May 1999

In this issue

3  Automated DASD/tape diagram generator
11  Dump restore utility for stacked 3590 tapes
34  An advanced dataset utility
50  RESET command performance group restrictions
58  Assembler instruction trace – part 5
72  MVS news

© Xephon plc 1999
MVS Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: xephon@compuserve.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75067
USA
Telephone: 940 455 7050

Australian office
Xephon/RSM
GPO Box 6258
Halifax Street
Adelaide, SA 5000
Australia
Telephone: 088 223 1391

Editor
Dr 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.

Subscriptions and back-issues
A year’s subscription to MVS Update, comprising twelve monthly issues, costs £310.00 in the UK; $465.00 in the USA and Canada; £316.00 in Europe; £322.00 in Australasia and Japan; and £320.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £27.00 ($39.00) each including postage.

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

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

© Xephon plc 1998. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.
Automated DASD/tape diagram generator

INTRODUCTION

For many years, I have been looking for sources of information that would consolidate information on hardware configurations. A few years ago, IBM added some extensions to the MVS DEVSERV command to display detailed information for DASD as well as tape devices. This is done with the QD and QT options on the DEVSERV command, respectively. By issuing these DEVSERV commands, trapping their output via REXX EXECs, and formatting it into a report, a useful set of configuration reports can be generated. The exact syntax of the DEVSERV commands are:

- for DASD devices
  
  DS QD,TYPE=ALL,DEFINED

- for tape devices
  
  DS QT,TYPE=ALL,DEFINED

In order to use the CONSOLE and CONSPROF commands required by the REXX EXEC in either a batch TMP step or directly in TSO, the user must have RACF (or equivalent) authority to the CONSOLE resource, or the installation must code the TSO/E CONSOLE and CONSPROF exits (IKJCNXAC and IKJCNXCI, respectively), or code the log-on pre-prompt exit (IKJEFLD or IKJEFLD1) to grant CONSOLE command authority. Each REXX EXEC directs its report into separate members of the same PDS; the PDS should be created with a record format of FBA and a logical record length of 133.

I have created two REXX EXECs, called DASDGRID and TAPEGRID, that generate some useful reports on DASD and tape configurations. The EXECs generate DASD and tape diagrams in the 16 device-across orientation, requiring the reports to be printed in landscape mode. The blocks of 16 addresses across correspond to address 0 through F of a string of 16 devices. If a device is not defined, the corresponding block is left entirely blank.
For defined DASD devices, each block is filled in with the device number, volume serial, DASD subsystem-id, hardware device type, and the last five positions of the device serial number. Since DEVSERV returns information for all devices defined, even devices that are set up for sparing, members of dual copy pairs, etc, will be listed. If a device is not on-line, six dashes (-) will be returned as the volume serial, which the REXX EXEC converts to six blanks for readability. An interesting thing I found out when running this was that the EMC 3700 DASD that we were in the process of testing did not return a valid device serial number when queried by DEVSERV processing. In that case, the REXX EXEC prints five full stops (.) as the device serial number.

For defined tape devices, each block is filled in with the device number, hardware device type, and the last five positions of the device serial number. For those who still have 3420-type tape drives (including 3422s), no device serial number is available for them, so again I substitute five fullstops. Additionally, for off-line 3420-type devices, four zeros are returned as the device type; in this case I use the response from the DEVSERV DTYPE column as the device type.

A sample batch job to run both reports and have their output printed is shown below:

```bash
//IKJEFT01 EXEC PGM=IKJEFT01,DYNAMNBR=99
//SYSPROC DD DISP=SHR,DSN=userid.CLIST
//SYSTSPRT DD SYSOUT=* 
//SYSTSIN DD *
%DASDGRID
%TAPEGRID
/*
//IEBGENER EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=* 
//SYSUT1 DD DISP=SHR,DSN=SYSTEMS.CONFIG.TEXT(DASDGRID)
//SYSUT2 DD SYSOUT=* 
//SYSIN DD Dummy,DCB=BLKSIZE=80 
//IEBGENER EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=* 
//SYSUT1 DD DISP=SHR,DSN=SYSTEMS.CONFIG.TEXT(TAPEGRID)
//SYSUT2 DD SYSOUT=* 
//SYSIN DD Dummy,DCB=BLKSIZE=80 
```
DASDGRID EXEC

