMF post processor reports. These files can then be copied back to the directory C:\Rmfpp\Listing with any file transfer utility available, such as IND$FILE over an SNA link to a 3270 emulation package. The names of the files do not seem to matter, but IBM uses the format Thhmmss.ddd where hhmmss and ddd are the time and Julian day that the file was created. If you do use the Collector, you have to add a profile for each host where you must specify its IP address, a user-id and password, and some accounting information, which is used on the JOB card of the Collector job.

Then you have to tell the Collector where it can find the SMF records that the RMF is generating, either in SMF datasets or SMF buffers, the type of RMF post processor reports to run, and the intervals to be covered. Overview report parameters, if required, must be specified seperately. The first time I ran the Collector it failed, producing the following message:

```
*** Error *** FTPGET failed!
```

It also recommended looking in C:\Rmfpp\Progs\ftperr.log, where I found the following:

```
550 Data set userid.D215.T151232.REPORT not found
```

This did not inspire confidence, particularly as at this point I had no real grasp of the JCL submission process that I have described above. I could see that the file userid.D215.T151232.REPORT indeed did not exist on my system, but had no clue as to why it *should* exist.

It turned out that the Collector had called the job it submitted userid$ and this job had failed with a JCL error. The error was caused by lines in the step ALLOC with the format

```
<file://PPSØ1>//PPSØ1 DD
DISP=(NEW,CATLG),DSN=userid.SUMMØ1.RMFDATA,REFDD=*.MSG
```

This generates the message:

```
UNIT FIELD SPECIFIES INCORRECT DEVICE NAME
```

The referred DDname was quite correct and specified UNIT=SYSDA, so I overcame the problem by editing C:\Rmfpp\Progs\rmfpp.jcl, which is just a standard text file, with Notepad and adding UNIT=SYSDA to all the PPSnn DD statements in step ALLOC. Rerunning the Collector with these modifications resulted in a clean job and a file was downloaded to the PC using FTP. This file can get to be fairly sizeable. A run on 24 hours of RMF data with a (default) 30 minute interval and requesting the Collector to run all available reports resulted in a file of 83 tracks of 3390 DASD, or about 4 MB of data. While this represents no problem for a high-speed local or broadband remote connection, anyone using dial-up access might need to keep the Collector requests to a minimum.

The second icon represents the Extractor. This is used to extract reports from the downloaded Collector data and to create a Report-Work-Set, which is the basic structure used by the Spreadsheet Reporter to keep sets of reports together.

The third icon is for the Converter, which is used to convert the reports in a Report-Work-Set from RMF report format into spreadsheet format, .wk1 for Lotus 1-2-3 or .xls for MS-Excel. If Overview reports are required, then the Converter function of icon 6, RecConvert does the same type of conversion for the Overview data. At this point the preparation is complete, you are ready to actually fire up your spreadsheet program and run the supplied macros.

Clicking the fourth icon, Spreadsheet, opens a folder which has the supplied macro RMFR9MN and an embedded folder with all the other macros. In general it is recommended to click on RMFR9MN, which launches your spreadsheet program and runs the macro inside it. RMFR9MN is the Main Dialog and all the other macros are accessible from within it.

From the Main Dialog you select which report type you wish to process, and typically a new dialog appears where you specify which Report-Work-Set contains the data that you are interested in examining. To go into detail about all the reports and graphs that are available is beyond the scope of this article, but as an example in RMFR9DAS, the DASD Activity Report, there is:

- System sheet – (this is in Excel 97) which shows installed DASD capacity, overall connect/disconnect/service/IOSQ/pending/ response time, I/O/Service time/Path intensities, and skews.
- Summary sheet – with the data for the top five LCUs and top 10 devices sorted by I/O intensity.
- LCUACT graph – which graphs LCU activity rate, I/O intensity, Service time and Path intensity.
- LCURT graph – which graphs IOSQ/Pend/Disc/Conn and Activity rate.
- TOP10ACT graph – like LCUACT but for the top 10 devices.
- TOP10RT graph – like LCURT but for the top 10 devices.

CONCLUSIONS

All these sheets and graphs are populated automatically by the system. RMF Spreadsheet Reporter really is a very slick tool for quickly and easily producing the kind of reports and graphs that I, and I have no doubt very many other people who look at RMF reports, have often laboured long and hard over, using a spreadsheet and RMF data.

Patrick Mullen
Systems Programmer (Canada)

A REXX to display the HFS directory structure

INTRODUCTION

The following REXX program uses OS/390 Unix callable services to display the directory structure of your mounted HFS datasets. These 'syscall' commands are documented in the IBM manual *Using Rexx and OS/390 Unix Systems Services* (SC28-1905-04).

The REXX first uses the 'getmntent' service to obtain information about mounted HFS datasets and then navigates through the directory structure using the 'readdir' service to obtain a directory listing. The 'lstat' service is then used to determine which of the resulting files is a directory. A recursive call is used to process successive directories; this is controlled via a test for 'path depth'.

REXX EXEC

```
/* REXX */
x = syscalls('ON')
address syscall
'getmntent mount_info.'
count = 0
do i = 1 to mount_info.0
   if mount_info.mnte_fstype.i = 'HFS' then
   do
      if mount_info.mnte_path.i = '/' then
         do
            root_hfs = mount_info.mnte_fsname.i
            iterate
         end
      count = count + 1
      hfs.count = mount_info.mnte_path.i','mount_info.mnte_fsname.i
   end
end

hfs.0 = count
say 'Root (Hfs = 'strip(root_hfs)')'
x = process_dir('/')
exit
count_depth : procedure

   arg path
   k = 0
   do until path = ''
      k = k + 1
```

```
      parse var path '/' path
   end

return (k)
process_dir : procedure expose hfs.

   parse arg path
   x = syscalls('ON')
   path_depth = count_depth(path)
   if path_depth > 4 then           /* controls level of recursion */
      return (dont_care)
   'readdir (path) dir_info.'
   count = 0

   do i = 1 to dir_info.0
      if path = '/' then
         file_name = path || dir_info.i
      else
         file_name = path || '/' || dir_info.i
      'lstat (file_name) stat_info.'
      if stat_info.st_type ¬= S_ISDIR then
         iterate
      if substr(dir_info.i,1,1) = '.' then
         iterate
      count = count + 1
      directory.count = file_name
   end

   directory.0 = count
   drop dir_info.
   do j = 1 to directory.0
      indentation = copies(' ',path_depth*3)
      say indentation || directory.j || lookup(directory.j)
      if directory.j = '/SERVICE' then      /* skip this one */
         iterate j
      x = process_dir(directory.j)
end

return (dont_care)
lookup : procedure expose hfs.
   parse arg path
   hfs_name = ''
   do i = 1 to hfs.0
      parse var hfs.i mount_point ',' name
      if path = mount_point then
         do
            hfs_name = ' (Hfs = 'strip(name)')'
            leave
         end
   end
return (hfs_name)
```

SAMPLE OUTPUT

```
Root (Hfs = SYS5.OMVS.ROOT)
   /archive
      /archive/etc
   /bin
      /bin/IBM
      /bin/X11
   /dev
   /etc (Hfs = SYS5.OMVS.ETC)
      /etc/booksrv
      /etc/bpa
      /etc/cmx
      /etc/dce
         /etc/dce/dcecp
```


CA-1 tape catalog utility

The administration and care of the tape management environment is a standard task in most data centres. CA-1 is one of the most widely used tape management products. We have developed a simple routine to augment other reporting tools such as TMSGRW and CA-EARL. This program should be very easy to adapt to your needs, because it provides a foundation from which to add more report modules. The program can produce up to six different reports. Any combination can be selected. The JCL to invoke the program is shown below:

```
//jobname  JOB your job card
//STEP0001 EXEC PGM=CA1REPRT
//STEPLIB  DD DISP=SHR,DSN=your step library if needed
//CA1TMC   DD DISP=SHR,DSN=name of your TMC
//MESSAGES DD SYSOUT=*
//CONTROL  DD *
DSNB ALL
DSNB ACTIVE
VOLUME ALL
VOLUME ACTIVE
VOLUME ERROR
VOLUME DSN=DFHSM
/*
//
```

The reports that are available can be requested by using the following key words as input to the control dataset. They are as follows:

- DSNB ALL – request all DSNB entries from the TMC.
- DSNB ACTIVE – request active DSNB entries from the TMC.
- VOLUME ALL – request all tape volume records from the TMC.
- VOLUME ACTIVE – request only active tape volume records from the TMC.
- VOLUME ERROR – request tape volumes that have exceeded error thresholds.
- 'VOLUME DSN=' – request only tape volumes matching a dataset name mask.

Each of the requested reports will be written to a unique report dataset. Each of the report datasets is allocated dynamically through the SVC 99 facilities of the operating system.

All of the macros that were used to produce this routine have been included. The $TMSRLO and $DSNBRLO macros provide DSECTs that are used to map out the DSNB and volume records from the TMC. At the time the macros were developed, support for 3590 tape devices was not documented in CA-1 documentation, but was included where known. All of the datasets are processed in 31-bit mode. This program has been used in a MVS 5.2.2, DFSMS 1.3 environment, as well as in OS/390 Version 2 Release 8 and DFSMS 1.5 environments. The TMC itself was at the 5.2 level.

CA1REPRT

```
         TITLE 'CA1REPRT - CA1 TAPE CATALOG REPORTING SERVICES'
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* CSECT   : CA1REPRT                                                  *
* MODULE  : CA1REPRT                                                  *
* AUTHOR  : ENTERPRISE DATA TECHNOLOGIES                              *
* DESC    : CA1REPRT IS A SIMPLE UTILITY PROGRAM DESIGNED TO READ THE *
*           CA1 TMC AND PRODUCE SEVERAL SIMPLE OUTPUT REPORTS.        *
* MACROS  : $ESAPRO $ESAEPI $ESASTG OPEN CLOSE DCB DCBD DCBE          *
*           GET PUT WTO                                               *
* DSECTS  : IHADCBD IEFZB4DØ IEFZB4D2 $TMSRLO $DSNBRLO                *
* INPUT   : CA1TMC   - CA1 TAPE MANAGEMENT CATALOG                    *
```

```
*           CONTROL  - CONTROL STATEMENTS                              *
* OUTPUT  : RPTØ1    - DYNAMICALLY ALLOCATED REPORT FILE               *
*           RPTØ2    - DYNAMICALLY ALLOCATED REPORT FILE               *
*           RPTØ3    - DYNAMICALLY ALLOCATED REPORT FILE               *
*           RPTØ4    - DYNAMICALLY ALLOCATED REPORT FILE               *
*           RPTØ5    - DYNAMICALLY ALLOCATED REPORT FILE               *
*           RPTØ6    - DYNAMICALLY ALLOCATED REPORT FILE               *
*           MESSAGES - OUTPUT FILE FOR ERRORS AND INFORMATIONAL DATA   *
* PLIST   : NONE                                                       *
* CALLS   : NONE                                                       *
* NOTES   : 31 BIT ADDRESSING USED FOR ALL FILES.                      *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
CA1REPRT $ESAPRO R11,R12,RM=24,AM=31
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* SET ALL OF THE REPORT LINE COUNTERS TO THE MAXIMUM VALUE.            *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         MVC   CNT_Ø1D,RPT_MLIN        SET COUNTER TO MAX
         MVC   CNT_Ø2D,RPT_MLIN        SET COUNTER TO MAX
         MVC   CNT_Ø3D,RPT_MLIN        SET COUNTER TO MAX
         MVC   CNT_Ø4D,RPT_MLIN        SET COUNTER TO MAX
         MVC   CNT_Ø5D,RPT_MLIN        SET COUNTER TO MAX
         MVC   CNT_Ø6D,RPT_MLIN        SET COUNTER TO MAX
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* SET UP THE TRANSLATE TABLE.                                          *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         LA    R14,TRAN_TAB            GET @(TRANSLATE TABLE)
         MVI   C' '(R14),C' '          ENTER THE DELIMITER
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* OPEN UP THE MESSAGES FILE.                                           *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         OPEN  (MSG,(OUTPUT)),MODE=31
         SPACE 1
         USING IHADCB,R1               DECLARE A BASE
         LA    R1,MSG                  GET @(DCB WE JUST OPENED)
         TM    DCBOFLGS,DCBOFOPN       Q. OPEN CLEAN?
         BO    MSG_OPEN                A. YES, PROCEED
         DROP  R1
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* SYNAD CONTROL POINT FOR PHYSICAL ERROR ON THE MESSAGES DATASET.      *
* ISSUE A WTO TO USER, SET A RETURN CODE AND EXIT BACK TO OP. SYS.     *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
SYN_MSG  DS    ØH
```

```
         SPACE 1
         LA    R1,WTO_MSG              POINT TO THE WTO
         WTO   TEXT=ER_MSGØ1,                                          +
               ROUTCDE=(2,1Ø),                                         +
               DESC=(6),                                               +
               MF=(E,(1))
         SPACE 1
         MVC   RET_CODE,RCØØ1Ø         SET THE RETURN CODE
         B     MSG_CLO                 EXIT PROGRAM
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* MESSAGES DATASET IS OPEN. SEE IF WE CAN OPEN UP THE CONTROL DATASET *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
MSG_OPEN DS    ØH
         SPACE 1
         MVI   MSG_FLAG,DCBOFOPN       INDICATE THE MESSAGES DATASET
         SPACE 1
         OPEN  (CON,(INPUT)),MODE=31
         USING IHADCB,R1               DECLARE A BASE
         LA    R1,CON                  GET @(DCB WE JUST OPENED)
         TM    DCBOFLGS,DCBOFOPN       Q. OPEN CLEAN?
         BO    CON_OPEN                A. YES, PROCEED
         DROP  R1
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* SYNAD CONTROL POINT FOR PHYSICAL ERROR ON THE CONTROL DATASET.      *
* ISSUE A MESSAGE, SET A RETURN CODE AND RETURN BACK TO THE OP. SYS.  *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
SYN_CON  DS    ØH
         SPACE 1
         MVI   MSG_BUFF,C' '           BLANK IN FIRST BYTE
         MVC   MSG_BUFD(MSG_BUFL-1),MSG_BUFF BLANK THE REMAINDER
         MVI   MSG_BUFC,C'1'           SET IN CARRIAGE CONTROL
         LH    R14,EL_MSGØ3            PICK UP MESSAGE LENGTH
         BCTR  R14,Ø                   DECREMENT IT DOWN BY 1
         LA    R1,ER_MSGØ3             PICK UP THE ADDRESS OF MESSAGE
         EX    R14,MSG_MOVE            MOVE IN THE MESSAGE
         PUT   MSG,MSG_BUFF
         MVC   RET_CODE,RCØØ1Ø         SET THE RETURN CODE
         B     EXIT_PGM                EXIT PROGRAM
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* THE CONTROL DATASET IS OPEN, SET THE FLAG.                          *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
CON_OPEN DS    ØH
         SPACE 1
         MVI   CON_FLAG,DCBOFOPN       INDICATE THE MESSAGES DATASET
         SPACE 1
         OPEN  (TMC,(INPUT)),MODE=31
```

```
         USING IHADCB,R1               DECLARE A BASE
         LA    R1,TMC                  GET @(DCB WE JUST OPENED)
         TM    DCBOFLGS,DCBOFOPN       Q. OPEN CLEAN?
         BO    TMC_OPEN                A. YES, PROCEED
         DROP  R1
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* SYNAD CONTROL POINT FOR PHYSICAL ERROR ON THE TMC.                   *
* ISSUE A MESSAGE, SET A RETURN CODE AND RETURN BACK TO THE OP. SYS.  *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
SYN_TMC  DS    ØH
         SPACE 1
         MVI   MSG_BUFF,C' '           BLANK IN FIRST BYTE
         MVC   MSG_BUFD(MSG_BUFL-1),MSG_BUFF BLANK THE REMAINDER
         MVI   MSG_BUFC,C'1'           SET IN CARRIAGE CONTROL
         LH    R14,EL_MSGØ2            PICK UP MESSAGE LENGTH
         BCTR  R14,Ø                   DECREMENT IT DOWN BY 1
         LA    R1,ER_MSGØ2             PICK UP THE ADDRESS OF MESSAGE
         EX    R14,MSG_MOVE            MOVE IN THE MESSAGE
         PUT   MSG,MSG_BUFF
         MVC   RET_CODE,RCØØ1Ø         SET THE RETURN CODE
         B     EXIT_PGM                EXIT PROGRAM
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* THE TMC IS OPEN.  SET OPEN FLAG.                                     *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
TMC_OPEN DS    ØH
         SPACE 1
         MVI   TMC_FLAG,DCBOFOPN       INDICATE THE MESSAGES DATASET
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* READ THE CONTROL CARDS AND DECIDE WHICH REPORTS WILL BE NEEDED.     *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
READ_CON DS    ØH
         SPACE 1
         GET   CON
         SPACE 1
         CLC   RPT_Ø1,Ø(R1)            Q. DO WE WANT REPORT Ø1
         BNE   NO_RPTØ1                A. NO, CHECK NEXT ENTRY
         MVI   RPT_Ø1F+1,X'FF'         TURN ON REPORT STATUS
         LA    R14,RPT_Ø1D             GET @(REPORT DCB)
         STCM  R14,B'1111',RPT_Ø1F+2   SAVE DCB FOR LATER
         B     READ_CON                GET THE NEXT RECORD
         SPACE 1
NO_RPTØ1 DS    ØH
         SPACE 1
         CLC   RPT_Ø2,Ø(R1)            Q. DO WE WANT REPORT Ø1
         BNE   NO_RPTØ2                A. NO, CHECK NEXT ENTRY
         MVI   RPT_Ø2F+1,X'FF'         TURN ON REPORT STATUS
```

```
         LA    R14,RPT_Ø2D             GET @(REPORT DCB)
         STCM  R14,B'1111',RPT_Ø2F+2   SAVE DCB FOR LATER
         B     READ_CON                GET THE NEXT RECORD
         SPACE 1
NO_RPTØ2 DS    ØH
         SPACE 1
         CLC   RPT_Ø3,Ø(R1)            Q. DO WE WANT REPORT Ø1
         BNE   NO_RPTØ3                A. NO, CHECK NEXT ENTRY
         MVI   RPT_Ø3F+1,X'FF'         TURN ON REPORT STATUS
         LA    R14,RPT_Ø3D             GET @(REPORT DCB)
         STCM  R14,B'1111',RPT_Ø3F+2   SAVE DCB FOR LATER
         B     READ_CON                GET THE NEXT RECORD
         SPACE 1
NO_RPTØ3 DS    ØH
         SPACE 1
         CLC   RPT_Ø4,Ø(R1)            Q. DO WE WANT REPORT Ø1
         BNE   NO_RPTØ4                A. NO, CHECK NEXT ENTRY
         MVI   RPT_Ø4F+1,X'FF'         TURN ON REPORT STATUS
         LA    R14,RPT_Ø4D             GET @(REPORT DCB)
         STCM  R14,B'1111',RPT_Ø4F+2   SAVE DCB FOR LATER
         B     READ_CON                GET THE NEXT RECORD
         SPACE 1
NO_RPTØ4 DS    ØH
         SPACE 1
         CLC   RPT_Ø5,Ø(R1)            Q. DO WE WANT REPORT Ø5
         BNE   NO_RPTØ5                A. NO, CHECK NEXT ENTRY
         MVI   RPT_Ø5F+1,X'FF'         TURN ON REPORT STATUS
         LA    R14,RPT_Ø5D             GET @(REPORT DCB)
         STCM  R14,B'1111',RPT_Ø5F+2   SAVE DCB FOR LATER
         B     READ_CON                GET THE NEXT RECORD
         SPACE 1
NO_RPTØ5 DS    ØH
         SPACE 1
         CLC   RPT_Ø6,Ø(R1)            Q. DO WE WANT REPORT Ø6
         BNE   NO_RPTØ6                A. NO, CHECK NEXT ENTRY
         MVI   RPT_Ø6F+1,X'FF'         TURN ON REPORT STATUS
         LA    R14,RPT_Ø6D             GET @(REPORT DCB)
         STCM  R14,B'1111',RPT_Ø6F+2   SAVE DCB FOR LATER
         LA    R3,L'RPT_Ø6(Ø,R1)       BUMP POINTER
         LA    R14,8Ø                  GET MAX LENGTH OF INPUT BUFFER
         LA    R15,L'RPT_Ø6            GET LENGTH OF DIRECTIVE
         SR    R14,R15                 ADJUST THE LENGTH
         EX    R14,TRAN_I              EXECUTE THE TRANSLATE
         BC    8,NO_RPTØ6              DIDN'T FIND WHAT WE WANT
         SR    R1,R3                   CALCULATE THE LENGTH
         BCTR  R1,Ø                    ADJUST IT
         STH   R1,DSN_MASL             SAVE THE LENGTH FOR LATER
         LR    R14,R1                  GET THE LENGTH
         EX    R14,MOV_MASK            SAVE THE DSN MASK
         B     READ_CON                GET THE NEXT RECORD
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
```

```
* IF WE GET HERE, IT WAS A BAD CONTROL CARD.  ISSUE ERROR MESSAGE.    *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
NO_RPTØ6 DS    ØH
         MVI   MSG_BUFF,C' '           BLANK IN FIRST BYTE
         MVC   MSG_BUFD(MSG_BUFL-1),MSG_BUFF BLANK THE REMAINDER
         LH    R14,EL_MSGØ4            PICK UP MESSAGE LENGTH
         BCTR  R14,Ø                   DECREMENT IT BY 1
         LA    R1,ER_MSGØ4             PICK UP THE ADDRESS OF MESSAGE
         EX    R14,MSG_MOVE            MOVE IN THE MESSAGE
         PUT   MSG,MSG_BUFF
         B     READ_CON                GO GET ANOTHER CONTROL CARD
         SPACE 1
EOF_CON  DS    ØH
         SPACE 1
         CLOSE (CON),MODE=31
         MVI   CON_FLAG,X'ØØ'          INDICATE FILE IS CLOSED
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* SET UP THE PARAMETER LIST THAT WE WILL USE FOR THE DYNAMIC          *
* ALLOCATION OF THE REPORT DATASETS.                                  *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         MVC   UN99_ØØØ(MODEL99L),MODEL99 MOVE IN THE SVC 99 TEXT
         LA    R14,UN99_ØØØ+DDNAM$$$   GET @(TEXT UNIT)
         ST    R14,TP99_ØØØ            SAVE IT IN THE POINTER LIST
         LA    R14,UN99_ØØØ+SYSOU$$$   GET @(TEXT UNIT)
         ST    R14,TP99_Ø04            SAVE IT IN THE POINTER LIST
         LA    R14,UN99_ØØØ+DSORG$$$   GET @(TEXT UNIT)
         ST    R14,TP99_Ø08            SAVE IT IN THE POINTER LIST
         LA    R14,UN99_ØØØ+LRECL$$$   GET @(TEXT UNIT)
         ST    R14,TP99_Ø12            SAVE IT IN THE POINTER LIST
         LA    R14,UN99_ØØØ+RECFM$$$   GET @(TEXT UNIT)
         ST    R14,TP99_Ø16            SAVE IT IN THE POINTER LIST
         LA    R14,UN99_ØØØ+CLOSE$$$   GET @(TEXT UNIT)
         ST    R14,TP99_Ø2Ø            SAVE IT IN THE POINTER LIST
         OI    TP99_Ø2Ø,X'8Ø'          HIGH ORDER BIT ON, LAST ENTRY
         LA    R14,TP99_ØØØ            GET @(FIRST TEXT UNIT)
         ST    R14,RB99_Ø12            SAVE IT IN THE REQUEST BLOCK
         LA    R14,RB99_Ø04            GET @(REQUEST BLOCK)
         ST    R14,RB99_ØØØ            SAVE IT
         OI    RB99_ØØØ,X'8Ø'          HIGH ORDER BIT TURNED ON
         MVI   RB99_Ø04,X'14'          PLACE THE LENGTH IN THE RB
         MVI   RB99_Ø04+1,S99VRBAL     INDICATE ALLOCATION REQUEST
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* SVC99 PLIST IS READY.  BASED ON REPORT REQUESTED FLAGS, WE WILL     *
* DYNAMICALLY ALLOCATE THE NEEDED REPORT FILES.                       *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         CLI   RPT_Ø1F+1,X'FF'         Q. REPORT Ø1 REQUESTED?
         BNE   NDY_Ø1F                 A. NO, CHECK FOR NEXT REPORT
```

```
         MVC   UN99_ØØØ+RPTXX$$$(5),REPORTØ1 MOVE IN THE DDNAME
         LA    R1,SVC_99RB             GET @(SVC 99 PLIST)
         SVC   99                      TRY THE ALLOCATE
         LTR   R15,R15                 Q. ALLOCATE SUCCESSFUL
         BZ    NDY_Ø1F                 A. YES, CHECK FOR NEXT REPORT
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* COME THROUGH HERE ONLY IF WE ENCOUNTER AN ERROR.                     *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         MVI   RPT_Ø1F+1,X'ØØ'         TURN OFF THE REPORT FLAG
         MVI   MSG_BUFF,C' '           BLANK IN FIRST BYTE
         MVC   MSG_BUFD(MSG_BUFL-1),MSG_BUFF BLANK THE REMAINDER
         LH    R14,EL_MSGØ6            PICK UP MESSAGE LENGTH
         BCTR  R14,Ø                   DECREMENT IT BY 1
         LA    R1,ER_MSGØ6             PICK UP THE ADDRESS OF MESSAGE
         EX    R14,MSG_MOVE            MOVE IN THE MESSAGE
         MVC   MSG_BUFF+(ER_MSG66-ER_MSGØ6)(5),REPORTØ1
         PUT   MSG,MSG_BUFF
         SPACE 1
NDY_Ø1F  DS    ØH
         SPACE 1
         MVC   UN99_ØØØ(MODEL99L),MODEL99 MOVE IN THE SVC 99 TEXT
         CLI   RPT_Ø2F+1,X'FF'         Q. REPORT Ø2 REQUESTED?
         BNE   NDY_Ø2F                 A. NO, CHECK FOR NEXT REPORT
         MVC   UN99_ØØØ+RPTXX$$$(5),REPORTØ2 MOVE IN THE DDNAME
         LA    R1,SVC_99RB             GET @(SVC 99 PLIST)
         SVC   99                      TRY THE ALLOCATE
         LTR   R15,R15                 Q. ALLOCATE SUCCESSFUL
         BZ    NDY_Ø2F                 A. YES, CHECK FOR NEXT REPORT
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* COME THROUGH HERE ONLY IF WE ENCOUNTER AN ERROR.                     *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         MVI   RPT_Ø2F+1,X'ØØ'         TURN OFF THE REPORT FLAG
         MVI   MSG_BUFF,C' '           BLANK IN FIRST BYTE
         MVC   MSG_BUFD(MSG_BUFL-1),MSG_BUFF BLANK THE REMAINDER
         LH    R14,EL_MSGØ6            PICK UP MESSAGE LENGTH
         BCTR  R14,Ø                   DECREMENT IT DOWN BY 1
         LA    R1,ER_MSGØ6             PICK UP THE ADDRESS OF MESSAGE
         EX    R14,MSG_MOVE            MOVE IN THE MESSAGE
         MVC   MSG_BUFF+(ER_MSG66-ER_MSGØ6)(5),REPORTØ2
         PUT   MSG,MSG_BUFF
         SPACE 1
NDY_Ø2F  DS    ØH
         SPACE 1
         MVC   UN99_ØØØ(MODEL99L),MODEL99 MOVE IN THE SVC 99 TEXT
         CLI   RPT_Ø3F+1,X'FF'         Q. REPORT Ø3 REQUESTED?
         BNE   NDY_Ø3F                 A. NO, CHECK FOR NEXT REPORT
         MVC   UN99_ØØØ+RPTXX$$$(5),REPORTØ3 MOVE IN THE DDNAME
         LA    R1,SVC_99RB             GET @(SVC 99 PLIST)
```

```
         SVC   99                      TRY THE ALLOCATE
         LTR   R15,R15                 Q. ALLOCATE SUCCESSFUL
         BZ    NDY_Ø3F                 A. YES, CHECK FOR NEXT REPORT
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* COME THROUGH HERE ONLY IF WE ENCOUNTER AN ERROR.                     *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         MVI   RPT_Ø3F+1,X'ØØ'         TURN OFF THE REPORT FLAG
         MVI   MSG_BUFF,C' '           BLANK IN FIRST BYTE
         MVC   MSG_BUFD(MSG_BUFL-1),MSG_BUFF BLANK THE REMAINDER
         LH    R14,EL_MSGØ6            PICK UP MESSAGE LENGTH
         BCTR  R14,Ø                   DECREMENT IT BY 1
         LA    R1,ER_MSGØ6             PICK UP THE ADDRESS OF MESSAGE
         EX    R14,MSG_MOVE            MOVE IN THE MESSAGE
         MVC   MSG_BUFF+(ER_MSG66-ER_MSGØ6)(5),REPORTØ3
         PUT   MSG,MSG_BUFF
         SPACE 1
NDY_Ø3F  DS    ØH
         SPACE 1
         MVC   UN99_ØØØ(MODEL99L),MODEL99 MOVE IN THE SVC 99 TEXT
         CLI   RPT_Ø4F+1,X'FF'         Q. REPORT Ø4 REQUESTED?
         BNE   NDY_Ø4F                 A. NO, CHECK FOR NEXT REPORT
         MVC   UN99_ØØØ+RPTXX$$$(5),REPORTØ4 MOVE IN THE DDNAME
         LA    R1,SVC_99RB             GET @(SVC 99 PLIST)
         SVC   99                      TRY THE ALLOCATE
         LTR   R15,R15                 Q. ALLOCATE SUCCESSFUL
         BZ    NDY_Ø4F                 A. YES, CHECK FOR NEXT REPORT
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* COME THROUGH HERE ONLY IF WE ENCOUNTER AN ERROR.                     *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         MVI   RPT_Ø4F+1,X'ØØ'         TURN OFF THE REPORT FLAG
         MVI   MSG_BUFF,C' '           BLANK IN FIRST BYTE
         MVC   MSG_BUFD(MSG_BUFL-1),MSG_BUFF BLANK THE REMAINDER
         LH    R14,EL_MSGØ6            PICK UP MESSAGE LENGTH
         BCTR  R14,Ø                   DECREMENT IT BY 1
         LA    R1,ER_MSGØ6             PICK UP THE ADDRESS OF MESSAGE
         EX    R14,MSG_MOVE            MOVE IN THE MESSAGE
         MVC   MSG_BUFF+(ER_MSG66-ER_MSGØ6)(5),REPORTØ4
         PUT   MSG,MSG_BUFF
         SPACE 1
NDY_Ø4F  DS    ØH
         SPACE 1
         MVC   UN99_ØØØ(MODEL99L),MODEL99 MOVE IN THE SVC 99 TEXT
         CLI   RPT_Ø5F+1,X'FF'         Q. REPORT Ø4 REQUESTED?
         BNE   NDY_Ø5F                 A. NO, CHECK FOR NEXT REPORT
         MVC   UN99_ØØØ+RPTXX$$$(5),REPORTØ5 MOVE IN THE DDNAME
         LA    R1,SVC_99RB             GET @(SVC 99 PLIST)
         SVC   99                      TRY THE ALLOCATE
         LTR   R15,R15                 Q. ALLOCATE SUCCESSFUL
```

```
         BZ    NDY_Ø5F                 A. YES, CHECK FOR NEXT REPORT
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* COME THROUGH HERE ONLY IF WE ENCOUNTER AN ERROR.                     *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         MVI   RPT_Ø5F+1,X'ØØ'         TURN OFF THE REPORT FLAG
         MVI   MSG_BUFF,C' '           BLANK IN FIRST BYTE
         MVC   MSG_BUFD(MSG_BUFL-1),MSG_BUFF BLANK THE REMAINDER
         LH    R14,EL_MSGØ6            PICK UP MESSAGE LENGTH
         BCTR  R14,Ø                   DECREMENT IT BY 1
         LA    R1,ER_MSGØ6             PICK UP THE ADDRESS OF MESSAGE
         EX    R14,MSG_MOVE            MOVE IN THE MESSAGE
         MVC   MSG_BUFF+(ER_MSG66-ER_MSGØ6)(5),REPORTØ5
         PUT   MSG,MSG_BUFF
         SPACE 1
NDY_Ø5F  DS    ØH
         SPACE 1
         MVC   UN99_ØØØ(MODEL99L),MODEL99 MOVE IN THE SVC 99 TEXT
         CLI   RPT_Ø6F+1,X'FF'         Q. REPORT Ø4 REQUESTED?
         BNE   NDY_Ø6F                 A. NO, CHECK FOR NEXT REPORT
         MVC   UN99_ØØØ+RPTXX$$$(5),REPORTØ6 MOVE IN THE DDNAME
         LA    R1,SVC_99RB             GET @(SVC 99 PLIST)
         SVC   99                      TRY THE ALLOCATE
         LTR   R15,R15                 Q. ALLOCATE SUCCESSFUL
         BZ    NDY_Ø6F                 A. PROCEED TO OPEN THE FILES
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* COME THROUGH HERE ONLY IF WE ENCOUNTER AN ERROR.                     *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         MVI   RPT_Ø6F+1,X'ØØ'         TURN OFF THE REPORT FLAG
         MVI   MSG_BUFF,C' '           BLANK IN FIRST BYTE
         MVC   MSG_BUFD(MSG_BUFL-1),MSG_BUFF BLANK THE REMAINDER
         LH    R14,EL_MSGØ6            PICK UP MESSAGE LENGTH
         BCTR  R14,Ø                   DECREMENT IT BY 1
         LA    R1,ER_MSGØ6             PICK UP THE ADDRESS OF MESSAGE
         EX    R14,MSG_MOVE            MOVE IN THE MESSAGE
         MVC   MSG_BUFF+(ER_MSG66-ER_MSGØ6)(5),REPORTØ5
         PUT   MSG,MSG_BUFF
         SPACE 1
NDY_Ø6F  DS    ØH
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* LOAD REGISTERS FOR A BXLE LOOP TO OPEN ALL NEEDED REPORT FILES.      *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         LA    R8,RPT_LLL              GET THE INCREMENT SIZE
         LA    R9,RPT_EEE              GET @(LAST ENTRY)
         SR    R9,R8                   ADJUST TO @(LAST ENTRY - 1)
         LA    R7,RPT_Ø1F              GET @(FIRST ENTRY)
         SPACE 1
```

```
OPN_RPT  DS    ØH
         SPACE 1
         CLI   1(R7),X'FF'             Q. IS THIS FILE NEEDED ?
         BNE   OPN_RPTZ                A. NO, CONSIDER NEXT ENTRY
         ICM   R2,B'1111',2(R7)        A. YES, GET @(DCB)
         USING IHADCB,R2               DECLARE A BASE
         MVI   MSG_BUFF,C' '           BLANK IN FIRST BYTE
         MVC   MSG_BUFD(MSG_BUFL-1),MSG_BUFF BLANK THE REMAINDER
         LH    R14,EL_MSGØ5            PICK UP MESSAGE LENGTH
         BCTR  R14,Ø                   DECREMENT IT DOWN BY 1
         LA    R1,ER_MSGØ5             PICK UP THE ADDRESS OF MESSAGE
         EX    R14,MSG_MOVE            MOVE IN THE MESSAGE
         LA    R14,MSG_BUFD            GET @(DATA AREA)
         LA    R14,ER_MSG55-ER_MSGØ5(,R14) INCREMENT THE POINTER
         MVC   Ø(8,R14),DCBDDNAM       MOVE IN THE DDNAME
         OPEN  ((R2),OUTPUT),MODE=31
         ICM   R2,B'1111',2(R7)        REFRESH REGISTER 2
         TM    DCBOFLGS,DCBOFOPN       Q. OPEN CLEAN?
         BO    OPN_RPTZ                A. YES, PROCEED
         PUT   MSG,MSG_BUFF
         DROP  R2
         SPACE 1
OPN_RPTZ DS    ØH
         SPACE 1                       CONTINUE TO PROCESS ALL ENTRIES
         BXLE  R7,R8,OPN_RPT
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* READ A RECORD FROM THE TMC, AND PROCESS IT ACCORDINGLY.              *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
READ_TMC DS    ØH
         SPACE 1
         GET   TMC
         LR    R2,R1                   PICK UP @(CURRENT TMC RECORD)
         CLC   TMSCTL,Ø(R2)            Q. TMC CONTROL RECORD
         BE    READ_TMC                A. YES, BYPASS THE RECORD
         CLI   Ø(R2),X'FF'             Q. DSNB RECORD
         BNE   GOT_TMCR                A. NO, THIS IS A VOLUME RECORD
         CLI   RPT_Ø1F+1,X'FF'         Q. REPORT 1 REQUESTED
         BNE   RPT_ØØ1F                A. NO, CHECK NEXT REPORT
         LA    R14,RPT_ØØ1F            PICK UP RETURN ADDRESS
         STCM  R14,B'1111',RET_ADDR    SAVE IT
         B     $EPORTØ1                PROCESS REPORT 1
         SPACE 1
RPT_ØØ1F DS    ØH
         SPACE 1
         CLI   RPT_Ø2F+1,X'FF'         Q. REPORT 2 REQUESTED
         BNE   READ_TMC                A. NO, GET NEXT TMC RECORD
         LA    R14,RPT_ØØ2F            PICK UP RETURN ADDRESS
         STCM  R14,B'1111',RET_ADDR    SAVE IT
         B     $EPORTØ2                PROCESS REPORT 2
         SPACE 1
```

```
RPT_ØØ2F DS    ØH
         SPACE 1
         B     READ_TMC                GO READ ANOTHER TMC RECORD
         SPACE 1
GOT_TMCR DS    ØH
         SPACE 1
         CLI   RPT_Ø3F+1,X'FF'         Q. REPORT 3 REQUESTED
         BNE   RPT_ØØ3F                A. NO, CHECK NEXT REPORT
         LA    R14,RPT_ØØ3F            PICK UP RETURN ADDRESS
         STCM  R14,B'1111',RET_ADDR    SAVE IT
         B     $EPORTØ3                PROCESS THE REPORT
         SPACE 1
RPT_ØØ3F DS    ØH
         SPACE 1
         CLI   RPT_Ø4F+1,X'FF'         Q. REPORT 4 REQUESTED
         BNE   RPT_ØØ4F                A. NO, PROCESS NEXT REPORT
         LA    R14,RPT_ØØ4F            PICK UP THE RETURN ADDRESS
         STCM  R14,B'1111',RET_ADDR    SAVE IT FOR LATER
         B     $EPORTØ4                PROCESS THE REPORT
         SPACE 1
RPT_ØØ4F DS    ØH
         SPACE 1
         CLI   RPT_Ø5F+1,X'FF'         Q. REPORT 5 REQUESTED
         BNE   RPT_ØØ5F                A. NO, PROCESS NEXT REPORT
         LA    R14,RPT_ØØ5F            PICK UP RETURN ADDRESS
         STCM  R14,B'1111',RET_ADDR    SAVE IT FOR LATER
         B     $EPORTØ5                PROCESS THE REPORT
         SPACE 1
RPT_ØØ5F DS    ØH
         SPACE 1
         CLI   RPT_Ø6F+1,X'FF'         Q. REPORT 6 REQUESTED
         BNE   RPT_ØØ6F                A. NO
         LA    R14,RPT_ØØ6F            PICK UP RETURN ADDRESS
         STCM  R14,B'1111',RET_ADDR    SAVE IT FOR LATER
         B     $EPORTØ6                PROCESS THE REPORT
         SPACE 1
RPT_ØØ6F DS    ØH
         SPACE 1
         B     READ_TMC
         SPACE 1
EOF_TMC  DS    ØH
EXIT_PGM DS    ØH
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* CLOSE UP ALL FILES THAT ARE OPEN.                                    *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         CLI   TMC_FLAG,DCBOFOPN       Q. TMC STILL OPEN
         BNE   TMC_CLO                 A. NO, IT IS CLOSED
         CLOSE (TMC),MODE=31
         SPACE 1
TMC_CLO  DS    ØH
```

```
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+---+----+----+----*
* LOAD REGISTERS FOR A BXLE LOOP TO CLOSE ALL OPEN REPORT FILES.      *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         LA    R8,RPT_LLL              GET THE INCREMENT SIZE
         LA    R9,RPT_EEE              GET @(LAST ENTRY)
         SR    R9,R8                   ADJUST TO @(LAST ENTRY - 1)
         LA    R7,RPT_Ø1F              GET @(FIRST ENTRY)
         SPACE 1
NXT_RPTE DS    ØH
         SPACE 1
         CLI   Ø(R7),DCBOFOPN          A. REPORT FILE STILL OPEN
         BNE   NXT_RPTF                A. NO, CHECK NEXT FILE
         ICM   R1,B'1111',2(R7)        PICK UP THE DCB
         CLOSE ((R1)),MODE=31
         SPACE 1
NXT_RPTF DS    ØH
         SPACE 1
         BXLE  R7,R8,NXT_RPTE          LOOP TILL ALL ENTRIES DONE
         SPACE 1
         CLI   MSG_FLAG,DCBOFOPN       Q. TMC STILL OPEN
         BNE   MSG_CLO                 A. NO, IT IS CLOSED
         CLOSE (MSG),MODE=31
         SPACE 1
MSG_CLO  DS    ØH
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* EXIT PROGRAM AND RETURN TO THE OPERATING SYSTEM.                     *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         $ESAEPI RET_CODE
         TITLE 'CA1REPRT - COMMON SYNAD ROUTINE FOR REPORT FILES'
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* COMMON CODE FOR PHYSICAL ERRORS ON ANY REPORT FILE.                  *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
SYN_RPT  DS    ØH
         MVI   MSG_BUFF,C' '           BLANK IN FIRST BYTE
         MVC   MSG_BUFD(MSG_BUFL-1),MSG_BUFF BLANK THE REMAINDER
         LH    R14,EL_MSGØ5            PICK UP MESSAGE LENGTH
         BCTR  R14,Ø                   DECREMENT IT DOWN BY 1
         LA    R1,ER_MSGØ5             PICK UP THE ADDRESS OF MESSAGE
         EX    R14,MSG_MOVE            MOVE IN THE MESSAGE
         LA    R14,MSG_BUFD            GET @(DATA AREA)
         LA    R14,ER_MSG55-ER_MSGØ5(,R14) INCREMENT THE POINTER
         MVC   Ø(L'RPT_ID,R14),RPT_ID  MOVE IN THE REPORT ID
         PUT   MSG,MSG_BUFF
         ICM   R14,B'1111',B_ADDR      PICK UP A RETURN ADDRESS
         BR    R14                     BRANCH BACK
```

```
         TITLE 'CA1REPRT - PROCESSING SECTION FOR REPORT Ø1'
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* RPTØ1 - PROCESS ALL DSNB ENTRIES.                                    *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
$EPORTØ1 DS    ØH
         CLC   CNT_Ø1D,RPT_MLIN        Q. MAX LINES PRINTED
         BNE   RE$ORTØ1                A. NO
         MVC   RPT_BUFF(RDH_Ø1ØØ),HDR_Ø1ØØ GET REPORT HEADER
         PUT   RPT_Ø1D,RPT_BUFF
         MVI   RPT_BUFF,C' '           BLANK OUT FIRST BYTE
         MVC   RPT_BUFF+1(L'RPT_BUFF-1),RPT_BUFF BLANK REMAINDER
         PUT   RPT_Ø1D,RPT_BUFF
         LA    R14,2                   SET LINE COUNTER
         STH   R14,CNT_Ø1D             SAVE IT
         SPACE 1
RE$ORTØ1 DS    ØH
         SPACE 1
         USING DSNBRLO,R2              SET UP A BASE FOR THE DSNB
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* MOVE IN THE OUTPUT MODEL, AND POPULATE IT WITH DSNB INFORMATION.     *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         MVC   RPT_BUFF(LDM_Ø1ØØ),MDL_Ø1ØØ
         MVC   RPT_BUFF+MDL_Ø1Ø1(L'DSNBFVSN),DSNBFVSN
         MVC   RPT_BUFF+MDL_Ø1Ø2(L'DSNBVSN),DSNBVSN
         MVC   RPT_BUFF+MDL_Ø1Ø4(L'DSNBDSN),DSNBDSN
         MVC   RPT_BUFF+MDL_Ø1Ø5(L'DSNBCJN),DSNBCJN
         MVC   RPT_BUFF+MDL_Ø1Ø6(L'DSNBCSN),DSNBCSN
         MVC   RPT_BUFF+MDL_Ø1Ø7(L'DSNBCPGM),DSNBCPGM
         XR    R14,R14                 CLEAR OUT REGISTER 14
         ICM   R14,B'ØØ11',DSNBFSN     GET FILE SEQUENCE NUMBER
         CVD   R14,D_WORK              CONVERT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø1Ø3(4),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø1Ø3+3,X'FØ' FIX THE SIGN
         PUT   RPT_Ø1D,RPT_BUFF
         LH    R14,CNT_Ø1D             GET THE LINE COUNTER
         LA    R14,1(,R14)             BUMP IT UP
         STH   R14,CNT_Ø1D             SAVE IT
R$PORTØ1 DS    ØH
         ICM   R14,B'1111',RET_ADDR    PICK UP A RETURN ADDRESS
         BR    R14                     BRANCH BACK
         DROP  R2
         TITLE 'CA1REPRT - PROCESSING SECTION FOR REPORT Ø2'
$EPORTØ2 DS    ØH
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* RPTØ2 - PROCESS ALL ACTIVE DSNB ENTRIES.                             *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
```

```
         SPACE 1
         USING DSNBRLO,R2              SET UP A BASE FOR THE DSNB
         CLI   DSNBACT,DSNBACTV        Q. ACTIVE DSNB
         BNE   R$PORTØ2                A. NO, BYPASS THIS DSNB
         CLC   CNT_Ø2D,RPT_MLIN        Q. MAX LINES PRINTED
         BNE   RE$ORTØ2                A. NO
         MVC   RPT_BUFF(RDH_Ø2ØØ),HDR_Ø2ØØ GET THE HEADER
         PUT   RPT_Ø2D,RPT_BUFF
         MVI   RPT_BUFF,C' '           BLANK FIRST BYTE
         MVC   RPT_BUFF+1(L'RPT_BUFF-1),RPT_BUFF BLANK THE REST
         PUT   RPT_Ø2D,RPT_BUFF
         LA    R14,2                   SET LINE COUNT
         STH   R14,CNT_Ø2D             SAVE IT
RE$ORTØ2 DS    ØH
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* MOVE IN THE OUTPUT MODEL, AND POPULATE IT WITH DSNB INFORMATION.    *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         MVC   RPT_BUFF(LDM_Ø2ØØ),MDL_Ø3ØØ
         MVC   RPT_BUFF+MDL_Ø2Ø1(L'DSNBFVSN),DSNBFVSN
         MVC   RPT_BUFF+MDL_Ø2Ø2(L'DSNBVSN),DSNBVSN
         MVC   RPT_BUFF+MDL_Ø2Ø4(L'DSNBDSN),DSNBDSN
         MVC   RPT_BUFF+MDL_Ø2Ø5(L'DSNBCJN),DSNBCJN
         MVC   RPT_BUFF+MDL_Ø2Ø6(L'DSNBCSN),DSNBCSN
         MVC   RPT_BUFF+MDL_Ø2Ø7(L'DSNBCPGM),DSNBCPGM
         XR    R14,R14                 CLEAR REGISTER 14
         ICM   R14,B'ØØ11',DSNBFSN     GET FILE SEQUENCE NUMBER
         CVD   R14,D_WORK              CONVERT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø2Ø3(4),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø2Ø3+3,X'FØ' FIX THE SIGN
         ICM   R14,B'1111',DSNBLREC    GET THE LRECL
         CVD   R14,D_WORK              CONVERT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø2Ø8(5),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø2Ø8+4,X'FØ' FIX THE SIGN
         ICM   R14,B'1111',DSNBBLKS    GET BLOCK SIZE
         CVD   R14,D_WORK              CONVERT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø2Ø9(5),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø2Ø9+4,X'FØ' FIX THE SIGN
         ICM   R14,B'1111',DSNBBLKC    GET NUMBER OF BLOCKS
         CVD   R14,D_WORK              CONVERT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø21Ø(5),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø21Ø+4,X'FØ' FIX THE SIGN
         XR    R8,R8                   CLEAR REGISTER 8
         ICM   R9,B'1111',DSNBBLKS     GET BLOCK SIZE
         ICM   R7,B'1111',DSNBBLKC     GET NUMBER OF BLOCKS
         MR    R8,R7                   COMPUTE PRODUCT
         D     R8,F_1Ø24               CONVER TO 1K UNITS
         LTR   R8,R8                   Q. REMAINDER > Ø
         BZ    REP$RTØ2                A. NO
         LA    R9,1(,R9)               A. YES, ROUND UP 1 BLOCK
         SPACE 1
```

```
REP$RTØ2 DS    ØH
         SPACE 1
         CVD   R9,D_WORK               CONVERT IT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø211(7),D_WORK+4(4) UNPACK IT
         OI    RPT_BUFF+MDL_Ø211+6,X'FØ' FIX THE SIGN
         PUT   RPT_Ø2D,RPT_BUFF
         LH    R14,CNT_Ø2D             GET LINE COUNTER
         LA    R14,1(,R14)             INCREMENT IT
         STH   R14,CNT_Ø2D             SAVE IT
         SPACE 1
R$PORTØ2 DS    ØH
         SPACE 1
         ICM   R14,B'1111',RET_ADDR    PICK UP A RETURN ADDRESS
         BR    R14                     BRANCH BACK
         DROP  R2
         TITLE 'CA1REPRT - PROCESSING SECTION FOR REPORT Ø3'
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* RPTØ3 - PROCESS ALL TMC VOLUME RECORDS.                              *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
$EPORTØ3 DS    ØH
         USING TMSRECLO,R2             SET UP A BASE FOR THE DSNB
         CLC   CNT_Ø3D,RPT_MLIN        Q. MAX LINES PRINTED
         BNE   RE$ORTØ3                A. NO
         MVC   RPT_BUFF(RDH_Ø3ØØ),HDR_Ø3ØØ GET THE HEADER
         PUT   RPT_Ø3D,RPT_BUFF
         MVI   RPT_BUFF,C' '           BLANK IN BYTE 1
         MVC   RPT_BUFF+1(L'RPT_BUFF-1),RPT_BUFF NOW COPY TO REST
         PUT   RPT_Ø3D,RPT_BUFF
         LA    R14,2                   SET THE COUNTER
         STH   R14,CNT_Ø3D             SAVE IT
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* OUTPUT THE HEADER AND DATA RECORDS.                                  *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
RE$ORTØ3 DS    ØH
         SPACE 1
         MVC   RPT_BUFF(LDM_Ø3ØØ),MDL_Ø3ØØ MOVE IN MODEL
         MVC   RPT_BUFF+MDL_Ø3Ø1(L'RLVOLSER),RLVOLSER GET VOLSER
         MVC   RPT_BUFF+MDL_Ø3Ø2(L'T_342Ø),T_342Ø ASSUME 342Ø
         TM    RLTRTCH,RL18TRK         Q. 348Ø DEVICE TYPE
         BNO   R$$ORTØ3                A. NO, CHECK NEXT TYPE
         MVC   RPT_BUFF+MDL_Ø3Ø2(L'T_348Ø),T_348Ø
         SPACE 1
R$$ORTØ3 DS    ØH
         SPACE 1
         TM    RLTRTCH,RL36TRK         Q. 349Ø DEVICE TYPE
         BNO   R$$$RTØ3                A. NO, CHECK NEXT TYPE
         MVC   RPT_BUFF+MDL_Ø3Ø2(L'T_349Ø),T_349Ø
         SPACE 1
```

```
R$$$RTØ3 DS    ØH
         SPACE 1
         TM    RLTRTCH,RL36TRK2        Q. 349Ø EXTENDED MEDIA
         BNO   R$$$$TØ3                A. NO, CHECK NEXT TYPE
         MVC   RPT_BUFF+MDL_Ø3Ø2(L'T_349ØE),T_349ØE
         SPACE 1
R$$$$TØ3 DS    ØH
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* PROCESS THE FLAG BYTES.                                             *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         TM    RLFLAG1,RLDELET         Q. DELETE STATUS
         BNO   R$#$$TØ3                A. NO, NEXT
         MVI   RPT_BUFF+MDL_Ø3Ø3,C'D'  A. YES, INDICATE IT
R$#$$TØ3 DS    ØH
         TM    RLFLAG1,RLSCRTCH        Q. SCRATCH STATUS
         BNO   R$##$TØ3                A. NO, NEXT
         MVI   RPT_BUFF+MDL_Ø3Ø3+1,C'S' A. INDICATE IT
R$##$TØ3 DS    ØH
         TM    RLFLAG1,RLUPDATE        Q. MANUAL UPDATE
         BNO   R$###TØ3                A. NO, NEXT
         MVI   RPT_BUFF+MDL_Ø3Ø3+2,C'U' A. INDICATE IT
R$###TØ3 DS    ØH
         TM    RLFLAG3,RLEDMTAP        Q. EXTERNAL MANAGER
         BNO   R$_$$TØ3                A. NO, NEXT
         MVI   RPT_BUFF+MDL_Ø3Ø3+3,C'E' A. INDICATE IT
R$_$$TØ3 DS    ØH
         TM    RLFLAG4,RLISCAT         Q. IN MVS CATALOG
         BNO   R$__$TØ3                A. NO
         MVI   RPT_BUFF+MDL_Ø3Ø3+4,C'C' A. INDICATE IT
         SPACE 1
R$__$TØ3 DS    ØH
         SPACE 1
         MVC   RPT_BUFF+MDL_Ø3Ø4(L'RLDSN),RLDSN
         MVC   RPT_BUFF+MDL_Ø3Ø5(L'RLJOBNM),RLJOBNM
         MVC   RPT_BUFF+MDL_Ø3Ø6(L'RLSTPNAM),RLSTPNAM
         MVC   RPT_BUFF+MDL_Ø3Ø7(L'RLCPGM),RLCPGM
         MVC   RPT_BUFF+MDL_Ø3Ø8(L'RLSMSMC),RLSMSMC
         ICM   R14,B'1111',RLLRECL     GET LRECL
         CVD   R14,D_WORK              CONVERT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø3Ø9(5),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø3Ø9+4,X'FØ' FIX THE SIGN
         ICM   R14,B'1111',RLBLKSI     GET THE BLOCK SIZE
         CVD   R14,D_WORK              CONVERT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø31Ø(5),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø31Ø+4,X'FØ' FIX THE SIGN
         ICM   R14,B'1111',RLBLKCNT    GET THE NUMBER OF BLOCKS
         CVD   R14,D_WORK              CONVERT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø311(5),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø311+4,X'FØ' FIX THE SIGN
         SPACE 1
```

```
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* TAKE THE BLOCKSIZE AND THE BLOCK COUNT AND COMPUTE THE NUMBER OF    *
* KILOBYTES IN THE CURRENT TAPE DATASET.                              *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         XR    R8,R8                   CLEAR REGISTER 8
         ICM   R9,B'1111',RLBLKSI      PICK UP THE BLOCKSIZE
         ICM   R7,B'1111',RLBLKCNT     PICK UP THE BLOCK COUNT
         MR    R8,R7                   COMPUTE THE PRODUCT
         D     R8,F_1Ø24               DIVIDE BY 1K
         LTR   R8,R8                   Q. REMAINDER ZERO?
         BZ    REP#RTØ3                A. YES, BRANCH
         LA    R9,1(,R9)               A. NO, BUMP 1K BLOCKS UP
REP#RTØ3 DS    ØH
         CVD   R9,D_WORK               MAKE IT DECIMAL
         UNPK  RPT_BUFF+MDL_Ø312(7),D_WORK+4(4) UNPACK IT
         OI    RPT_BUFF+MDL_Ø312+6,X'FØ' FIX THE SIGN
         PUT   RPT_Ø3D,RPT_BUFF
         LH    R14,CNT_Ø3D             GET THE LINE COUNTER
         LA    R14,1(,R14)             BUMP IT UP BY 1
         STH   R14,CNT_Ø3D             SAVE IT
R$PORTØ3 DS    ØH
         ICM   R14,B'1111',RET_ADDR    PICK UP A RETURN ADDRESS
         BR    R14                     BRANCH BACK
         DROP  R2
         TITLE 'CA1REPRT - PROCESSING SECTION FOR REPORT Ø4'
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* RPTØ4 - PROCESS ALL ACTIVE TMC RECORDS.                             *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
$EPORTØ4 DS    ØH
         USING TMSRECLO,R2             SET UP A BASE FOR THE DSNB
         TM    RLFLAG1,RLDELET         Q. VOLUME IN DELETE STATUS
         BO    R$PORTØ4                A. YES, BYPASS THIS RECORD
         TM    RLFLAG1,RLSCRTCH        Q. VOLUME IN SCRATCH STATUS
         BO    R$PORTØ4                A. YES, BYPASS THIS RECORD
         CLC   CNT_Ø4D,RPT_MLIN        Q. MAX LINES PRINTED
         BNE   RE$ORTØ4                A. NO
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* OUTPUT THE HEADER AND DATA RECORDS.                                 *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         MVC   RPT_BUFF(RDH_Ø4ØØ),HDR_Ø4ØØ GET THE HEADER
         PUT   RPT_Ø4D,RPT_BUFF
         MVI   RPT_BUFF,C' '           BLANK THE FIRST BYTE
         MVC   RPT_BUFF+1(L'RPT_BUFF-1),RPT_BUFF NOW THE REST
         PUT   RPT_Ø4D,RPT_BUFF
         LA    R14,2                   SET THE LINE COUNTER
         STH   R14,CNT_Ø4D             SAVE IT
         SPACE 1
```

```
RE$ORTØ4 DS    ØH
         SPACE 1
         MVC   RPT_BUFF(LDM_Ø4ØØ),MDL_Ø4ØØ GET THE MODEL
         MVC   RPT_BUFF+MDL_Ø4Ø1(L'RLVOLSER),RLVOLSER VOLSER
         MVC   RPT_BUFF+MDL_Ø4Ø2(L'RLDSN),RLDSN DATASET NAME
         MVC   RPT_BUFF+MDL_Ø4Ø3(L'T_342Ø),T_342Ø ASSUME 342Ø
         TM    RLTRTCH,RL18TRK         Q. 348Ø DEVICE TYPE
         BNO   R$$ORTØ4                A. NO, CHECK NEXT TYPE
         MVC   RPT_BUFF+MDL_Ø4Ø3(L'T_348Ø),T_348Ø
         SPACE 1
R$$ORTØ4 DS    ØH
         SPACE 1
         TM    RLTRTCH,RL36TRK         Q. 349Ø DEVICE TYPE
         BNO   R$$$RTØ4                A. NO, CHECK NEXT TYPE
         MVC   RPT_BUFF+MDL_Ø4Ø3(L'T_349Ø),T_349Ø
         SPACE 1
R$$$RTØ4 DS    ØH
         SPACE 1
         TM    RLTRTCH,RL36TRK2        Q. 349Ø EXTENDED MEDIA
         BNO   R$$$$TØ4                A. NO, CHECK NEXT TYPE
         MVC   RPT_BUFF+MDL_Ø4Ø3(L'T_349ØE),T_349ØE
         SPACE 1
R$$$$TØ4 DS    ØH
         SPACE 1
         MVC   RPT_BUFF+MDL_Ø4Ø4(L'RLJOBNM),RLJOBNM
         MVC   RPT_BUFF+MDL_Ø4Ø5(L'RLSTPNAM),RLSTPNAM
         MVC   RPT_BUFF+MDL_Ø4Ø6(L'RLCPGM),RLCPGM
         MVC   RPT_BUFF+MDL_Ø4Ø7(L'RLSMSMC),RLSMSMC
         ICM   R14,B'1111',RLLRECL     GET THE LRECL
         CVD   R14,D_WORK              CONVERT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø4Ø8(5),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø4Ø8+4,X'FØ' FIX THE SIGN
         ICM   R14,B'1111',RLBLKSI     GET THE BLOCK SIZE
         CVD   R14,D_WORK              CONVER TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø4Ø9(5),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø4Ø9+4,X'FØ' FIX THE SIGN
         ICM   R14,B'1111',RLBLKCNT    GET THE BLOCK COUNT
         CVD   R14,D_WORK              CONVERT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø41Ø(5),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø41Ø+4,X'FØ' FIX THE SIGN
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* TAKE THE BLOCKSIZE AND THE BLOCK COUNT AND COMPUTE THE NUMBER OF    *
* KILOBYTES THE CURRENT TAPE DATASET                                  *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         XR    R8,R8                   CLEAR REGISTER 8
         ICM   R9,B'1111',RLBLKSI      PICK UP THE BLOCKSIZE
         ICM   R7,B'1111',RLBLKCNT     PICK UP THE BLOCK COUNT
         MR    R8,R7                   COMPUTE THE PRODUCT
         D     R8,F_1Ø24               DIVIDE BY 1K
         LTR   R8,R8                   Q. REMAINDER ZERO?
```

```
         BZ    REP#RTØ4                A. YES, BRANCH
         LA    R9,1(,R9)               A. NO, BUMP 1K BLOCKS UP
         SPACE 1
REP#RTØ4 DS    ØH
         SPACE 1
         CVD   R9,D_WORK               MAKE IT DECIMAL
         UNPK  RPT_BUFF+MDL_Ø411(7),D_WORK+4(4) UNPACK IT
         OI    RPT_BUFF+MDL_Ø411+6,X'FØ' FIX THE SIGN
         PUT   RPT_Ø4D,RPT_BUFF
         LH    R14,CNT_Ø4D             GET THE LINE COUNTER
         LA    R14,1(,R14)             BUMP IT UP BY 1
         STH   R14,CNT_Ø4D             SAVE IT
R$PORTØ4 DS    ØH
         ICM   R14,B'1111',RET_ADDR    PICK UP A RETURN ADDRESS
         BR    R14                     BRANCH BACK
         DROP  R2
         TITLE 'CA1REPRT - PROCESSING SECTION FOR REPORT Ø5'
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* RPTØ5 - TAPE ERROR REPORT.                                          *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
$EPORTØ5 DS    ØH
         USING TMSRECLO,R2             SET UP A BASE FOR THE VOL REC
         TM    RLFLAG1,RLDELET         Q. VOLUME IN DELETE STATUS
         BO    R$PORTØ5                A. YES, BYPASS THIS RECORD
         TM    RLFLAG1,RLSCRTCH        Q. VOLUME IN SCRATCH STATUS
         BO    R$PORTØ5                A. YES, BYPASS THIS RECORD
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* CHECK THE ERROR COUNTERS TO SEE IF WE NEED TO PROCESS THIS RECORD.  *
* THRESHOLD TO CHECK AGAINST IS USER DEFINABLE IN E_THRESH.           *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         XR    R14,R14
         ICM   R14,B'ØØ11',RLPRERRC    GET ERROR COUNT
         C     R14,E_THRESH            Q. COMPARE TO THRESHOLD
         BNL   E_NOTLOW                A. PROCESS THIS RECORD
         ICM   R14,B'ØØ11',RLPWERRC    GET ERROR COUNT
         C     R14,E_THRESH            Q. COMPARE TO THRESHOLD
         BNL   E_NOTLOW                A. PROCESS THIS RECORD
         ICM   R14,B'ØØ11',RLPRERRI    GET ERROR COUNT
         C     R14,E_THRESH            Q. COMPARE TO THRESHOLD
         BNL   E_NOTLOW                A. PROCESS THIS RECORD
         ICM   R14,B'ØØ11',RLPWERRI    GET ERROR COUNT
         C     R14,E_THRESH            Q. COMPARE TO THRESHOLD
         BL    R$PORTØ5                A. SKIP TO NEXT RECORD
         SPACE 1
E_NOTLOW DS    ØH
         SPACE 1
         CLC   CNT_Ø5D,RPT_MLIN        Q. MAX LINES PRINTED
         BNE   RE$ORTØ5                A. NO
```

```
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* OUTPUT THE HEADER AND DATA RECORDS.                                  *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         MVC   RPT_BUFF(RDH_Ø5ØØ),HDR_Ø5ØØ GET THE HEADER
         PUT   RPT_Ø5D,RPT_BUFF
         MVI   RPT_BUFF,C' '           BLANK IN FIRST BYTE
         MVC   RPT_BUFF+1(L'RPT_BUFF-1),RPT_BUFF NOW THE REST
         PUT   RPT_Ø5D,RPT_BUFF
         LA    R14,2                   SET THE COUNTER
         STH   R14,CNT_Ø5D             SAVE IT
         SPACE 1
RE$ORTØ5 DS    ØH
         SPACE 1
         MVC   RPT_BUFF(LDM_Ø5ØØ),MDL_Ø5ØØ MOVE IN THE MODEL
         MVC   RPT_BUFF+MDL_Ø5Ø1(L'RLVOLSER),RLVOLSER VOLSER
         MVC   RPT_BUFF+MDL_Ø5Ø2(L'RLDSN),RLDSN DATASET NAME
         MVC   RPT_BUFF+MDL_Ø5Ø3(L'T_342Ø),T_342Ø ASSUME 342Ø
         TM    RLTRTCH,RL18TRK         Q. 348Ø DEVICE TYPE
         BNO   R$$ORTØ5                A. NO, CHECK NEXT TYPE
         MVC   RPT_BUFF+MDL_Ø5Ø3(L'T_348Ø),T_348Ø
         SPACE 1
R$$ORTØ5 DS    ØH
         SPACE 1
         TM    RLTRTCH,RL36TRK         Q. 349Ø DEVICE TYPE
         BNO   R$$$RTØ5                A. NO, CHECK NEXT TYPE
         MVC   RPT_BUFF+MDL_Ø5Ø3(L'T_349Ø),T_349Ø
         SPACE 1
R$$$RTØ5 DS    ØH
         SPACE 1
         TM    RLTRTCH,RL36TRK2        Q. 349Ø EXTENDED MEDIA
         BNO   R$$$$TØ5                A. NO, CHECK NEXT TYPE
         MVC   RPT_BUFF+MDL_Ø5Ø3(L'T_349ØE),T_349ØE
         SPACE 1
R$$$$TØ5 DS    ØH
         SPACE 1
         XR    R14,R14                 CLEAR REGISTER 14
         ICM   R14,B'ØØ11',RLPRERRC    PICK UP ERROR COUNT
         CVD   R14,D_WORK              CONVERT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø5Ø4(5),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø5Ø4+4,X'FØ' FIX THE SIGN
         XR    R14,R14                 CLEAR REGISTER 14
         ICM   R14,B'ØØ11',RLPWERRC    PICK UP ERROR COUNT
         CVD   R14,D_WORK              CONVERT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø5Ø5(5),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø5Ø5+4,X'FØ' FIX THE SIGN
         XR    R14,R14                 CLEAR REGISTER 14
         ICM   R14,B'ØØ11',RLPRERRI    PICK UP ERROR COUNT
         CVD   R14,D_WORK              CONVERT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø5Ø6(5),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø5Ø6+4,X'FØ' FIX THE SIGN
```

```
         XR    R14,R14                 CLEAR REGISTER 14
         ICM   R14,B'ØØ11',RLPWERRI    PICK UP ERROR COUNT
         CVD   R14,D_WORK              CONVERT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø5Ø7(5),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø5Ø7+4,X'FØ' FIX THE SIGN
         PUT   RPT_Ø5D,RPT_BUFF
         LH    R14,CNT_Ø5D             GET THE LINE COUNTER
         LA    R14,1(,R14)             BUMP IT UP BY 1
         STH   R14,CNT_Ø5D             SAVE IT
R$PORTØ5 DS    ØH
         ICM   R14,B'1111',RET_ADDR    PICK UP A RETURN ADDRESS
         BR    R14                     BRANCH BACK
         DROP  R2
         TITLE 'CA1REPRT - PROCESSING SECTION FOR REPORT Ø6'
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* RPTØ6 - PROCESS ALL ACTIVE TMC RECORDS.                              *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
$EPORTØ6 DS    ØH
         USING TMSRECLO,R2             SET UP A BASE FOR THE DSNB
         TM    RLFLAG1,RLDELET         Q. VOLUME IN DELETE STATUS
         BO    R$PORTØ6                A. YES, BYPASS THIS RECORD
         TM    RLFLAG1,RLSCRTCH        Q. VOLUME IN SCRATCH STATUS
         BO    R$PORTØ6                A. YES, BYPASS THIS RECORD
         LH    R14,DSN_MASL            GET MASK LENGTH
         EX    R14,CLC_MASK            Q. DSN MASK WE WAMT
         BNE   R$PORTØ6                A. NO, BYPASS THIS RECORD
         CLC   CNT_Ø6D,RPT_MLIN        Q. MAX LINES PRINTED
         BNE   RE$ORTØ6                A. NO
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* OUTPUT THE HEADER AND DATA RECORDS.                                  *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         MVC   RPT_BUFF(RDH_Ø6ØØ),HDR_Ø6ØØ GET THE HEADER
         PUT   RPT_Ø6D,RPT_BUFF
         MVI   RPT_BUFF,C' '           BLANK THE FIRST BYTE
         MVC   RPT_BUFF+1(L'RPT_BUFF-1),RPT_BUFF NOW THE REST
         PUT   RPT_Ø6D,RPT_BUFF
         LA    R14,2                   SET THE LINE COUNTER
         STH   R14,CNT_Ø6D             SAVE IT
         SPACE 1
RE$ORTØ6 DS    ØH
         SPACE 1
         MVC   RPT_BUFF(LDM_Ø6ØØ),MDL_Ø6ØØ GET THE MODEL
         MVC   RPT_BUFF+MDL_Ø6Ø1(L'RLVOLSER),RLVOLSER VOLSER
         MVC   RPT_BUFF+MDL_Ø6Ø2(L'RLDSN),RLDSN DATASET NAME
         MVC   RPT_BUFF+MDL_Ø6Ø3(L'T_342Ø),T_342Ø ASSUME 342Ø
         TM    RLTRTCH,RL18TRK         Q. 348Ø DEVICE TYPE
         BNO   R$$ORTØ6                A. NO, CHECK NEXT TYPE
         MVC   RPT_BUFF+MDL_Ø6Ø3(L'T_348Ø),T_348Ø
```

```
         SPACE 1
R$$ORTØ6 DS    ØH
         SPACE 1
         TM    RLTRTCH,RL36TRK         Q. 349Ø DEVICE TYPE
         BNO   R$$$RTØ6                A. NO, CHECK NEXT TYPE
         MVC   RPT_BUFF+MDL_Ø6Ø3(L'T_349Ø),T_349Ø
         SPACE 1
R$$$RTØ6 DS    ØH
         SPACE 1
         TM    RLTRTCH,RL36TRK2        Q. 349Ø EXTENDED MEDIA
         BNO   R$$$$TØ6                A. NO, CHECK NEXT TYPE
         MVC   RPT_BUFF+MDL_Ø6Ø3(L'T_349ØE),T_349ØE
         SPACE 1
R$$$$TØ6 DS    ØH
         SPACE 1
         MVC   RPT_BUFF+MDL_Ø6Ø4(L'RLJOBNM),RLJOBNM
         MVC   RPT_BUFF+MDL_Ø6Ø5(L'RLSTPNAM),RLSTPNAM
         MVC   RPT_BUFF+MDL_Ø6Ø6(L'RLCPGM),RLCPGM
         MVC   RPT_BUFF+MDL_Ø6Ø7(L'RLSMSMC),RLSMSMC
         ICM   R14,B'1111',RLLRECL     GET THE LRECL
         CVD   R14,D_WORK              CONVERT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø6Ø8(5),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø6Ø8+4,X'FØ' FIX THE SIGN
         ICM   R14,B'1111',RLBLKSI     GET THE BLOCK SIZE
         CVD   R14,D_WORK              CONVER TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø6Ø9(5),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø6Ø9+4,X'FØ' FIX THE SIGN
         ICM   R14,B'1111',RLBLKCNT    GET THE BLOCK COUNT
         CVD   R14,D_WORK              CONVERT TO DECIMAL
         UNPK  RPT_BUFF+MDL_Ø61Ø(5),D_WORK+5(3) UNPACK IT
         OI    RPT_BUFF+MDL_Ø61Ø+4,X'FØ' FIX THE SIGN
         SPACE 1
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
* TAKE THE BLOCKSIZE AND THE BLOCK COUNT AND COMPUTE THE NUMBER OF    *
* KILOBYTES THE CURRENT TAPE DATASET                                  *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
         SPACE 1
         XR    R8,R8                   CLEAR REGISTER 8
         ICM   R9,B'1111',RLBLKSI      PICK UP THE BLOCKSIZE
         ICM   R7,B'1111',RLBLKCNT     PICK UP THE BLOCK COUNT
         MR    R8,R7                   COMPUTE THE PRODUCT
         D     R8,F_1Ø24               DIVIDE BY 1K
         LTR   R8,R8                   Q. REMAINDER ZERO?
         BZ    REP#RTØ6                A. YES, BRANCH
         LA    R9,1(,R9)               A. NO, BUMP 1K BLOCKS UP
         SPACE 1
REP#RTØ6 DS    ØH
         SPACE 1
         CVD   R9,D_WORK               MAKE IT DECIMAL
         UNPK  RPT_BUFF+MDL_Ø611(7),D_WORK+4(4) UNPACK IT
         OI    RPT_BUFF+MDL_Ø611+6,X'FØ' FIX THE SIGN
         PUT   RPT_Ø6D,RPT_BUFF
```

```
         LH    R14,CNT_Ø6D             GET THE LINE COUNTER
         LA    R14,1(,R14)             BUMP IT UP BY 1
         STH   R14,CNT_Ø6D             SAVE IT
R$PORTØ6 DS    ØH
         ICM   R14,B'1111',RET_ADDR    PICK UP A RETURN ADDRESS
         BR    R14                     BRANCH BACK
         TITLE 'CA1REPRT - DEFINE REPORTS WE CAN REQUEST, WITH FLAGS'
         SPACE 1
         DS    ØH                      ALIGN THINGS
TRAN_I   TRT   Ø(*-*,R3),TRAN_TAB      TARGET OF A EXECUTE
CLC_MASK CLC   DSN_MASK(*-*),RLDSN     TARGET OF A EXECUTE
MOV_MASK MVC   DSN_MASK(*-*),Ø(R3)     TARGET OF A EXECUTE
E_THRESH DC    F'25'                   TAPE ERROR THRESHOLD VALUE
F_1Ø24   DC    F'1Ø24'                 USED FOR KILOBYTE CALCULATIONS
RPT_MLIN DC    H'6Ø'                   MAX LINES PER REPORT/PAGE
RPT_Ø1   DC    C'DSNB ALL'             REQUEST ALL DSNB RECORDS
RPT_Ø2   DC    C'DSNB ACTIVE'          REQUEST ACTIVE DSNB RECORDS
RPT_Ø3   DC    C'VOLUME ALL'           REQUEST ALL VOLUME RECORDS
RPT_Ø4   DC    C'VOLUME ACTIVE'        REQUEST ACTIVE VOLUME RECORDS
RPT_Ø5   DC    C'VOLUME ERROR'         REQUEST ACTIVE VOLUME W/ERRORS
RPT_Ø6   DC    C'VOLUME DSN='          REQUEST ACTIVE VOLUME BY/DSN
TMSCTL   DC    CL6'TMSCTL'             TMC CONTROL RECORD HEADER
T_342Ø   DC    CL5'342Ø'
T_348Ø   DC    CL5'348Ø'
T_349Ø   DC    CL5'349Ø'
T_349ØE  DC    CL5'349ØE'
T_359Ø   DC    CL5'359Ø'
REPORTØ1 DC    CL5'RPTØ1'              DDNAME FOR REPORT Ø1
REPORTØ2 DC    CL5'RPTØ2'              DDNAME FOR REPORT Ø2
REPORTØ3 DC    CL5'RPTØ3'              DDNAME FOR REPORT Ø3
REPORTØ4 DC    CL5'RPTØ4'              DDNAME FOR REPORT Ø4
REPORTØ5 DC    CL5'RPTØ5'              DDNAME FOR REPORT Ø5
REPORTØ6 DC    CL5'RPTØ6'              DDNAME FOR REPORT Ø6
         DROP  R2
         TITLE 'CA1REPRT - DEFINE PROGRAM MESSAGES'
         SPACE 1
         DS    ØH
MSG_MOVE MVC   MSG_BUFD+1(*-*),Ø(R1)   TARGET OF AN EXECUTE
ER_MSGØ1 DC    C'CA1RPTØ1 ERROR OPENING MESSSAGES DATASET, TERMINATING'
EL_MSGØ1 DC    Y(*-ER_MSGØ1)
ER_MSGØ2 DC    C'CA1RPTØ1 ERROR OPENING THE TAPE MANAGEMENT CATALOG, TE+
               RMINATING EXECUTION'
EL_MSGØ2 DC    Y(*-ER_MSGØ2)
ER_MSGØ3 DC    C'CA1RPTØ1 ERROR OPENING CONTROL DATASET, TERMINATING EX+
               ECUTION'
EL_MSGØ3 DC    Y(*-ER_MSGØ3)
ER_MSGØ4 DC    C'CA1RPTØ1 BAD CONTROL CARD ENCOUNTERED'
EL_MSGØ4 DC    Y(*-ER_MSGØ4)
ER_MSGØ5 DC    C'CA1RPTØ1 ERROR ENCOUNTERED ON REPORT FILE, DDNAME='
ER_MSG55 DC    CL8'        '
EL_MSGØ5 DC    Y(*-ER_MSGØ5)
ER_MSGØ6 DC    C'CA1RPTØ1 ERROR ENCOUNTERED ALLOCATING FILE DDNAME='
```

```
ER_MSG66 DC    CL5'        '
EL_MSGØ6 DC    Y(*-ER_MSGØ6)
         TITLE 'CA1REPRT - DEFINE THE SVC 99 MODEL PARAMETER LIST'
         SPACE 1
MODEL99  DS    ØF
DDNAM$$$ EQU   *-MODEL99              LET ASM CALCULATE DISPLACEMENT
         DC    AL2(DALDDNAM)          DDNAME
         DC    X'ØØØ1'
         DC    X'ØØØ5'                LENGTH OF THE DDNAME
RPTXX$$$ EQU   *-MODEL99              LET ASM CALCULATE DISPLACEMENT
         DC    C'RPTØ1'               SPECIFIED NAME
SYSOU$$$ EQU   *-MODEL99              LET ASM CALCULATE DISPLACEMENT
         DC    AL2(DALSYSOU)          SYSOUT DATASET
         DC    X'ØØØØ'
DSORG$$$ EQU   *-MODEL99              LET ASM CALCULATE DISPLACEMENT
         DC    AL2(DALDSORG)          DATASET ORGANIZATION
         DC    X'ØØØ1'
         DC    X'ØØØ2'
         DC    X'4ØØØ'                PHYSICAL SEQUENTIAL
LRECL$$$ EQU   *-MODEL99              LET ASM CALCULATE DISPLACEMENT
         DC    AL2(DALLRECL)          LOGICAL RECORD LENGTH
         DC    X'ØØØ1'
         DC    X'ØØØ2'
         DC    X'ØØ85'                SPECIFY 133
RECFM$$$ EQU   *-MODEL99              LET ASM CALCULATE DISPLACEMENT
         DC    AL2(DALRECFM)          RECORD FORMAT
         DC    X'ØØØ1'
         DC    X'ØØØ1'
         DC    X'9Ø'                  SPECIFY FIXED BLOCK
CLOSE$$$ EQU   *-MODEL99              LET ASM CALCULATE DISPLACEMENT
         DC    AL2(DALCLOSE)          UNALLOCATE ON CLOSE
         DC    X'ØØØØ'
MODEL99L EQU   *-MODEL99
         TITLE 'CA1REPRT - REPORT Ø1 RECORD LAYOUT'
HDR_ØlØØ DS    ØX
         DC    C'1'
         DC    CL6'VOLSER'
         DC    CL2' '
         DC    CL6'VOLSER'            VOLSER OF FIRST VOLUME
         DC    CL2' '                 FILE STARTS ON THIS VOLUME
         DC    CL4'FS #'              FILE SEQUENCE #
         DC    CL2' '
         DC    CL44'              DATASET NAME                  '
         DC    CL2' '
         DC    CL8'JOB NAME'          CREATING JOB
         DC    CL2' '
         DC    CL8'JOB STEP'          CREATING STEP
         DC    CL2' '
         DC    CL8'PROGRAM'           CREATING PROGRAM
         DC    (133-(*-HDR_ØlØØ))CL1' ' LET ASM FILL OUT
RDH_ØlØØ EQU   *-HDR_ØlØØ
MDL_ØlØØ DS    ØX
```

```
         DC    C' '
MDL_Ø1Ø1 EQU   *-MDL_Ø1ØØ
         DC    CL6' '                  FILE STARTS ON THIS VOLUME
         DC    CL2' '                  FILLER
MDL_Ø1Ø2 EQU   *-MDL_Ø1ØØ
         DC    CL6' '                  VOLSER OF FIRST VOLUME
         DC    CL2' '                  FILLER
MDL_Ø1Ø3 EQU   *-MDL_Ø1ØØ
         DC    CL4' '                  FILE SEQUENCE #
         DC    CL2' '                  FILLER
MDL_Ø1Ø4 EQU   *-MDL_Ø1ØØ
         DC    CL44' '                 DATASET NAME
         DC    CL2' '                  FILLER
MDL_Ø1Ø5 EQU   *-MDL_Ø1ØØ
         DC    CL8' '                  CREATING JOB
         DC    CL2' '                  FILLER
MDL_Ø1Ø6 EQU   *-MDL_Ø1ØØ
         DC    CL8' '                  CREATING STEP
         DC    CL2' '                  FILLER
MDL_Ø1Ø7 EQU   *-MDL_Ø1ØØ
         DC    CL8' '                  CREATING PROGRAM
         DC    (133-(*-MDL_Ø1ØØ))CL1' ' LET ASM FILL OUT
LDM_Ø1ØØ EQU   *-MDL_Ø1ØØ
         TITLE 'CA1REPRT - REPORT Ø2 RECORD LAYOUT'
HDR_Ø2ØØ DS    ØX
         DC    C'1'
         DC    CL6'VOLSER'
         DC    CL2' '
         DC    CL6'VOLSER'             VOLSER OF FIRST VOLUME
         DC    CL2' '                  FILE STARTS ON THIS VOLUME
         DC    CL4'FS #'               FILE SEQUENCE #
         DC    CL2' '
         DC    CL44'              DATASET NAME                  '
         DC    CL2' '
         DC    CL8'JOB NAME'           CREATING JOB
         DC    CL2' '
         DC    CL8'JOB STEP'           CREATING STEP
         DC    CL2' '
         DC    CL8'PROGRAM'            CREATING PROGRAM
         DC    CL2' '
         DC    CL5'LRECL'              LOGICAL RECORD SIZE
         DC    CL2' '
         DC    CL5'BLKSZ'              BLOCK SIZE
         DC    CL2' '
         DC    CL5'BLKCT'              BLOCK COUNT
         DC    CL2' '
         DC    CL7'K-BYTES'            KILOBYTES IN DATASET
         DC    CL2' '
         DC    (133-(*-HDR_Ø2ØØ))CL1' ' LET ASM FILL OUT
RDH_Ø2ØØ EQU   *-HDR_Ø2ØØ
MDL_Ø2ØØ DS    ØX
         DC    C' '
```

```
MDL_Ø2Ø1 EQU   *-MDL_Ø2ØØ
         DC    CL6' '                  FILE STARTS ON THIS VOLUME
         DC    CL2' '                  FILLER
MDL_Ø2Ø2 EQU   *-MDL_Ø2ØØ
         DC    CL6' '                  VOLSER OF FIRST VOLUME
         DC    CL2' '                  FILLER
MDL_Ø2Ø3 EQU   *-MDL_Ø2ØØ
         DC    CL4' '                  FILE SEQUENCE #
         DC    CL2' '                  FILLER
MDL_Ø2Ø4 EQU   *-MDL_Ø2ØØ
         DC    CL44' '                 DATASET NAME
         DC    CL2' '                  FILLER
MDL_Ø2Ø5 EQU   *-MDL_Ø2ØØ
         DC    CL8' '                  CREATING JOB
         DC    CL2' '                  FILLER
MDL_Ø2Ø6 EQU   *-MDL_Ø2ØØ
         DC    CL8' '                  CREATING STEP
         DC    CL2' '                  FILLER
MDL_Ø2Ø7 EQU   *-MDL_Ø2ØØ
         DC    CL8' '                  CREATING PROGRAM
         DC    CL2' '                  FILLER
MDL_Ø2Ø8 EQU   *-MDL_Ø2ØØ
         DC    CL5' '                  LOGICAL RECORD SIZE
         DC    CL2' '                  FILLER
MDL_Ø2Ø9 EQU   *-MDL_Ø2ØØ
         DC    CL5' '                  BLOCK SIZE
         DC    CL2' '                  FILLER
MDL_Ø21Ø EQU   *-MDL_Ø2ØØ
         DC    CL5' '                  BLOCK COUNT
         DC    CL2' '                  FILLER
MDL_Ø211 EQU   *-MDL_Ø2ØØ
         DC    CL7' '                  SIZE IN KILOBYTES
         DC    CL2' '                  FILLER
         DC    (133-(*-MDL_Ø2ØØ))CL1' ' LET ASM FILL OUT
LDM_Ø2ØØ EQU   *-MDL_Ø2ØØ
         TITLE 'CA1REPRT - REPORT Ø3 RECORD LAYOUT'
         SPACE 1
HDR_Ø3ØØ DS    ØX
         DC    C'1'
         DC    CL6'VOLSER'
         DC    CL2' '
         DC    CL5'DTYPE'
         DC    CL2' '
         DC    CL5'FLAGS'
         DC    CL2'  '
         DC    CL44'              DATASET NAME                 '
         DC    CL2' '
         DC    CL8'JOB NAME'           CREATING JOB
         DC    CL2' '
         DC    CL8'JOB STEP'           CREATING STEP
         DC    CL2' '
         DC    CL8'PROGRAM'            CREATING PROGRAM
```

```
         DC    CL2' '
         DC    CL8'MGTCLASS'           SMS MANAGEMENT CLASS
         DC    CL1' '
         DC    CL5'LRECL'              LOGICAL RECORD SIZE
         DC    CL1' '
         DC    CL5'BLKSZ'              BLOCK SIZE
         DC    CL1' '
         DC    CL5'BLKCT'              BLOCK COUNT
         DC    CL1' '
         DC    CL7'K-BYTES'            KILOBYTES IN DATASET
         DC    (133-(*-HDR_Ø3ØØ))CL1' ' LET ASM FILL OUT
RDH_Ø3ØØ EQU   *-HDR_Ø3ØØ
MDL_Ø3ØØ DS    ØX
         DC    C' '
MDL_Ø3Ø1 EQU   *-MDL_Ø3ØØ
         DC    CL6' '                  FILE STARTS ON THIS VOLUME
         DC    CL2' '                  FILLER
MDL_Ø3Ø2 EQU   *-MDL_Ø3ØØ
         DC    CL5' '                  DEVICE TYPE
         DC    CL2' '                  FILLER
MDL_Ø3Ø3 EQU   *-MDL_Ø3ØØ
         DC    CL5' '                  FLAG SETTINGS
         DC    CL2' '                  FILLER
MDL_Ø3Ø4 EQU   *-MDL_Ø3ØØ
         DC    CL44' '                 DATASET NAME
         DC    CL2' '                  FILLER
MDL_Ø3Ø5 EQU   *-MDL_Ø3ØØ
         DC    CL8' '                  CREATING JOB
         DC    CL2' '                  FILLER
MDL_Ø3Ø6 EQU   *-MDL_Ø3ØØ
         DC    CL8' '                  CREATING STEP
         DC    CL2' '                  FILLER
MDL_Ø3Ø7 EQU   *-MDL_Ø3ØØ
         DC    CL8' '                  CREATING PROGRAM
         DC    CL2' '                  FILLER
MDL_Ø3Ø8 EQU   *-MDL_Ø3ØØ
         DC    CL8' '                  MANAGEMENT CLASS
         DC    CL1' '                  FILLER
MDL_Ø3Ø9 EQU   *-MDL_Ø3ØØ
         DC    CL5' '                  LOGICAL RECORD SIZE
         DC    CL1' '                  FILLER
MDL_Ø31Ø EQU   *-MDL_Ø3ØØ
         DC    CL5' '                  BLOCK SIZE
         DC    CL1' '                  FILLER
MDL_Ø311 EQU   *-MDL_Ø3ØØ
         DC    CL5' '                  BLOCK COUNT
         DC    CL1' '                  FILLER
MDL_Ø312 EQU   *-MDL_Ø3ØØ
         DC    CL7' '                  SIZE IN KILOBYTES
         DC    (133-(*-MDL_Ø3ØØ))CL1' ' LET ASM FILL OUT
LDM_Ø3ØØ EQU   *-MDL_Ø3ØØ
         TITLE 'CA1REPRT - REPORT Ø4 RECORD LAYOUT'
```

```
HDR_Ø4ØØ DS    ØX
         DC    C'1'
         DC    CL6'VOLSER'
         DC    CL2' '
         DC    CL44'              DATASET NAME                  '
         DC    CL2' '
         DC    CL5'DTYPE'         DEVICE TYPE
         DC    CL2' '
         DC    CL8'JOB NAME'           CREATING JOB
         DC    CL2' '
         DC    CL8'JOB STEP'           CREATING STEP
         DC    CL2' '
         DC    CL8'PROGRAM'            CREATING PROGRAM
         DC    CL2' '
         DC    CL8'MGTCLASS'           CREATING PROGRAM
         DC    CL2' '
         DC    CL5'LRECL'              LOGICAL RECORD SIZE
         DC    CL2' '
         DC    CL5'BLKSZ'              BLOCK SIZE
         DC    CL2' '
         DC    CL5'BLKCT'              BLOCK COUNT
         DC    CL2' '
         DC    CL7'K-BYTES'            KILOBYTES IN DATASET
         DC    (133-(*-HDR_Ø4ØØ))CL1' ' LET ASM FILL OUT
RDH_Ø4ØØ EQU   *-HDR_Ø4ØØ
MDL_Ø4ØØ DS    ØX
         DC    C' '
MDL_Ø4Ø1 EQU   *-MDL_Ø4ØØ
         DC    CL6' '                  FILE STARTS ON THIS VOLUME
         DC    CL2' '                  FILLER
MDL_Ø4Ø2 EQU   *-MDL_Ø4ØØ
         DC    CL44' '                 DATASET NAME
         DC    CL2' '                  FILLER
MDL_Ø4Ø3 EQU   *-MDL_Ø4ØØ
         DC    CL5' '                  DEVICE TYPE
         DC    CL2' '                  FILLER
MDL_Ø4Ø4 EQU   *-MDL_Ø4ØØ
         DC    CL8' '                  CREATING JOB
         DC    CL2' '                  FILLER
MDL_Ø4Ø5 EQU   *-MDL_Ø4ØØ
         DC    CL8' '                  CREATING STEP
         DC    CL2' '                  FILLER
MDL_Ø4Ø6 EQU   *-MDL_Ø4ØØ
         DC    CL8' '                  CREATING PROGRAM
         DC    CL2' '                  FILLER
MDL_Ø4Ø7 EQU   *-MDL_Ø4ØØ
         DC    CL8' '                  SMS MANAGEMENT CLASS
         DC    CL2' '                  FILLER
MDL_Ø4Ø8 EQU   *-MDL_Ø4ØØ
         DC    CL5' '                  LOGICAL RECORD SIZE
         DC    CL2' '                  FILLER
MDL_Ø4Ø9 EQU   *-MDL_Ø4ØØ
```

```
         DC    CL5' '                  BLOCK SIZE
         DC    CL2' '                  FILLER
MDL_Ø41Ø EQU   *-MDL_Ø4ØØ
         DC    CL5' '                  BLOCK COUNT
         DC    CL2' '                  FILLER
MDL_Ø411 EQU   *-MDL_Ø4ØØ
         DC    CL7' '                  SIZE IN KILOBYTES
         DC    CL2' '                  FILLER
         DC    (133-(*-MDL_Ø4ØØ))CL1' ' LET ASM FILL OUT
LDM_Ø4ØØ EQU   *-MDL_Ø4ØØ
         TITLE 'CA1REPRT - REPORT Ø5 RECORD LAYOUT'
HDR_Ø5ØØ DS    ØX
         DC    C'1'
         DC    CL6'VOLSER'
         DC    CL2' '
         DC    CL44'              DATASET NAME                  '
         DC    CL2' '
         DC    CL5'DTYPE'         DEVICE TYPE
         DC    CL2' '
         DC    CL8'PERM R C'           CREATING JOB
         DC    CL2' '
         DC    CL8'PERM W C'           CREATING STEP
         DC    CL2' '
         DC    CL8'PERM R I'           CREATING PROGRAM
         DC    CL2' '
         DC    CL8'PERM W I'           CREATING PROGRAM
         DC    (133-(*-HDR_Ø5ØØ))CL1' ' LET ASM FILL OUT
RDH_Ø5ØØ EQU   *-HDR_Ø5ØØ
MDL_Ø5ØØ DS    ØX
         DC    C' '
MDL_Ø5Ø1 EQU   *-MDL_Ø5ØØ
         DC    CL6' '                  FILE STARTS ON THIS VOLUME
         DC    CL2' '                  FILLER
MDL_Ø5Ø2 EQU   *-MDL_Ø5ØØ
         DC    CL5' '                  DEVICE TYPE
         DC    CL2' '                  FILLER
MDL_Ø5Ø3 EQU   *-MDL_Ø5ØØ
         DC    CL44' '                 DATASET NAME
         DC    CL2' '                  FILLER
MDL_Ø5Ø4 EQU   *-MDL_Ø5ØØ
         DC    CL5' '                  PERM READ SINCE CLEANED
         DC    CL5' '                  FILLER
MDL_Ø5Ø5 EQU   *-MDL_Ø5ØØ
         DC    CL5' '                  PERM WRITE SINCE CLEANED
         DC    CL5' '                  FILLER
MDL_Ø5Ø6 EQU   *-MDL_Ø5ØØ
         DC    CL5' '                  PERM READ SINCE INIT
         DC    CL5' '                  FILLER
MDL_Ø5Ø7 EQU   *-MDL_Ø5ØØ
         DC    CL5' '                  PERM WRITE SINCE INIT
         DC    (133-(*-MDL_Ø5ØØ))CL1' ' LET ASM FILL OUT
LDM_Ø5ØØ EQU   *-MDL_Ø5ØØ
```

```
         TITLE 'CA1REPRT - REPORT Ø4 RECORD LAYOUT'
HDR_ØGØØ DS    ØX
         DC    C'1'
         DC    CL6'VOLSER'
         DC    CL2' '
         DC    CL44'              DATASET NAME                  '
         DC    CL2' '
         DC    CL5'DTYPE'         DEVICE TYPE
         DC    CL2' '
         DC    CL8'JOB NAME'           CREATING JOB
         DC    CL2' '
         DC    CL8'JOB STEP'           CREATING STEP
         DC    CL2' '
         DC    CL8'PROGRAM'            CREATING PROGRAM
         DC    CL2' '
         DC    CL8'MGTCLASS'           CREATING PROGRAM
         DC    CL2' '
         DC    CL5'LRECL'              LOGICAL RECORD SIZE
         DC    CL2' '
         DC    CL5'BLKSZ'              BLOCK SIZE
         DC    CL2' '
         DC    CL5'BLKCT'              BLOCK COUNT
         DC    CL2' '
         DC    CL7'K-BYTES'            KILOBYTES IN DATASET
         DC    (133-(*-HDR_ØGØØ))CL1' ' LET ASM FILL OUT
RDH_ØGØØ EQU   *-HDR_ØGØØ
MDL_ØGØØ DS    ØX
         DC    C' '
MDL_ØG1 EQU   *-MDL_ØGØØ
         DC    CL6' '                  FILE STARTS ON THIS VOLUME
         DC    CL2' '                  FILLER
MDL_ØG2 EQU   *-MDL_ØGØØ
         DC    CL44' '                 DATASET NAME
         DC    CL2' '                  FILLER
MDL_ØG3 EQU   *-MDL_ØGØØ
         DC    CL5' '                  DEVICE TYPE
         DC    CL2' '                  FILLER
MDL_ØG4 EQU   *-MDL_ØGØØ
         DC    CL8' '                  CREATING JOB
         DC    CL2' '                  FILLER
MDL_ØG5 EQU   *-MDL_ØGØØ
         DC    CL8' '                  CREATING STEP
         DC    CL2' '                  FILLER
MDL_ØG6 EQU   *-MDL_ØGØØ
         DC    CL8' '                  CREATING PROGRAM
         DC    CL2' '                  FILLER
MDL_ØG7 EQU   *-MDL_ØGØØ
         DC    CL8' '                  SMS MANAGEMENT CLASS
         DC    CL2' '                  FILLER
MDL_ØG8 EQU   *-MDL_ØGØØ
         DC    CL5' '                  LOGICAL RECORD SIZE
         DC    CL2' '                  FILLER
```

```
MDL_Ø6Ø9 EQU   *-MDL_Ø6ØØ
         DC    CL5' '                  BLOCK SIZE
         DC    CL2' '                  FILLER
MDL_Ø61Ø EQU   *-MDL_Ø6ØØ
         DC    CL5' '                  BLOCK COUNT
         DC    CL2' '                  FILLER
MDL_Ø611 EQU   *-MDL_Ø6ØØ
         DC    CL7' '                  SIZE IN KILOBYTES
         DC    CL2' '                  FILLER
         DC    (133-(*-MDL_Ø6ØØ))CL1' ' LET ASM FILL OUT
LDM_Ø6ØØ EQU   *-MDL_Ø6ØØ
         TITLE 'CA1REPRT - DEFINE THE DCB EXTENSIONS'
         SPACE 1
CON_DCBE DCBE  RMODE31=BUFF,SYNAD=SYN_TMC,EODAD=EOF_CON
TMC_DCBE DCBE  RMODE31=BUFF,SYNAD=SYN_TMC,EODAD=EOF_TMC
MSG_DCBE DCBE  RMODE31=BUFF,SYNAD=SYN_MSG
RPT_Ø1E  DCBE  RMODE31=BUFF,SYNAD=SYN_RPT
RPT_Ø2E  DCBE  RMODE31=BUFF,SYNAD=SYN_RPT
RPT_Ø3E  DCBE  RMODE31=BUFF,SYNAD=SYN_RPT
RPT_Ø4E  DCBE  RMODE31=BUFF,SYNAD=SYN_RPT
RPT_Ø5E  DCBE  RMODE31=BUFF,SYNAD=SYN_RPT
RPT_Ø6E  DCBE  RMODE31=BUFF,SYNAD=SYN_RPT
         TITLE 'CA1REPRT - DEFINE THE DCB CONTROL BLOCKS'
         SPACE 1
TMC      DCB   DDNAME=CA1TMC,DSORG=PS,MACRF=(GL),LRECL=34Ø,           +
               BLKSIZE=34Ø,RECFM=FB,DCBE=TMC_DCBE
CON      DCB   DDNAME=CONTROL,DSORG=PS,MACRF=(GL),LRECL=8Ø,           +
               BLKSIZE=8Ø,RECFM=FB,DCBE=CON_DCBE
         SPACE 1
MSG      DCB   DDNAME=MESSAGES,DSORG=PS,MACRF=PM,LRECL=133,           +
               RECFM=FBA,DCBE=MSG_DCBE
RPT_Ø1D  DCB   DDNAME=RPTØ1,DSORG=PS,MACRF=PM,LRECL=133,             +
               RECFM=FBA,DCBE=RPT_Ø1E
RPT_Ø2D  DCB   DDNAME=RPTØ2,DSORG=PS,MACRF=PM,LRECL=133,             +
               RECFM=FBA,DCBE=RPT_Ø2E
RPT_Ø3D  DCB   DDNAME=RPTØ3,DSORG=PS,MACRF=PM,LRECL=133,             +
               RECFM=FBA,DCBE=RPT_Ø3E
RPT_Ø4D  DCB   DDNAME=RPTØ4,DSORG=PS,MACRF=PM,LRECL=133,             +
               RECFM=FBA,DCBE=RPT_Ø4E
RPT_Ø5D  DCB   DDNAME=RPTØ5,DSORG=PS,MACRF=PM,LRECL=133,             +
               RECFM=FBA,DCBE=RPT_Ø5E
RPT_Ø6D  DCB   DDNAME=RPTØ6,DSORG=PS,MACRF=PM,LRECL=133,             +
               RECFM=FBA,DCBE=RPT_Ø6E
         TITLE 'CA1REPRT - DYNAMIC WORKING STORAGE'
         SPACE 1
         $ESASTG
         SPACE 1
D_WORK   DS    D                       USED FOR DATA CONVERSION
RET_CODE DS    F
RET_ADDR DS    F                       HOLDING AREA FOR ADDRESS
B_ADDR   DS    F                       HOLDING AREA FOR ADDRESS
RPT_ID   DS    XL8                     REPORT ID
```

```
DSN_MASK DS    XL44                    RESERVE SPACE FOR DSN MASK
DSN_MASL DS    H                       LENGTH OF CURRENT MASK
         SPACE 1
CNT_Ø1D  DS    H                       LINE COUNTER FOR PRINT CONTROL
CNT_Ø2D  DS    H                       LINE COUNTER FOR PRINT CONTROL
CNT_Ø3D  DS    H                       LINE COUNTER FOR PRINT CONTROL
CNT_Ø4D  DS    H                       LINE COUNTER FOR PRINT CONTROL
CNT_Ø5D  DS    H                       LINE COUNTER FOR PRINT CONTROL
CNT_Ø6D  DS    H                       LINE COUNTER FOR PRINT CONTROL
         SPACE 1
CON_FLAG DS    XL1                     FILE STATUS FLAG
         DS    XL1                     RESERVED
         DS    XL4                     @(DCB)
TMC_FLAG DS    XL1                     FILE STATUS FLAG
         DS    XL1                     RESERVED
         DS    XL4                     @(DCB)
MSG_FLAG DS    XL1                     FILE STATUS FLAG
         DS    XL1                     RESERVED
         DS    XL4                     @(DCB)
RPT_Ø1F  DS    XL1                     FILE STATUS FLAG
         DS    XL1                     REPORT STATUS FLAG
         DS    XL4                     @(DCB)
RPT_LLL  EQU   *-RPT_Ø1F               LET ASM CALCULATE SIZE
RPT_Ø2F  DS    XL1                     FILE STATUS FLAG
         DS    XL1                     REPORT STATUS FLAG
         DS    XL4                     @(DCB)
RPT_Ø3F  DS    XL1                     FILE STATUS FLAG
         DS    XL1                     REPORT STATUS FLAG
         DS    XL4                     @(DCB)
RPT_Ø4F  DS    XL1                     FILE STATUS FLAG
         DS    XL1                     REPORT STATUS FLAG
         DS    XL4                     @(DCB)
RPT_Ø5F  DS    XL1                     FILE STATUS FLAG
         DS    XL1                     REPORT STATUS FLAG
         DS    XL4                     @(DCB)
RPT_Ø6F  DS    XL1                     FILE STATUS FLAG
         DS    XL1                     REPORT STATUS FLAG
         DS    XL4                     @(DCB)
RPT_EEE  DS    ØXL1
MSG_BUFF DS    ØXL133
MSG_BUFC DS    XL1
MSG_BUFD DS    XL132
MSG_BUFL EQU   *-MSG_BUFF
RPT_BUFF DS    ØXL133
RPT_BUFC DS    XL1
RPT_BUFD DS    XL132
RPT_BUFL EQU   *-RPT_BUFF
         SPACE 1
TRAN_TAB DS    XL256
         TITLE 'CA1REPRT - DYNAMIC FILE ALLOCATION WORK AREA'
         SPACE 1
SVC_99RB DS    ØF                      @(SVC99 REQUEST BLOCK)
```

```
RB99_ØØØ DS    F
RB99_Ø04 DS    F                       LENGTH   XL1
*                                      VERB     XL1
*                                      RESERVED XL2
RB99_Ø08 DS    F                       ZERO
RB99_Ø12 DS    F                       @(TEXT POINTERS)
RB99_Ø16 DS    F                       ZERO
RB99_Ø2Ø DS    F                       ZERO
TP99_ØØØ DS    F                       @(TEXT UNIT)
TP99_Ø04 DS    F                       @(TEXT UNIT)
TP99_Ø08 DS    F                       @(TEXT UNIT)
TP99_Ø12 DS    F                       @(TEXT UNIT)
TP99_Ø16 DS    F                       @(TEXT UNIT)
TP99_Ø2Ø DS    F                       @(TEXT UNIT)
UN99_ØØØ DS    XL(MODEL99L)            TEXT POINTERS
         TITLE 'CA1REPRT - DYNAMIC STORAGE FOR MACROS'
         SPACE 1
WTO_MSG  WTO   'PLACE HOLDER',MF=L
         TITLE 'CA1REPRT - MAP OUT THE CA1 VOLUME RECORD'
         $TMSRLO LIST=YES
         TITLE 'CA1REPRT - MAP OUT THE CA1 DSNB RECORD'
         $DSNBRLO LIST=YES
         TITLE 'CA1REPRT - MAP OUT THE DCB AREA'
         DCBD  DSORG=(QS)
         TITLE 'CA1REPRT - MAP OUT AREAS FOR DYNAMIC ALLOCATE'
         IEFZB4DØ
         IEFZB4D2
         END CA1REPRT
```