/**************************** REXX ***************************/
a = ' ' || copies('|',16) || ' ' 
l = length(a) 
b = '+|' || copies('_1',1) 
b = overlay(' ',b,2,1) 
b = overlay(' ',b,1,2) 
c = '+|' || copies('________|',16) 
c = overlay('|',c,1,1) 
t = centre('Data Center DASD Configuration',132) 
header = '1' substr(t,1,118) date() 
block_count = Ø 
lst1 = a 
lst2 = a 
lst3 = a 
lst4 = a 
lst5 = a 
work_addr = 'xxxx' 
call devserv 
o = 1 
line. 
line.o = header 
o = o + 1 
line.o = substr(b,1,1) center(substr(b,2),132) 
do i = 1 to t 
   addr = substr(rec.i,2,4) 
   if work_addr = 'xxxx' then work_addr = addr 
   if substr(work_addr,2,2) ≠ substr(addr,2,2) then 
     = substr(addr,2,2) then 
       do 
         o = o + 1 
         line.o = substr(lst1,1,1) center(substr(lst1,2),132) 
         o = o + 1 
         line.o = substr(lst2,1,1) center(substr(lst2,2),132) 
         o = o + 1 
         line.o = substr(lst5,1,1) center(substr(lst5,2),132) 
         o = o + 1 
         line.o = substr(lst3,1,1) center(substr(lst3,2),132) 
         o = o + 1 
         line.o = substr(lst4,1,1) center(substr(lst4,2),132) 
         o = o + 1 
         line.o = substr(c,1,1) center(substr(c,2),132) 
         lst1 = a 
         lst2 = a 
         lst3 = a 
         lst4 = a 
         lst5 = a 
         work_addr = addr 
         block_count = block_count + 1 
   if block_count = 11 then 
     do
block_count = Ø
line.o = substr(b,1,1) center(substr(b,2),132)
o = o + 1
line.o = header
o = o + 1
line.o = substr(b,1,1) center(substr(b,2),132)
end
offset = x2d(substr(addr,4)) * 8 + 2 + 1
vol = substr(rec.i,7,6)
if vol = '------' then vol = '      '
type = substr(rec.i,22,7)
serial = substr(rec.i,58,5)
ssid = substr(rec.i,37,4)
if serial = 'LID* ' then serial = '.....'
lst1 = overlay(addr,lst1,offset,6)
lst2 = overlay(vol,lst2,offset+1,6)
lst3 = overlay(type,lst3,offset,7)
lst4 = overlay(serial,lst4,offset+2,5)
lst5 = overlay(ssid,lst5,offset+3,4)
end
o = o + 1
line.o = substr(lst1,1,1) center(substr(lst1,2),132)
o = o + 1
line.o = substr(lst2,1,1) center(substr(lst2,2),132)
o = o + 1
line.o = substr(lst5,1,1) center(substr(lst5,2),132)
o = o + 1
line.o = substr(lst3,1,1) center(substr(lst3,2),132)
o = o + 1
line.o = substr(lst4,1,1) center(substr(lst4,2),132)
o = o + 1
line.o = substr(b,1,1) center(substr(b,2),132)
address "TSO"
"ALLOC DD(DASDGRID) DA('SYSTEMS.CONFIG.TEXT(DASDGRID)') SHR REU"
"EXECIO * DISKW DASDGRID (FINIS STEM line."
"FREE DD(DASDGRID)"
exit

/***********************************************************************************/
/*  DEVSERV subroutine: issue DEVSERV, trap and parse response          */
/***********************************************************************************/
devserv:
wait_time = 60                         /*  seconds to wait for reply */
"CONSOLE ACTIVATE"
lastrc = rc
if lastrc ≠ Ø then
  do
    say ""
end
say "*** Unable to activate TSO CONSOLE services!"
say "*** The return code from 'console activate' was:" lastrc
say "*** Attempting to recover..."
"CONSOLE DEACT"
lastrc = rc
say "*** CONSOLE DEACT return code was:" lastrc
"CONSOLE ACTIVATE"
lastrc = rc
if lastrc = Ø then say "*** Recovery successful!"
else
do
say "*** Recovery attempt failed (I issued CONSOLE DEACT)";
"return code was:" lastrc;
say "*** Perhaps you don't have TSO CONSOLE authority?"
exit(16)end
end