$TMSRLO MACRO

```
         MACRO
         $TMSRLO  &LIST=NO
.*********************************************************************
.* THE SOURCE FOR THIS MACRO WAS OBTAINED FROM THE CA-1 SYSTEMS
.* PROGRAMMERS GUIDE, RELEASE 5.2
.* INFORMATION FOR 359Ø MAGSTAR DEVICES WAS ADDED WHERE KNOWN
.*********************************************************************
         AIF   ('&LIST' EQ 'YES').LTMS
         PUSH  PRINT
         PRINT OFF
.LTMS    ANOP
TMSRECLO DSECT RECORD LAYOUT FOR CA1 TMC
RLVOLSER DS    CL6                     VOLUME SERIAL
RLDSN    DS    CL44                    DATASET NAME
RLEXPDT  DS    PL4                     EXPIRATION DATE
RLVOLSEQ DS    XL2                     VOLUME SEQUENCE NUMBER
RLFRSVOL DS    CL6                     FIRST VOLSER OF THE DATASET
RLPRVVOL DS    CL6                     PREVIOUS VOLSER OF THE DATASET
RLNXTVOL DS    CL6                     NEXT VOLSER OF THE DATASET
RL#DSNBS DS    XL2                     NUMBER OF DATASET NAME BLOCKS
```

```
RLADSNB  DS    XL4                ADDRESS AND # OF FIRST DSNB
RLALDSNB DS    XL4                ADDRESS AND # OF LAST DSNB
RLFLAG1  DS    XL1                INTERNAL FLAG 1
RLDFAULT EQU   X'01'              VOLUME ELIGIBLE FOR RDS OVERIDE
RLDELET  EQU   X'02'              VOLUME IN DELETE STATUS
RLSCRTCH EQU   X'04'              VOLUME IN SCRATCH STATUS
RLCLEAN  EQU   X'08'              VOLUME LISTED TO BE CLEANED
RLABEND  EQU   X'10'              VOLUME CLOSED BY ABEND
RLUPDATE EQU   X'20'              VOLUME RECORD UPDATED
RLCLOSED EQU   X'40'              VOLUME CLOSED NORMALLY
RLINTAL  EQU   X'80'              INTERNAL FIELD CHANGED BY USER
RLFLAG2  DS    XL1                INTERNAL FLAG 2
RLETMS   EQU   X'01'              EXPIRED BY CA1, CA11 OR EDM
RLELDATE EQU   X'02'              EXPIRED FROM LDATE CONTROL
RLECYCLE EQU   X'04'              EXPIRED FROM CYCLE CONTROL
RLECATLG EQU   X'08'              EXPIRED FROM CATALOG CONTROL
RLTEMPDS EQU   X'10'              TEMPORARY DATASET
RLREUSE  EQU   X'20'              DATA SET RECREATED
RLOUTPUT EQU   X'40'              VOLUME OPENED FOR OUTPUT
RLCATLOG EQU   X'80'              DATA SET WAS ON MVS CATALOG
RLFLAG3  DS    XL1                INTERNAL FLAG 3
RLFILCPY EQU   X'01'              CREATED BY CA1-COPYCAT
RLULTIF  EQU   X'02'              ADDITIONAL FILES EXIST IN VOLSET
RLDFEXU  EQU   X'04'              DEFAULT EXPDT USED AT OPEN OUT
RLERASE  EQU   X'08'              DATA SET ERASE REQUIRED
RLDYNAM  EQU   X'10'              CONTROLLED BY CA-DYNAMT
RLEDMTAP EQU   X'20'              CONTROLLED BY EXT. DATA MGR.
RLRELEVM EQU   X'40'              TAPE RELEASED BY EXT. VAULT MGR.
RLBADTAP EQU   X'80'              CA9 R+ INDICATES BAD TAPE
RLFLAG4  DS    XL1                INTERNAL FLAG 4
RLNOSTAK EQU   X'01'              NO FURTHER STACKING ALLOWED
RLINUSE  EQU   X'02'              TAPE IS IN USE FOR RTS
RLRS     EQU   X'04'              NON-RESIDENT TAPE
RLISCAT  EQU   X'08'              FILE IS ON OS CATALOG
RLDEGAU  EQU   X'10'              TAPE HAS BEEN DEGAUSED
RLVSR    EQU   X'20'              VAULT SPECIFIC REQUEST
RLACVOLI EQU   X'40'              ACTUAL VOLSER IN USE
RLESMS   EQU   X'80'              TAPE EXPIRED BY SMS MAX RETN.
RLTRTCH  DS    XL1                TAPE RECORDING TECHNIQUE
RLDACON  EQU   X'13'              DATA CONVERSION
RLEPAR   EQU   X'23'              EVEN PARITY
RLEPART  EQU   X'2B'              EVEN PARITY AND TRANSLATION
RLBETRAN EQU   X'38'              BCD-EBCDIC TRANSLATION
RL9TRK   EQU   X'80'              NINE TRACK TAPE
RL18TRK  EQU   X'C0'              3480 CART. TAPE 18 TRACKS
RL36TRK  EQU   X'E0'              3490 CART. TAPE 36 TRACKS
RL36TRK2 EQU   X'E1'              3490E CART. TAPE 36 TRACKS
RLDEN    DS    XL1                RECORDING DENSITY
RL200    EQU   X'03'              200 BPI
RL556    EQU   X'43'              556 BPI
RL800    EQU   X'83'              800 BPI
RL1600   EQU   X'C3'              1600 BPI
```

```
RL6250   EQU   X'D3'                  6250 BPI
RL38000  EQU   X'E3'                  38K BPI CARTRIDGE
RL38KC   EQU   X'E7'                  38K CARTRIDGE COMPACTED
RLMAGS   EQU   X'E8'                  3590 MAGSTAR CARTRIDGE
RLLTYPE  DS    XL1                    LABEL TYPE
RLNL     EQU   X'01'                  NL NO LABEL
RLSL     EQU   X'02'                  SL STANDARD LABEL
RLNSL    EQU   X'04'                  NSL NON-STANDARD LABEL
RLSUL    EQU   X'0A'                  SUL STANDARD USER LABLE
RLBLP    EQU   X'10'                  BLP BYPASS LABEL PROCESS
RLAL1    EQU   X'40'                  AL1 ANSI LABEL V.1
RLAU1    EQU   X'48'                  AU1 ANSI USER LABEL V.1
RLAL3    EQU   X'C0'                  AL3 ANSI LABEL V.3
RLAU3    EQU   X'C8'                  AU3 ANSI USER LABEL V.3
RLRECFM  DS    XL1                    RECORD FORMAT
RLSB     EQU   X'08'                  STANDARD BLOCK
RLBLOCK  EQU   X'10'                  BLOCKED
RLVBAS   EQU   X'30'                  VARIABLE BLOCKED ASCII
RLVAR    EQU   X'40'                  VARIABLE
RLVA     EQU   X'44'                  VARIABLE ANSI
RLVS     EQU   X'48'                  VARIABLE SEQUENTIAL
RLVB     EQU   X'50'                  VARIABLE BLOCKED
RLVBM    EQU   X'52'                  VARIABLE BLOCKED MACHINE
RLVBAN   EQU   X'54'                  VARIABLE BLOCKED ANSI
RLVBS    EQU   X'58'                  VARIABLE BLOCKED SPANNED
RLFIX    EQU   X'80'                  FIXED
RLFM     EQU   X'82'                  FIXED MACHINE
RLFA     EQU   X'84'                  FIXED ANSI
RLFS     EQU   X'88'                  FIXED STANDARD
RLFB     EQU   X'90'                  FIXED BLOCK
RLFBM    EQU   X'92'                  FIXED BLOCK MACHINE
RLFBA    EQU   X'94'                  FIXED BLOCK ANSI
RLFBS    EQU   X'98'                  FIXED BLOCK SEQUENTIAL
RLUNDEF  EQU   X'C0'                  UNDEFINED
RLUNBLK  EQU   X'D0'                  UNBLOCKED
RLLRECL  DS    XL4                    LOGICAL RECORD LENGTH
RLBLKSI  DS    XL4                    MAXIMUM BLOCK SIZE
RLBLKCNT DS    XL4                    DATA SET BLOCK COUNT
RLOUTDAT DS    PL4                    DATE TAPE MARKED OUT OF AREA
RLOUTAR  DS    CL4                    LOCATION ID OF OUT OF AREA TAPE
RLSLOT   DS    XL4                    VAULT SLOT NUMBER
RLCRTDT  DS    PL4                    TAPE CREATION DATE
         DS    XL1                    RESERVED
RLCRTTI  DS    PL3                    TAPE CREATION TIME
RLJOBNM  DS    CL8                    CREATING JOB NAME
RLSTPNAM DS    CL8                    CREATING STEP NAME
RLDDNAME DS    CL8                    CREATING DD NAME
RLCRUNI  DS    XL2                    ADDRESS OF UNIT CREATED ON
RLLASUSD DS    PL4                    DATE TAPE WAS LAST USED
         DS    XL1                    RESERVED
RLLASUST DS    PL3                    TIME TAPE WAS LAST USED
RLLASUSJ DS    CL8                    NAME OF JOB WHO LAST USED
```

```
RLUSUNI  DS    XL2                ADDRESS OF LAST UNIT USED ON
RLACTVLI DS    CL5                ACTUAL VOLUME SERIAL
RLCLNCNT DS    XL1                NUMBER OF TIMES TAPE CLEANED
RLUSECLN DS    XL2                USE COUNT AT LAST CLEANING
RLDATCLN DS    PL4                DATE TAPE LAST CLEANED
RLBTHDT  DS    PL4                DATE THE TAPE WAS FIRST USED
RLUCOUNT DS    XL2                # TIMES TAPE OPENED SINCE BDATE
RLVENDOR DS    CL8                TAPE VENDOR NAME
RLEDMID  DS    CL4                EXTERNAL DATA MANAGER ID
RLTRERRC DS    XL2                TEMP READ ERROR SINCE CLEANED
RLTWERRC DS    XL2                TEMP WRITE ERRORS SINCE CLEANED
RLPRERRC DS    XL2                PERM READ ERRORS SINCE CLEANED
RLPWERRC DS    XL2                PERM WRITE ERRORS SINCE CLEANED
RLTRERRI DS    XL2                TEMP READ ERRORS SINCE INIT
RLTWERRI DS    XL2                TEMP WRITE ERRORS SINCE INIT
RLPRERRI DS    XL2                PERM READ ERRORS SINCE INIT
RLPWERRI DS    XL2                PERM WRITE ERRORS SINCE INIT
RLDSN17  DS    CL17               LAST 17 BYTES OF THE DSN
RLROBTY  DS    XL1                TAPE IN ROBOTIC DEVICE
RLB1INT  DS    XL3                B1 SECURITY INTEGRITY LABEL
RLFLAG5  DS    XL1                INTERNAL FLAG 5
RLSTACK  EQU   X'8Ø'              TAPE HAS BEEN USED BY RTS
RLB1DIS  DS    XL3                B1 SECURITY DISCLOSURE LABEL
RLFLAG6  DS    XL1                INTERNAL FLAG 6
RLSMSMC  DS    CL8                SMS MANAGEMENT CLASS NAME
RLCPGM   DS    CL8                CREATING PROGRAM NAME
RLLPGM   DS    CL8                NAME OF PROGRAM LAST USED TAPE
RLROBID  DS    XL1                ROBOTIC DEVICE INDICATOR
RLACTVL2 DS    XL1                ACTUAL INTERNAL VOLSER
RLUSER   DS    CL5Ø               USER JOB ACCOUNTING AREA
RLVATSA  DS    ØXL24              TIME STAMP AREA
RLVABTCH DS    XL1                ID OF LAST CA1 PGM TO UPDATE
RLVAHOOK DS    XL1                ID OF LAST INTERCEPT TO UPDATE
RLHOOKØØ EQU   X'ØØ'              OPEN NL INPUT-OUTPUT
RLHOOK12 EQU   X'12'              EOV NL OUTPUT
RLHOOK24 EQU   X'24'              EOV NL INPUT
RLHOOKØ8 EQU   X'Ø8'              OPEN SL INPUT
RLHOOK2Ø EQU   X'2Ø'              OPEN SL OUTPUT
RLHOOKØ4 EQU   X'Ø4'              OPEN SL OUTPUT
RLHOOK68 EQU   X'68'              CLOSE INPUT-OUTPUT
RLHOOK64 EQU   X'64'              CLOSE EOV OUTPUT
RLHOOK16 EQU   X'16'              EOV SL OUTPUT
RLHOOK6Ø EQU   X'6Ø'              CLOSE EOV OUTPUT
RLHOOK28 EQU   X'28'              EOV SL INPUT
RLVADATE DS    PL4                DATE OF LAST UPDATE
         DS    XL1                RESERVED
RLVATIME DS    PL3                TIME OF LAST UPDATE
RLVAUSER DS    CL8                ID OF LAST USER TO UPD. RECORD
RLVACPU  DS    CL4                ID OF CPU FOR LAST UPDATE
RLVACODE DS    XL1                AUDIT CODE
RLVAFLG1 DS    XL1                AUDIT FLAG
RL_RECLN EQU   *-TMSRECLO         LET ASM CALC THE LENGTH
```

```
         AIF   ('&LIST' EQ 'YES').LLTMS
         PRINT ON
         POP   PRINT
.LLTMS   ANOP
         MEND
```

$DSNBRLO MACRO

```
         MACRO
         $DSNBRLO &LIST=NO
.*****************************************************************
.* THE SOURCE FOR THIS MACRO WAS OBTAINED FROM THE CA-1 SYSTEMS
.* PROGRAMMERS GUIDE, RELEASE 5.2
.* INFORMATION FOR 3590 MAGSTAR DEVICES WAS ADDED WHERE KNOWN
.*****************************************************************
         AIF   ('&LIST' EQ 'YES').LDSNB
         PUSH  PRINT
         PRINT OFF
.LDSNB   ANOP
DSNBRLO  DSECT RECORD LAYOUT FOR CA1 TMC DSNB RECORD
DSNBID   DS    XL1                    DSNB RECORD IDENTIFIER
DSNBACT  DS    XL1                    DSNB USED INDICATOR
DSNBLBL  EQU   X'40'                  B1 SECURITY LABEL
DSNBACTV EQU   X'80'                  DSNB ACTIVE BIT
DSNBFLG1 DS    XL1                    MISC FLAGS
DSNBDFLT EQU   X'01'                  ELIGIBLE FOR RDS OVERIDE
DSNBWSCA EQU   X'02'                  FILE WAS ON OS CATALOG
DSNBDFXU EQU   X'04'                  DEFAULT EXPIRATION USED
*                                     AT OPEN
DSNBISCA EQU   X'08'                  FILE IS ON OS CATALOG
DSNBABND EQU   X'10'                  FILE WAS CLOSED BY ABEND
*                                     PROCESSING
DSNBECAT EQU   X'20'                  FILE WAS EXPIRED BY CATALOG
*                                     CONTROL
DSNBTMSI EQU   X'40'                  FILE WAS EXPIRED BY TMS
*                                     INTERFACE
DSNBUSRU EQU   X'80'                  DSNB UPDATED BY USER
DSNBFLG2 DS    XL1                    MISC FLAGS
DSNBCURR DS    XL4                    ADDRESS (RELATIVE TO BASE IN
*                                     TMSCTL#2) OF THIS DSNB RECORD
DSNBPREV DS    XL4                    ADDRESS (RELATIVE TO BASE IN
*                                     TMSCTL#2) OR NUMBER OF THE
*                                     PREVIOUS DSNB RECORD
DSNBNEXT DS    XL4                    ADDRESS (RELATIVE TO BASE IN
*                                     TMSCTL#2) OR NUMBER OF NEXT
*                                     DSNB RECORD
DSNBVSN  DS    XL6                    VOLUME SERIAL NUMBER OF THE
*                                     FIRST VOLUME ON WHICH FILE
*                                     2 WAS OPENED
DSNBFVSN DS    XL6                    FILE STARTS ON THIS VOLUME
DSNBFSN  DS    XL2                    FILE SEQUENCE NUMBER
```

```
DSNBDSN  DS    XL44                DATA SET NAME
DSNBEXDT DS    PL4                 EXPIRATION DATE
DSNBCRDT DS    PL4                 CREATION DATE
         DS    XL1                 RESERVED
DSNBCRTM DS    PL3                 CREATION TIME
DSNBCJN  DS    XL8                 CREATING JOB NAME
DSNBCSN  DS    XL8                 CREATING STEP NAME
         DS    XL2                 RESERVED
DSNBLREC DS    XL4                 LOGICAL RECORD LENGTH
DSNBBLKS DS    XL4                 BLOCK SIZE
DSNBBLKC DS    XL4                 BLOCK COUNT
DSNBRFM  DS    XL1                 RECORD FORMAT
DSNBSB   EQU   X'Ø8'               STANDARD BLOCK
DSNBB    EQU   X'1Ø'               BLOCKED
DSNBVBAN EQU   X'3Ø'               VARIABLE BLOCKED ANSI
DSNBV    EQU   X'4Ø'               VARIABLE
DSNBVA   EQU   X'44'               VARIABLE ANSI
DSNBVS   EQU   X'48'               VARIABLE SEQUENTIAL
DSNBVB   EQU   X'5Ø'               VARIABLE BLOCKED
DSNBVBM  EQU   X'52'               VARIABLE BLOCKED MACHINE
DSNBVBAS EQU   X'54'               VARIABLE BLOCKED ANSI
DSNBVBS  EQU   X'58'               VARIABLE BLOCKED SPANNED
DSNBF    EQU   X'8Ø'               FIXED
DSNBFM   EQU   X'82'               FIXED MACHINE
DSNBFA   EQU   X'84'               FIXED ANSI
DSNBFB   EQU   X'9Ø'               FIXED BLOCK
DSNBFBM  EQU   X'92'               FIXED BLOCK MACHINE
DSNBFBA  EQU   X'94'               FIXED BLOCK ANSI
DSNBFBS  EQU   X'98'               FIXED BLOCK SEQUENTIAL
DSNBUD   EQU   X'CØ'               UNDEFINED
DSNBUB   EQU   X'DØ'               UNBLOCKED
DSNBSMSM DS    XL8                 SMS MANAGEMENT CLASS
DSNBCPGM DS    XL8                 CREATING PROGRAM NAME
DSNBUNOS DS    XL13                RESERVED
DSAUTSA  DS    ØXL23               TIME STAMP AREA
DSAUBTCH DS    XL1                 ID OF LAST CA-1 PROGRAM TO
*                                  UPDATE RECORD
DSAUHOOK DS    XL1                 ID OF LAST INTERCEPT TO
*                                  UPDATE RECORD
DSHOOKØØ EQU   X'ØØ'               OPEN NL INPUT-OUTPUT
DSHOOK12 EQU   X'12'               EOV NL OUTPUT
DSHOOK24 EQU   X'24'               EOV NL INPUT
DSHOOKØ8 EQU   X'Ø8'               OPEN SL INPUT
DSHOOK2Ø EQU   X'2Ø'               OPEN SL OUTPUT
DSHOOKØ4 EQU   X'Ø4'               OPEN SL OUTPUT
DSHOOK68 EQU   X'68'               CLOSE INPUT-OUTPUT
DSHOOK64 EQU   X'64'               CLOSE EOV OUTPUT
DSHOOK16 EQU   X'16'               EOV SL OUTPUT
DSHOOK6Ø EQU   X'6Ø'               CLOSE EOV OUTPUT
DSHOOK28 EQU   X'28'               EOV SL INPUT
DSAUDATE DS    PL4                 DATE OF LAST UPDATE
```

```
         DS    XL1                     RESERVED
DSAUTIME DS    PL3                     TIME OF LAST UPDATE
DSAUUSER DS    XL8                     ID OF LAST USER TO UPDATE RECORD
DSAUCPU  DS    XL4                     ID OF CPU FOR LAST UPDATE
DSAUCODE DS    XL1                     AUDIT CODE
DSAUFLG1 DS    XL1                     AUDIT FLAG
DSNBRGHT EQU   X'8Ø'                   RIGHT DSNB INDICATOR - IF THIS
*                                      BIT IS ON, IT IS THE RIGHT DSNB,
*                                      IF OFF, IT IS THE LEFT DSNB
DSNBRLOL EQU   *-DSNBRLO               CLACULATE LENGTH, SHOULD = X'AA'
         AIF   ('&LIST' EQ 'YES').LLDSNB
         PRINT ON
         POP   PRINT
.LLDSNB  ANOP
         MEND
```