"CONSPROF SOLDISPLAY(NO) SOLNUM(1000)"
cart="DS" || time('M')
devserv_cmd="DS QD,TYPE=ALL,DEFINED"
address "TSO"
"CONSOLE SYSCMD("devserv_cmd") CART(""cart""")"
getcode = getmsg("msgs."."SOL",cart,.wait_time)
if getcode ¬= Ø then
do
say "*** GETMSG return code was:" lastrc
"CONSPROF SOLDISPLAY(YES) SOLNUM(1000)"
"CONSOLE DEACTIVATE"
exit
end
address "TSO"
"CONSPROF SOLDISPLAY(YES) SOLNUM(1000)"
"CONSOLE DEACTIVATE"

rec. = ''
rec.Ø = Ø
t = Ø
do i = 1 to msgs.Ø
filt = substr(msgs.i,2,7)
select
  when filt = 'IEE459I' then iterate
  when filt = 'UNIT VO' then iterate
  when substr(filt,1,4) = '****' then iterate
  when substr(filt,1,2) = '' then iterate
  otherwise nop
end
t = t + 1
rec.t = msgs.i
end
rec.Ø = t
return(Ø)
TAPEGRID EXEC

/******************************* REXX ***********************************/
a = ' ' || copies('|       ',16) || '|
1 = length(a)
b = '+' || copies('_',1)
b = overlay(' ',b,2,1)
b = overlay(' ',b,1,2)
c = '+' || copies('_______|',16)
c = overlay('|',c,1,1)
t = center('Data Centre Tape Configuration',132)
header = '1' substr(t,1,118) date()
block_count = Ø
lst1 = a
lst2 = a
lst3 = a
lst4 = a
work_addr = 'xxxx'
call devserv
o = 1
line.=
line.o = header
o = o + 1
line.o = substr(b,1,1) center(substr(b,2),132)
do i = 1 to t
   addr = substr(rec.i,2,4)
   if work_addr = 'xxxx' then work_addr = addr
   if substr(work_addr,2,2) ¬= substr(addr,2,2) then
      do
         o = o + 1
         line.o = substr(lst1,1,1) center(substr(lst1,2),132)
         o = o + 1
         line.o = substr(lst2,1,1) center(substr(lst2,2),132)
         o = o + 1
         line.o = substr(lst3,1,1) center(substr(lst3,2),132)
         o = o + 1
         line.o = substr(lst4,1,1) center(substr(lst4,2),132)
         lst1 = a
         lst2 = a
         lst3 = a
         lst4 = a
         work_addr = addr
         block_count = block_count + 1
         if block_count = 11 then
            do
               block_count = Ø
line.o = substr(b,1,1) center(substr(b,2),132)
o = o + 1
line.o = header
o = o + 1
line.o = substr(b,1,1) center(substr(b,2),132)
end

offset = x2d(substr(addr,4)) * 8 + 2 + 1
type = substr(rec.i,30,7)
if type = '0000' then type = substr(rec.i,7,6)
serial = substr(rec.i,55,5)
if serial = 'ATA- ' then serial = '.....'
lst1 = overlay(addr,lst1,offset,6)
lst3 = overlay(type,lst3,offset,7)
lst4 = overlay(serial,lst4,offset+2,5)
end
o = o + 1
line.o = substr(lst1,1,1) center(substr(lst1,2),132)
o = o + 1
line.o = substr(a,1,1) center(substr(a,2),132)
o = o + 1
line.o = substr(a,1,1) center(substr(a,2),132)
o = o + 1
line.o = substr(lst3,1,1) center(substr(lst3,2),132)
o = o + 1
line.o = substr(lst4,1,1) center(substr(lst4,2),132)
o = o + 1
line.o = substr(b,1,1) center(substr(b,2),132)

address "TSO"
"ALLOC DD(TAPEGRID) DA('SYSTEMS.CONFIG.TEXT(TAPEGRID)') SHR REU"
"EXECIO * DISKW TAPEGRID (FINIS STEM line."
"FREE DD(TAPEGRID)"
exit

="/*********************************************************************/
/ * DEVSERV subroutine: issue DEVSERV, trap and parse response    */
/**************************************************************************/
devserv:
wait_time = 6Ø                         /* seconds to wait for reply */
"CONSOLE ACTIVATE"
lastrc = rc
if lastrc ¬= Ø then
do
say ""
say "*** Unable to activate TSO CONSOLE services!"
say "*** The return code from 'console activate' was:" lastrc
say "*** Attempting to recover..."
"CONSOLE DEACT"
lastrc = rc
say "*** CONSOLE DEACT return code was:" lastrc
"CONSOLE ACTIVATE"
lastrc = rc
if lastrc = 0 then say "*** Recovery successful!"
else
do
say "*** Recovery attempt failed (I issued CONSOLE DEACT)"
"return code was:" lastrc;
say "*** Perhaps you do not have TSO CONSOLE authority?"
exit(16)
end
end