$ESAPRO MACRO

```
         MACRO
&LABEL   $ESAPRO &AM=31,&RM=ANY,&MODE=P
.*************************************************************************
.*       THIS MACRO WILL PROVIDE ENTRY LINKAGE AND OPTIONALLY
.*       MULTIPLE BASE REGISTERS.  TO USE THIS MACRO, YOU NEED TO
.*       ALSO USE THE $ESASTG MACRO.  THE $ESASTG DEFINES THE SYMBOL
.*       QLENGTH WHICH OCCURS IN THE CODE THAT &ESAPRO GENERATES.
.*       IF YOU DO NOT CODE ANY OPERANDS, THEN REGISTER 12 WILL BE
.*       USED AS THE BASE.  IF YOU CODE MULTIPLE SYMBOLS, THEN THEY
.*       WILL BE USED AS THE BASE REGISTERS.
.*
.*       EXAMPLES:
.*
.*              SECTNAME $ESAPRO          = REG 12 BASE
.*              SECTNAME $ESAPRO 5        = REG 5 BASE
.*              SECTNAME $ESAPRO R1Ø,R11  = REGS 1Ø AND 11 ARE BASES
.*************************************************************************
*
         LCLA  &AA,&AB,&AC
*
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
RA       EQU   1Ø
R11      EQU   11
```

```
RB       EQU   11
R12      EQU   12
RC       EQU   12
R13      EQU   13
RD       EQU   13
R14      EQU   14
RE       EQU   14
R15      EQU   15
RF       EQU   15
*
FPRØ     EQU   Ø
FPR2     EQU   2
FPR4     EQU   4
FPR6     EQU   6
*
&LABEL   CSECT
&LABEL   AMODE &AM
&LABEL   RMODE &RM
*
         SYSSTATE ASCENV=&MODE         SET THE ENVIRONMENT
*
         B     $$$$EYEC-*(R15)         BRANCH AROUND EYECATCHER
         DC    AL1(($$$$EYEC-*)-1)     EYECATCHER LENGTH
         DC    CL8'&LABEL'             MODULE ID
         DC    CL3' - '
         DC    CL8'&SYSDATE'           ASSEMBLY DATE
         DC    CL3' - '
         DC    CL8'&SYSTIME'           ASSEMBLY TIME
         DC    CL3'   '                FILLER
*
$$$$F1SA DC    CL4'F1SA'               USED FOR STACK OPERATIONS
$$$$4Ø96 DC    F'4Ø96'                 USED TO ADJUST BASE REGS
*
$$$$EYEC DS    ØH
*
         BAKR  R14,Ø                   SAVE GPRS AND ARS ON THE STACK
         AIF   (N'&SYSLIST EQ Ø).USER12
         LAE   &SYSLIST(1),Ø(R15,Ø)    LOAD OUR BASE REG
         USING &LABEL,&SYSLIST(1)      LET THE ASSEMBLER KNOW
         AGO   .GNBASE
.USER12  ANOP
         MNOTE *,'NO BASE REG SPECIFIED, REGISTER 12 USED'
         LAE   R12,Ø(R15,Ø)            LOAD OUR BASE REG
         USING &LABEL,R12              LET THE ASSEMBLER KNOW
         AGO   .STGOB
.GNBASE  ANOP
         AIF   (N'&SYSLIST LE 1).STGOB
&AA      SETA  2
&AC      SETA  4Ø96
.GNBASE1 ANOP
*
```

```
         AIF   (&AA GT N'&SYSLIST).STGOB
&AB      SETA  &AA-1
         LR    &SYSLIST(&AA),&SYSLIST(&AB) GET INITIAL BASE
         A     &SYSLIST(&AA),$$$$4Ø96      ADJUST NEXT BASE
         USING &LABEL+&AC,&SYSLIST(&AA)    LET THE ASSEMBLER KNOW
&AA      SETA  &AA+1
&AC      SETA  &AC+4Ø96
         AGO   .GNBASE1
.STGOB   ANOP
*
         L     RØ,QLENGTH              GET THE DSECT LENGTH
         STORAGE OBTAIN,LENGTH=(RØ),LOC=(RES,ANY)
*
         LR    R15,R1                  GET @(OBTAINED AREA)
         L     R13,QDSECT              GET DISPLACEMENT INTO AREA
         LA    R13,Ø(R13,R15)          GET @(OBTAINED AREA)
         LR    RØ,R13                  SET REG Ø = REG 13
         L     R1,QLENGTH              GET THE LENGTH OF THE AREA
         XR    R15,R15                 CLEAR REG 5
         MVCL  RØ,R14                  INTIALIZE THE AREA
         MVC   4(4,R13),$$$$F1SA       INDICATE STACK USAGE
         USING DSECT,R13               INFORM ASSEMBLER OF BASE
.MEND    ANOP
*
         EREG  R1,R1                   RESTORE REGISTER 1
         MEND
```

$ESAEPI MACRO

```
         MACRO
         $ESAEPI
.*************************************************************************
.*       THIS MACRO WILL PROVIDE EXIT LINKAGE. IT WILL FREE THE
.*       STORAGE AREA THAT WAS ACQUIRED BY THE $ESAPRO MACRO.  YOU
.*       CAN OPTIONALLY PASS IT A RETURN CODE VALUE.  THIS VALUE IS
.*       EITHER THE LABEL OF A FULL WORD IN STORAGE, OR IT IS A REG-
.*       ISTER. AS WITH THE $ESAPRO MACRO, YOU NEED TO USE THE $ESASTG
.*       MACRO.  THE SYMBOL QLENGTH WHICH OCCURS IN THE CODE THAT IS
.*       GENERATED BY THIS MACRO IS DEFINED BY $ESASTG
.*
.*       EXAMPLES:
.*
.*              $ESAEPI          = NO RETURN CODE SPECIFIED
.*              $ESAEPI (R5)     = RETURN CODE IS IN REG 5
.*              $ESAEPI RETCODE  = RETURN CODE IS IN THE FULLWORD AT
.*                                 RETCODE
.*************************************************************************
         AIF   (N'&SYSLIST EQ Ø).STGFRE
         AIF   ('&SYSLIST(1)'(1,1) EQ '(').REGRC
         L     R2,&SYSLIST(1)          GET RETURN CODE VALUE
         AGO   .STGFRE
```

```
.REGRC   ANOP
         LR    R2,&SYSLIST(1,1)        GET RETURN CODE VALUE
.STGFRE  ANOP
*
         L     RØ,QLENGTH              GET THE DSECT LENGTH
         STORAGE RELEASE,LENGTH=(RØ),ADDR=(R13)
*
         AIF   (N'&SYSLIST NE Ø).SETRC
         XR    R15,R15                 CLEAR THE RETURN CODE
         AGO   .MEND
.SETRC   ANOP
         LR    R15,R2                  SET THE RETURN CODE
.MEND    ANOP
         PR                            RETURN TO CALLER
* FOR ADDRESSABILITY PURPOSES
         LTORG
         MEND
```