"CONSPROF SOLDISPLAY(NO) SOLNUM(1000)"
cart="DS" || time('M')
devserv_cmd="DS QT,TYPE=ALL,DEFINED"
address "TSO"
"CONSOLE SYSCMD("devserv_cmd") CART(‘cart’)"
getcode = getmsg("msgs.\"SOL\",cart,wait_time)
if getcode ≠ 0 then
do
say "*** GETMSG return code was:" lastrc
"CONSPROF SOLDISPLAY(YES) SOLNUM(1000)"
"CONSOLE DEACTIVATE"
exit
end
address "TSO"
"CONSPROF SOLDISPLAY(YES) SOLNUM(1000)"
"CONSOLE DEACTIVATE"

rec. = ‘’
rec.Ø = Ø
t = Ø
do i = 1 to msgs.Ø
filt = substr(msgs.i,2,7)
select
 when filt = ‘IEE459I’ then iterate
 when filt = ‘UNIT DT’ then iterate
 when substr(filt,1,4) = ‘****’ then iterate
 when substr(filt,1,2) = ‘ ’ then iterate
 otherwise nop
end
t = t + 1
rec.t = msgs.i
end
rec.Ø = t
return(Ø)
Dump restore utility for stacked 3590 tapes

THE PROBLEM

Several sites that we have provided services for are utilizing 3590 Magstar tape technology to reduce the number of tapes needed for full volume dumps. Current Magstar technology will allow for as many as twelve full volume dumps to be stacked onto a single Magstar cartridge. Many shops that utilize DFSMSShsm to facilitate their full volume back-ups were surprised to find that HSM was not initially enabled to auto-stack dumps and exploit the high capacity of Magstar. Enhancements to HSM allowed the user to request the stacking of dumps onto a single Magstar cartridge. The auto stacking was a part of what was needed. No simple way was provided to generate the JCL that would be needed to get the dumps off the Magstar cartridge.

A SOLUTION

The focus of this article is to provide a sample program that can be used to create this restore JCL. Initial analysis of the problem led us to examine the various reports and information that HSM itself maintains and provides. We found that all of the data needed to effect a restore was indeed in HSM, as we would have expected. All of the data was found to be in the Dump Volume Records that are maintained in the HSM Back-up Control Dataset (BCDS). We utilized the DFSMSShsm Diagnosis Reference, LY27-9608-02 that IBM provides for DFSMS 1.3 to obtain and create a mapping DSECT for the DVL record. I have included a copy of that mapping with the sample program. See the $SMSDVL macro at the end of the article.

The program utilizes five datasets. One of the datasets is the HSM BCDS itself, which is a VSAM KSDS file. We also utilized a messages dataset, an audit dataset, a dataset with our input parameters, and an output dataset to write the generated JCL into. Currently the program has the output JCL hardcoded in the literals section. One possible enhancement could be to read in model JCL to be used for the generation process. The BCDS is accessed in sequential mode. It could be accessed in a skip sequential mode to cut down the run time.
The input parameters that were settled on were designed to provide a moderate amount of flexibility. The values that we chose were as follows. DC was used for the dump class the full volume dumps were assigned to. DD is used to specify the dump date in Julian format. We found that some dump operations might run across midnight, so the date would change. To allow for this we used the DO parameter, for dump offset. With it we can specify up to two additional dates from the DD specification. Q1 was used to specify the high-level qualifier for the output datasets. DT was used to specify the device type of the tape device we are working with, which in our case was 3590-1 for Magstar drives. NV is used to specify the number of DASD volumes we expected to process. This value is used to determine the size of a work area we will dynamically acquire. A sample set of input parameters might look as follows:

* This is a comment card
* Specify the dump class
  DC=WEEKLYV
* Specify the date
  DD=98323
* Allow for dump to run one extra day
  DO=+1
* Specify the output high-level qualifier
  Q1=DFHSM
* Specify the tape device, MAGSTAR for us
  DT=3590-1
* Process up to 450 DASD volumes
  NV=450

Sample JCL to execute the utility is shown in the following example:

```plaintext
//MYJOB JOB (acctng),myname,CLASS=*
//STEP0001 exec PGM=GEN3590J
//STEPLIB DD DISP=SHR,DSN=my.step.lib
//SYSUT10 DD DISP=SHR,DSN=my.inputparms
//HSMBCDS DD DISP=SHR,DSN=my.bcds
//SYSUT20 DD DISP=OLD,DSN=my.output.jcl.file
//MESSAGES DD SYSOUT=? ,DCB=(DSORG=PS,LRECL=133,BLKSIZE=0)
//AUDIT DD SYSOUT=? ,DCB=(DSORG=PS,LRECL=133,BLKSIZE=0)
```