$ESASTG MACRO

```
         MACRO
         $ESASTG
.*************************************************************************
.*       THIS MACRO IS USED IN CONJUNCTION WITH THE $ESAEPI AND $ESAPRO
.*       MACROS.  IT PROVIDES A Q TYPE ADDRESS CONSTANT WHICH WILL CON-
.*       THE LENGTH OF THE DSECT.  A REGISTER SAVE AREA ID PROVIDED AS
.*       WELL.
.*
.*       EXAMPLES:
.*
.*              $ESASTG
.*      XXX     DC    F          = DEFINE ADDITIONAL STORAGE AREA
.*      YYY     DC    XL255
.*       .       .      .
.*       .       .      .
.*       .       .      .
.*************************************************************************
RC0000   DC    F'Ø'                    USED TO SET RETURN CODES
RC0004   DC    F'4'                    USED TO SET RETURN CODES
RC0008   DC    F'8'                    USED TO SET RETURN CODES
RC000C   DC    F'12'                   USED TO SET RETURN CODES
RC0010   DC    F'16'                   USED TO SET RETURN CODES
QDSECT   DC    Q(DSECT)                DEFINE A QCON
QLENGTH  CXD                           LET ASM CALCULATE THE LENGTH
DSECT    DSECT
         DS    18F                     SET ASIDE REGISTER SAVE AREA
         MEND
```

REXX over IP – Part 1

One of the common themes of software these days is cross-system communication, either across differing platforms or across LPARS/ machines. Having been involved in a number of projects that involve such communication. I had been left with the impression that to achieve such function requires a team of people with varying disciplines. Then as luck would have it, I was given a project that required me to look at accessing basic storage administration information on multiple LPARs and finding a way to consolidate this information. Initially I resorted to traditional methods, namely using NJE to ship jobs around the system and send the data back through sysout. However, while all this was going on I had been reading some IP manuals for interest, and I had noticed the reference to using REXX over IP. So it seemed worth giving this a try to see if I (an old MVS sysprog) could get the hang of using TCP/IP to provide cross-system communication, and, of course, to try to improve the facility I was trying to develop. In the end it proved relatively easy to get a system running, though it has to be said I am still discovering things. Anyway it seemed worthwhile passing on the results of what I've learned, together with the code that was developed, so that other 'old sysprogs' who want to get cross system access can perhaps do so without (hopefully) bothering other technicians.

Working from the manual, my starting point for getting a system going was the IBM-supplied sample REXX routines RSCLIENT and RSSERVER. These routines are supplied in the TCP/IP library that is suffixed SEZAINST.

In order to get these routines working it was necessary to ensure that the member RXSOCKET from the TCP/IP load library was available. Otherwise the SOCKET calls fail (see the supplied code for an example). In my case this was in the link list, so it was available, but you may have to ensure that it is in your log-on STEPLIB for it to work.

Then all I did was try running these two routines on one of our test systems. The server routine RSSERVER was executed as a batch IKJEFT01 while the RSCLIENT command was issued as a simple TSO command. To my surprise it worked first time and the server

returned a random set of REXX source code from the server (which is the basic aim of the IBM sample code). Obviously, while the result is somewhat pointless, it did give me some confidence that it would be possible to get something more useful working. All that was required now was to understand how the code worked and to create something more generally useful. In the end this required me to completely re-build the server and client code from scratch to get around some reliability problems with the examples, and to optimIZe the code (as well as, hopefully, make it clearer how things fit together).

Doing this took quite a bit of research in the manuals, especially since a lot of the terms were unfamiliar to me. In the end I found a number of Web sites that were of great help, for example:

- http://www.s390.ibm.com/products/vse/rexx/REXXBasicServerExample.html.
- http://www.citl.co.uk/MVS.htm#Mainframe Connectivity.
- http://www.tcpip4vse.com/progsamps.html.
- http://www2.hursley.ibm.com/rexxtut/socktut1.htm.

The last in particular was the most useful because it contained an excellent simple diagram to explain the mechanism of making TCPIP calls in a client/server scenario (see the TCP/IP Basics entry at that Web site).

From the point of view of the code supplied in this article, the following are the main things to watch out for when creating your own code, or tailoring what is supplied in this article, for your own site.

The first thing is the 'port' variable (see the source code for LPARANSR below). This is the TCP/IP port that this server will exploit and it is important to ensure that the number specified has not already been associated with another server. To do this either check your TCP/IP parameters (if you know where they are) or ask the person responsible for maintaining them. Then ensure that the port in the code below is set to a 'safe' value. Hopefully you will find, as on my system, that the IBM suggested value is perfectly OK. Note that this value should be the same for both the client and the server code.

The next thing to understand is how to get the system running across two LPARs. This is achieved via the 'GetHostId' requests. Through this, the server obtains an IP address for the host on which it runs. This IP address can then be used in the client to connect to the server. Note that your site will probably have associated names for each host and you will not need the actual IP address, but either form can be used by the client. If you do not know how to find this address for use with the client, simply start the server and see what IP address is displayed in the output. At this point it is possibly also worth pointing out the TSO NETSTAT command. This command will tell you what ports are active on your host and will allow you to check if your server is running OK.

In the end, although it took several days to get things built, and the resulting code may look quite complicated (though it is shorter than the IBM sample code), it is in principle quite simple and mirrors primarily the structure of the diagram on the Web site mentioned earlier. If you watch for the points mentioned above the supplied code should run successfully as-is, and provide a basis for building extra client/server functionality.

Having said that and before showing all the code, the basic client/server function is not all that is included. Once the client/server technique was working I started looking at FTP for transmitting data around, and found that FTP could be used to submit jobs to other machines and LPARs, so I have also included the JCL to start a server from the client location. Note though, it is expected that the server REXX will have already been installed on the server machine if you wish to use the supplied code.

The code supplied consists primarily of two REXX routines – LPARANSR and LPARQUIZ. The first is the server and this routine is written in such a way that it should be easy to add function. In the supplied form it supports three functions – TSO commands, DISKSPACE, and SHUTDOWN. The results for TSO commands (assuming they can be trapped by OUTTRAP) and DISKSPACE are returned line by line to the client. SHUTDOWN is simply a command to allow a client to close down a server remotely. To add function, simply modify the SELECT statement in the code to invoke your own processing routine, and ensure that each line of information is specified as an array with the name msg.wsock.index, where index is the array

number. Then ensure that 'index' is specified on the RETURN statement. The rest of the code will then automatically transfer back the array. Note that the returned lines will have the LPAR name appended at the front of each line to enable easy identification of where the data came from. As a reminder here, the original purpose of the project was to retrieve storage administration information from multiple LPARs (hence the DISKSPACE command). To enable this to work requires an Assembler support routine to create the disk information for the server routine to return. This routine is included (RVOLDATA) below. Please note that this routine is completely independent of the client/server scenario and can easily be exploited by any REXX (see the code for a list of variables created by the code).

The LPARQUIZ routine is the one that makes the request of the server. The command shipped across is of the form user-id, followed by the command as used in the select statement, then by any parameters required by the processing routine in the server. If you look at the description at the start of the routine you will note one other additional required parameter, which is the IP address of the server. If the server is actually on the same LPAR as you are, this can be specified as 'NONE', since the system will be capable of detecting its own host name. To make this clearer, a request for usercatalog information from a server on a host with the name of (say) PRD1 would be of the form: TSO LPARQUIZ PRD1 TSO LISTC UCAT. In other words invoke LPARQUIZ and pass the command TSO LISTC UCAT across to PRD1.

The following elements are included:

- LPARANSR – the server REXX.
- LPARQUIZ – the client REXX.
- FTPSEND – an FTP job to start the server on a host, run the diskspace command and shut down the server 'all-in-on'.
- RVOLDATA – the code for obtaining the disk information.
- SAMPLE XLS – a short sample of the disk information after it has been downloaded into a spreadsheet.

LPARHTML routine is similar to LPARQUIZ, except that instead of

just outputting the data to the screen (or sysout) it will take the received server data and create a table of the information in HTML format, thus enabling you to download your diskspace information into a Web browser. The file created from the data is specified as a 'DS=' parameter on the LPARHTML command after the normal LPARQUIZ information. Note that once the file has been created, if you have an available file server, you can exploit the FTP function to simply 'PUT' this file directly onto that server if you wish.

LPARANSR

```
 /* REXX */
/*********************************************************************/
/*                                                                   */
/* This is the basic server REXX for developing server communication.*/
/* It is called from a client and is passed a string which should be */
/* of the form user-id followed by an action string.                 */
/*                                                                   */
/* If the string DISKSPACE is passed then information               */
/* on the currently online DASD is returned.                         */
/*                                                                   */
/* If the string starts with TSO, then all that follows this will    */
/* be issued as a TSO command and the data trapped and returned to   */
/* the caller (eg the results of a LISTC UCAT).                      */
/*                                                                   */
/* If the string SHUTDOWN is passed then the server will terminate   */
/*                                                                   */
/* If none of the above is passed then the string SERVER ERROR       */
/* followed by the string sent will be returned to the caller.       */
/*                                                                   */
/* The string passed will be parsed based on the assumption that it  */
/* will be in the following form:                                    */
/* userid() label (action)                                           */
/*                                                                   */
/* Where label is the routine to invoke, and action is the argument  */
/* to that label. See the SELECT statement later to see how to       */
/* implement additional function in this server.                     */
/*                                                                   */
/*********************************************************************/
/* */
/* need to trap possible syntax errors in case of incorrect parms */
/* being passed.                                                  */
/* */
SIGNAL ON syntax
/* */
linecount.=Ø
/* initialize control information                                    */
```

```
port = '1952'                /* The port used for the service        */
/* */
/* now obtain the name of the LAPR this server is running on */
/* */
CVTECVT=D2X(C2D(STORAGE(1Ø,4))+14Ø) /* point to cvtsysad */
lparname=STRIP(STORAGE(D2X(C2D(STORAGE(CVTECVT,4))+344),8))
/* Begin setup                                                       */
SAY 'RSSERVER: initialiZing'
/* */
/* a call to socket will return a string which gives an rcode */
/* followed by the unique name for this task (in this case    */
/* RSSERVER) followed by the maximum number of tasks and      */
/* finally the name of the IP started task.                   */
/* */
x= 'SOCKET'('Initialize','RSSERVER')
IF WORD(x,1)¬='Ø' THEN DO
   SAY 'ERROR while initialiZing'
   EXIT
   END
/* */
/* We now need to get the host IP address. This is done with a  */
/* gethostid request. In a similar manner to other requests the */
/* first character returned is a success or failure indicator   */
/* and in this case the second word is the IP address.          */
/* */
ipaddress='SOCKET'('GetHostId')
/* */
IF WORD(ipaddress,1)¬='Ø' THEN DO
   SAY 'ERROR while getting hostid'
   EXIT
   END
/* */
ipaddress=WORD(ipaddress,2)
/* */
SAY 'RSSERVER: initialised: ipaddress='ipaddress 'port='port
/* */
/* obtain a socket id. This is word 2 of the request. */
/* */
sock = 'SOCKET'('Socket')
/* */
IF WORD(sock,1)¬='Ø' THEN DO
   SAY 'ERROR while getting socket'
   EXIT
   END
/* */
sock=WORD(sock,2)
/* */
/* In case IP hasn't cleared itself up by the time the server */
/* restarts, set the reuse option to prevent the server being */
/* unable to start.                                           */
/* */
```

```
x = 'SOCKET'('SetSockOpt',sock,'Sol_Socket','So_REUSEADDR','On')
/* */
/* now its time to issue a bind. Only a single character RC */
/* should be returned this time.                            */
/* */
x='SOCKET'('Bind',sock,'AF_INET' port ipaddress)
/* */
IF x¬=Ø THEN DO
   SAY 'error during af_inet'
   EXIT
   END
/* */
/* now time to listen. */
/* */
x='SOCKET'('Listen',sock)
/* */
IF x¬=Ø THEN DO
   SAY 'error during listen'
   EXIT
   END
/* */
/* now set the io control mode with blocking. */
/* */
x='SOCKET'('Ioctl',sock,'FIONBIO','ON')
/* */
IF x¬=Ø THEN DO
   SAY 'error during set of io control mode'
   EXIT
   END
/* */
x='SOCKET'('Fcntl',sock,'F_SETFL','BLOCKING')
/* */
IF x¬=Ø THEN DO
   SAY 'error during set of io control mode'
   EXIT
   END
/* */
/* Wait for new connections and send lines. The array linecount will */
/* be used to keep track of data sent to each caller.                */
/* */
linecount. =  Ø
/* */
DO FOREVER
/* */
sellist='SOCKET'('SELECT','Write * Read * Exception')
/* */
PARSE UPPER VAR sellist . 'READ' rsock . 'WRITE' wsock . 'EXCEPTION' .
/* */
/* Now receive the information. If the socket id passed is the same */
/* as the one we are listening on, then we need to accept the       */
/* new connection.                                                  */
```

```
/* */
IF rsock¬='' THEN DO
   IF rsock=sock THEN DO
      x = 'SOCKET'('Accept',rsock)
      IF WORD(x,1)\='Ø' THEN DO
         SAY 'error adding another socket'
         EXIT
         END
      ELSE rsock=WORD(x,2)
      END
/* */
   x='SOCKET'('Recv',rsock)
/* */
   PARSE VAR x x . user string
/* */
   IF x¬='Ø' THEN DO
      SAY 'Connection lost'
      x='SOCKET'('Close',rsock)
      END
   ELSE DO
        stringuser.rsock=user
        stringword.rsock=string
        SAY 'User' user 'issued command' string 'at' TIME() DATE('E')
   END
END
/* */
/* Retrieve the command for this socket request and build the    */
/* information in the variable aray msg.wsock.msgnum.            */
/* It is assumed that RESULT will contain the number of lines    */
/* to return to the caller upon return from the subroutine.      */
/* If it doesn't then 1 line to return is assumed.               */
/* This will be passed to the caller as a message with the LPAR  */
/* name at the front of the data.                                */
/* As it is possible that the strings will become joined if the  */
/* network responses are slow, then a break character of X'ØD'   */
/* is used to indicate end of line.                              */
/*                                                               */
/* Lines will be returned one at a time and the linecount for    */
/* the write socket will gradually drop to zero as data leaves   */
/* for the client.                                               */
/* */
IF wsoc¬='' THEN DO
   IF linecount.wsock=Ø THEN DO
      PARSE VAR stringword.wsock command data
      SELECT
        WHEN command='DISKSPACE' THEN CALL diskspace_process
        WHEN command='TSO' THEN CALL tsocmds_process
        WHEN command='SHUTDOWN' THEN SIGNAL shutdown
        OTHERWISE CALL error_process
      END
      IF RESULT='' THEN linecount.wsock=1
      ELSE linecount.wsock=RESULT
```

```
   END
   msgnum=linecount.wsock
   msg=lparname msg.wsock.msgnum||'ØD'x
   x='SOCKET'('Send',wsock,msg)
   IF WORD(x,1)='Ø' THEN DO
      linecount.wsock = linecount.wsock - 1
      DROP msg.wsock.msgnum
      END
   IF WORD(x,1)\='Ø' THEN DO /* send failure - cleanup */
      linecount.wsock=Ø      /* indicate no lines      */
      DO x=1 TO msgnum
         DROP msg.wsock.x    /* release storage */
      END
      DROP stringword.wsock
      DROP stringuser.wsock
   END
   IF linecount.wsock=Ø THEN DO
      x='SOCKET'('Close',wsock)
      END
   END
END
/* */
/* Terminate the server and exit */
/* */
shutdown:
x='SOCKET'('Terminate')
SAY 'RSSERVER: Terminated'
EXIT Ø
/* */
/* ================== The processing subroutines ==================*/
/* */
diskspace_process:
/* */
CALL RVOLDATA
/* */
/* now do the index trick to avoid data being sent back in reverse order
*/
/* */
y=volser.Ø+1
/* */
/* first pass back the title line. */
/* */
msg.wsock.y='Address Volser Free_Extents Free_Cyls Free_Trks Large_Cyl',
    'Large_Trk Index Frag'
DO x=1 TO volser.Ø*1
   y=y-1
   msg.wsock.y=address.x volser.x 1*free_extents.x 1*free_cylinders.x,
   (1*free_tracks.x)+(15*free_cylinders.x) 1*largest_cylinder_extent.x,
   (1*largest_track_extent.x)+(15*largest_cylinder_extent.x),
   index_status.x 1*fragmentation_index.x
END
/* */
```



```
/* Note that as x is incremented it will contain volser.Ø +1 */
/* */
RETURN x
/* */
tsocmds_process:
ADDRESS TSO
CALL OUTTRAP('LINE.')
''data
/* */
/* now do the index frig to avoid data being sent back reverse order */
/* */
y=line.Ø
DO x=1 TO line.Ø
msg.wsock.y=line.x
y=y-1
DROP line.x
END
CALL OUTTRAP('OFF')
RETURN line.Ø
/* */
error_process:
msg.wsock.1='SERVER ERROR' stringword.wsock
RETURN 1
```

LPARQUIZ

```
/* REXX */
/* */
/*********************************************************************/
/*                                                                   */
/* The first part of this REXX checks to see if the socket has been  */
/* left active in error and terminates it. It then issues requests   */
/* to the specified server.                                          */
/*                                                                   */
/* This REXX requires that a string of information is supplied as    */
/* follows:                                                          */
/* WORD 1 - The ipaddress of the host. Use NONE if using the same    */
/*          host as the client.                                      */
/* WORD 2 onwards - the command string to issue to the server.       */
/*                                                                   */
/*********************************************************************/
/* */
x='SOCKET'('SocketSetStatus')
/* */
IF WORD(x,1)='Ø' THEN DO
   x='SOCKET'('Terminate')
   END
/* */
/*********************************************************************/
```

```
/*                                                                   */
/* An example of a client request REXX. This client sends a request  */
/* to the server so that it can carry out an action. This client then*/
/* retrieves the information line-by-line until the connection is    */
/* terminated by the server.                                         */
/*                                                                   */
/* To exploit this client, use as follows:                           */
/* Two parameters can be used. The first is the ipaddress to contact,*/
/* for example PRD1. The second is the data to be sent to the        */
/* server. If both parameters are not present then this REXX will    */
/* EXIT immediately.                                                 */
/* If the ipaddress is set to NONE then a gethostid will be issued   */
/* to get the ipaddress of our host.                                 */
/* If the server detects an error with the request supplied from     */
/* this client, then the string SERVER ERROR is returned from the    */
/* server followed by the command that was sent through from here.   */
/*                                                                   */
/* Note the userid of the caller is also supplied to the server for  */
/* diagnostic purposes.                                              */
/*                                                                   */
/*********************************************************************/
/* */
ip_proc:
/* */
ARG string
/* */
PARSE VAR string ipaddress string
/* */
/* Initialize control information */
/* */
port = '1952'               /* The port used by the server */
/* */
/* Initialise */
/* */
x='SOCKET'('Initialize','RSCLIENT')
/* */
IF WORD(x,1)¬='0' THEN DO
   SAY 'error initializing RSCLIENT'
   EXIT
   END
/* */
IF ipaddress='NONE' THEN DO
  x='SOCKET'('GetHostId')
  IF WORD(x,1)¬='0' THEN DO
     SAY 'error trying to get host id'
     SIGNAL clean_up
     END
  ELSE ipaddress=WORD(x,2)
  END
/* */
/* Initialize for receiving lines sent by the server. */
```

```
/* */
x = 'SOCKET'('Socket')
/* */
IF WORD(x,1)¬='0' THEN DO
   SAY 'error issuing socket'
   SIGNAL clean_up
   END
/* */
/* pick up the client socket id */
/* */
clisock=WORD(x,2)
/* */
/* */
/* now get the host name */
/* */
x='SOCKET'('GetHostName')
/* */
IF WORD(x,1)¬='0' THEN DO
   SAY 'error getting host name'
   SIGNAL clean_up
   END
/* */
hostname = WORD(x,2)
/* */
/* now issue af_inet */
/* */
x='SOCKET'('Connect',clisock,'AF_INET' port ipaddress)
/* */
IF WORD(x,1)¬='0' THEN DO
   SAY 'error issuing af_inet'
   SIGNAL clean_up
   END
/* */
/* now send the information to the server */
/* */
x='SOCKET'('Send',clisock,userid() string)
/* */
IF WORD(x,1)¬='0' THEN DO
   SAY 'error issuing send'
   SIGNAL clean_up
   END
/* */
/* Wait for lines sent by the server */
/* */
DO FOREVER
/* */
/* now read the data. Data is returned as a rc len data field */
/* */
x='SOCKET'('Read',clisock)
/* */
IF WORD(x,1)\='0' THEN DO
```

```
   PARSE VAR x . error
   SAY 'error issuing recv' error
   SIGNAL clean_up
   END
/* */
/* allow for the line being null. Abort the connection if it is. */
/* */
IF WORD(x,2)='Ø' THEN LEAVE
/* */
/* get the actual data */
/* */
PARSE VAR x . . dataline
/* */
/* As the data may have become strung together thanks to slow */
/* networks, the datalines have been prepared by the server   */
/* with a x'Ød' between the lines.                            */
/* */
DO UNTIL INDEX(dataline,'ØD'x)=Ø
   PARSE VAR dataline trueline 'ØD'x dataline
/* */
/* This is the point in the client REXX where the data is returned */
/* as a string and it is possible to insert your own processing.   */
/* */
   SAY trueline
   END
END
/* */
/* Terminate and exit                                              */
/* */
clean_up:
x='SOCKET'('Terminate','RSCLIENT')
RETURN
```

FTPSEND

```
//XXXXFTP  JOB XX,YYY,CLASS=X,MSGCLASS=T,MSGLEVEL=1 ( Your job card
//FTP      EXEC PGM=FTP,REGION=8M
//OUTPUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//INPUT    DD *
PRD1                    ( The host name or IP address for the server to run
on
Userid password         ( Your userid and password for that host
TYPE E
MODE B
SITE LRECL=80 BLOCKSIZE=3120 RECFM=FB
SITE FILETYPE=JES
PUT 'pds.containing.server.job(member)' ( the server job to send from client
LPAR
```

```
QUIT
//A EXEC PGM=IKJEFT01,DYNAMNBR=50,REGION=6M
//STEPLIB DD DSN=your.steplib,DISP=SHR ( steplib containg RVOLDATA code and
RXSOCKET
//*                                          if necessary
//SYSPROC  DD DSN=your.sysproc.containing.client.rexx,DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSTSOUT DD  SYSOUT=*
//SYSOUT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD  *
PROFILE NOPREFIX
LPARHTML PRD1 DISKSPACE DS=your.dsname ( request the creation of HTML file
LPARQUIZ PRD1 SHUTDOWN ( Shut the server down
```

Editor's Note: this article will be continued in the next issue.

Linux for System/390

INTRODUCTION

The Linux operating system is a variant of Unix. The Linux kernel was developed by Linus Torvalds and initially distributed in 1991.

Linux is gaining widespread support, especially in the server market, because it is simpler, more stable, and less costly than many current desktop operating systems, and it is capitalizing on the negative image Microsoft currently has in the software market.

The other attractive feature of Linux is that it is platform-independent and executes on architectures as diverse as Intel, Alpha or Sparc. IBM captured the headlines in August 2000 by successfully running Linux on a wristwatch!

Research by International Data Corporation (IDC) suggests that NT holds a 36% share of the 5.7 million server operating systems shipped on new platforms, while Linux has a 24% penetration and Novell

NetWare has 19%. IDC Research suggests that Linux is deployed in the following ways, 42% Linux systems were running as Web servers, 24% as Web infrastructure (messaging, file and print, cacheing and proxy serving) platforms, and around 10% were running 'enterprise-class' applications, such as commercial databases.

Linux is Open Source software that may be downloaded free of charge. For more information on the Open Source and the applications available visit: http://www.opensource.org/ or http://www.gnu.org.

BACKGROUND

The story of the Linux port to System/390 begins with the work of Linas Vepstas, and his colleagues, who started a port to System/390 called i370 or 'Bigfoot'. Before this port could reach maturity IBM announced its own port, called 'Linux for S/390', which has essentially superseded Bigfoot (http://oss.software.ibm.com/developerworks/opensource/linux390/index.html). Since the IBM port is the one receiving most or all of the active development work today, the Bigfoot project is in hibernation.

However, the Bigfoot port is extremely important because it was designed to run on older IBM hardware, whereas Linux for System/390 only runs on the more recent System/390 boxes. Unfortunately, the Bigfoot code is not advanced enough for general use, and because of the *de facto* dominance of the IBM port it is unlikely ever to be so.

The IBM port emanated from the IBM's laboratories in Germany, from a 'Skunk Works' project; however, the German code is completely incompatible with the 'Bigfoot' code. IBM open sourced all its patches to the stock kernel and the gcc *except* for the device driver for its OSA network adapter hardware.

The methodology for developing the two ports was different. Linux for System/390 was developed internally by IBM and then published (with source code) afterward. This differs from the 'Bigfoot' project, which was a true Open Source project from its inception. Of course, now that it has been released under the GPL, the IBM port is officially Open Source.

Probably the most active mailing list covering Linux on System/390 is hosted by the Marist College (http://www.marist.edu/linuxvm). IBM provides an e-mail contact at linux390@de.ibm.com, where users can send problems specific to System/390 implementation of the kernel, glibc, and the compiler.

THE BUSINESS APPLICATIONS AND BENEFITS

There are many benefits for users who want to deploy Linux on the System/390.

- Many companies already have staff who use Unix or Linux for CAD, databases, scientific computing, etc. Deploying Linux on a System/390 removes the need to retrain these people to use another command shell or menu system on the mainframe.
- Large mainframe-oriented companies that need some selectively-deployed Linux to meet specific needs, such as a DNS server or firewall, can simply run it on Linux within a logical partition or virtual machine. If you need to use Linux as a server, it is logical that you use mainframe hardware as a place to host it.

Using Linux as a VM/ESA guest

There are a number of compelling reasons for running Linux as a guest under VM/ESA. This will certainly extend the lifespan of the VM operating system, although IBM has been trying to migrate VM users to OS/390 for a long time.

- Server consolidation – running Linux on VM will be particularly attractive for users who see the System/390 as the place to centralize and consolidate their growing farms of distributed intranet and Web servers. After all, VM has about 25 years, worth of maturity as a 'hypervisor', and would offer huge flexibility with multiple Linux images. Running tens or hundreds of Linux systems on a single System/390 server offers customers savings in space and personnel required to manage real hardware. Resources can be shared among multiple Linux images running on the same VM/ESA system. These resources include: CPU cycles, memory, storage devices, and network adapters.

- Virtualization – the virtual machine environment is highly flexible and adaptable. New Linux guests can be added to a VM/ESA system quickly and easily without requiring dedicated resources. In the rapid pace of the Web arena this could be crucial. This is useful for replicating servers in addition to giving users a highly flexible test environment.
- System/390 hardware support – Linux guests can transparently take advantage of VM's support for System/390 hardware architecture and RAS features. Linux on System/390 includes a minidisk device driver that can access all DASD types supported by VM/ESA. Data-in-memory performance boosts are offered by VM's exploitation of the System/390 architecture.
- Communications – VM/ESA provides high-performance communication among virtual machines running Linux and other operating systems on the same processor. The underlying technologies enabling high-speed TCP/IP connections are virtual channel-to-channel (CTC) adapter support and VM's IUCV (Inter-User Communication Vehicle).
- Debugging – VM/ESA offers a functionally rich debug environment that is particularly valuable for diagnosing problems in the Linux kernel and device drivers.
- Growth – an effective and simple way to grow Linux workload capacity is to add more Linux guests to a VM/ESA system.

IBM BUSINESS STRATEGY

There are clear business benefits for users in deploying System/390 Linux. But what is IBM's business strategy for deploying Linux on System/390?

Services

Many vendors are cautious about Linux because it is still unclear how successful companies will be at extracting revenue from what is essentially free software. However, IBM is well positioned to make its money from tools, from integration with existing CICS, IMS, and DB2 sub-systems, and from education, training and support.

IBM Global Services will provide the back-end support to distributors SuSE and TurboLinux, and this is no small consideration. Linux may be the most polished and widely debugged OS available, but consolidating hundreds of business-critical Web and intranet servers onto the mainframe will place unprecedented demands on the system. And while the Open Source philosophy will gradually drive other software costs down, there is no shortage of service-based revenue to be had.

Hardware deployment

Linux could be used as a vehicle to shift more boxes. Although IBM now derives a considerable amount of revenue from services, the mainframe group has always been sacred, as much for historical reasons as for the profits to be made on big iron.

Now that Hitachi Data Systems (HDS) has essentially retired from the high-end System/390 market, at least for the next year, IBM is free to concentrate on the concerted attack from the 'alternative mainframe' vendors at the low end who are preventing the System/390 from expanding into the SME (Small to Medium Enterprise) market sector.

The biggest threat in this sector is Sun Microsystems. Sun has essentially stolen the massive e-business Web-server market from right under IBM's nose. If Linux gains significant momentum and acquires a dominant position in the market it could affect Sun's position with Solaris and serve to level the playing field in Unix servers. This would make the choice of hardware more important than the software.

Therefore, System/390 Linux is a considerable threat to Sun, both because it legitimizes the operating system in the large enterprise community and because IBM has such a wide variety of hardware available. Certainly, Sun has not made itself popular in the Open Source community, because of their restrictive licences.

As if to emphasize its interest in hardware, IBM deployed the System/390 Server Feature on Linux in August 2000, which allows users to purchase additional processor capacity to run Linux in a partitioned environment on their system, without incurring additional software costs for the extra processor.

Platform integration

One of the reasons that IBM has decided to adopt Linux could be to tie together all of its platforms. For one thing, Linux could be made to run on every IBM platform.

However, at the moment this is only applicable at the operating system level. There are still problems with application interoperability. True interoperability advantages come from well-known, well-designed, well-understood open standards, adhered to by all comers. The best example of this is the TCP/IP suite of standards (HTTP, SMTP, FTP, etc). The nature of the development model for Linux and related Open Source Software encourages this type of usage of standards, but it should be noted that simply adopting Linux is not a guaranteed solution to interoperability problems.

IBM is the leader in middleware because it has a serious legacy problem – none of its platforms are compatible, and the projects it has launched to integrate the platforms (Systems Application Architecture, Office vision LAN, and the Computer Desktop Environment) have all been stillborn. It is no coincidence that IBM first started out adding Linux hooks to its MQSeries middleware as part of a 'Skunk Works' project.

If IBM uses Linux as its universal operating system, connectivity with other systems would be built in, leaving the proprietary systems vendors such as Sun and Microsoft in a difficult situation.

Cost reductions

Linux development is supported by a very large, world-wide Open Source community of independent coders. There is an extremely active Internet community surrounding the Linux on System/390 ports, as evidenced by the Marist College e-mail list.

Supporting a server and desktop operating system is very expensive. The operating system monopoly held by Microsoft means that the other desktop players do not have sufficient market share to sustain the expense of such support. Therefore, Linux could become the 'Holy Grail' for IBM – low maintenance, open, and almost free.

However, this is entirely dependent on the production of a widely accepted Linux desktop environment. The KDE and GNOME environments are getting there, Helix has only just announced an initial alpha, but there is still a long way to go.

If Linux were adopted as a standardized operating system that could run on all of IBM's hardware, it would represent a considerable saving in terms of training and support.

CONCLUSIONS

The mainframe remains the most scalable and available platform around (even if competition from the alternatives is very fierce). The deployment of Linux on the System/390 represents a win-win situation for users and IBM. OS/390 and VM offers the tools, the scalability, the performance management; Linux brings new applications and potentially vast opportunities.

Linux helps IBM broaden the appeal of the System/390 platform, play down the 'dinosaur' image of its high-end machines, and increase the number of options on offer to its most influential customers.

It also provides further support for the Open Source Software movement, raising the profile of the System/390 in universities, which are currently producing computer science graduates with few mainframe skills, and further standardizing the range of software components that are available across IBM's four principal hardware ranges.

BMC has announced its PATROL Agent and Knowledge Module (KM) for Linux on the System/390 platform, its first management tools for Linux on System/390, and also a pilot programme geared to IBM's Linux for OS/390 pilot scheme.

Via a PATROL Agent and KM for Linux on the System/390, users will be able to evaluate the Linux environment, for an extended period of time, without a licence fee. They will be able to buy the full-functioning software afterwards.

As for the pilot programme, the company says it will support IBM's pricing policies for software running in a dedicated Linux partition by not increasing charges for OS/390 software for the additional capacity.

Customers can qualify for the preview programme when purchasing IBM G5 or G6 engine upgrades or OEM equivalent engine upgrades that are 100% dedicated to running Linux for System/390.

For further information contact:
BMC Software, 2101 City West Boulevard, Houston, TX 77042-2827, USA.
Tel: (713) 918 8800
Fax: (713) 918 8000

BMC Software, Compass House, 207-215 London Road, Camberley, Surrey, GU15 3EY, UK.
Tel: (01276) 24622

http://www.bmc.com

* * *

Sybase has begun shipping Version 12.0 of its MainframeConnect software, which allows information from mainframe sources and LAN datastores to be both moved and accessed.

Version 12 of MainframeConnect supports access to both DRDA/MVS and international character sets and enables access to foreign datastores. Support for SQL server has been added as a source for data replication, as well as enhanced support for access to DB2, Informix and Oracle datastores.

Mainframe connect provides connectivity between client/server databases and mainframe data, as well as access to DB2/MVS data and on-line production applications in CICS, IMS/TM and MVS environments. Production applications in these environments can also act as clients to LAN-based data applications.

For further information contact:
Sybase, 6475 Christie Ave, Emeryville, CA 94608, USA.
Tel: (510) 922 3500
Fax: (510) 922 3210

Sybase United Kingdom, First Floor, 63 St Mary Axe, London, UK
Tel: (0207) 285 4000
Fax : (0207) 285 4005

http://www.sybase.com

* * *