**OPERATIONAL ENVIRONMENT**

While we used this specifically for 3590 Magstar devices, it should be feasible to use it for any device that supports auto-stacking out of HSM. The code was developed and tested on an MVS 5.2.2 system running DFSMS/MVS 1.3.
**GEN3590J**

**TITLE 'GEN3590J - GENERATE JCL FOR FULL VOLUME RESTORE'**

* CSECT : GEN3590J
* MODULE : GEN3590J
* AUTHOR : ENTERPRISE DATA TECHNOLOGIES
* DESC : GEN3590J IS A UTILITY WHICH IS EXECUTED TO GENERATE JCL FOR FULL VOLUME RESTORES FROM 3590 CARTRIDGE TAPES. THE FULL VOLUME DUMPS ARE STACKED ONTO THE 3590 CARTRIDGES. THE DFHSM BACKUP CONTROL DATASET IS READ DIRECTLY TO OBTAIN THE NECESSARY DATA TO GENERATE THE JCL.
* MACROS : $ESAPRO $ESAEP $ESASTG OPEN CLOSE DCB DCBD DCBE
* DSECTS : IHADCBD IDARMRCD $SMSLVL
* INPUT : SYSUT1Ø - PARAMETERS USED FOR BCDS READ AND JCL OUT
* HSMBCDS - HSM BACKUP CONTROL DATASET
* OUTPUT : SYSUT2Ø - OUTPUT FILE CONTAINING GENERATED JCL
* MESSAGES - OUTPUT FILE FOR ERRORS AND INFORMATIONAL DATA
* AUDIT - OUTPUT FILE, AUDIT TRAIL FOR JCL GENERATED
* PLIST : NONE
* CALLS : NONE
* NOTES : 31 BIT ADDRESSING USED FOR ALL FILES

**EJECT**

GEN3590J $ESAPRO R12,R11,AM=31,RM=24

**OPEN UP THE MESSAGES FILE**

OPEN (MESSAGES,(OUTPUT)),MODE=31
USING IHADCBD,R1
LA R1,MESSAGES
TM DCBOFLGS,DCBOFOPN
BO MSG_OPEN
DROP R1

**SYNAD CONTROL POINT FOR PHYSICAL ERROR ON THE MESSAGES DATASET**

SYN_MSG DS ØH

**ISSUE A WTO FOR THIS FILE ONLY SINCE WE HAVE NO OTHER WAY TO SEND A MESSAGE, SET THE RETURN CODE AND THEN EXIT THE PROGRAM**

LA R1,WTO_MSG
WTO MF=(E,(1))
MVC RET_CODE,RCØØ1Ø
B EXIT_PG9

**THE MESSAGE DATASET IS OPEN. WE NEED TO SET UP A TRANSLATE TABLE**

MSG_OPEN DS ØH

MVI FLAG_MSG,DCBOFOPN

L R3,DELIM PICK UP THE DELIMITER
LA R4,TRAN_TAB GET @(TRANSLATE TABLE)
STC R3,Ø(R3,R4) PUT THE DELIMITER IN THE TABLE

* OPEN THE FILE THAT CONTAINS THE DIRECTIVES WE WILL USE TO READ THE *
* HSM BACKUP CONTROL DATASET *

OPEN (SYSUT1Ø,(INPUT)),MODE=31

* EXAMINE THE DCB TO MAKE SURE THE FILE HAS BEEN OPENED *

USING IHADCB,R1 TELL THE ASSEMBLER
LA R1,SYSUT1Ø GET @(DCB WE JUST OPENED)
TM DCOFLOGS,DCBOFOPN Q. OPEN SUCCESSFULL?
BO U1Ø_OPEN A. YES
DROP R1

* SYNAD CONTROL POINT FOR PHYSICAL ERROR ON THE SYSUT1Ø DATASET *

SYN_U1Ø DS ØH SYNAD EXIT CODE
MVC RET_CODE,RCØØ1Ø SET THE RETURN CODE
B EXIT_PGM EXIT PROGRAM

* PARM DATASET IS OPEN, OPEN UP OUR AUDIT DATASET *

U1Ø_OPEN DS ØH
OPEN (AUDIT,(OUTPUT)),MODE=31

* EXAMINE THE DCB TO MAKE SURE THE FILE HAS BEEN OPENED *

USING IHADCB,R1 TELL THE ASSEMBLER
LA R1,AUDIT GET @(DCB WE JUST OPENED)
TM DCOFLOGS,DCBOFOPN Q. OPEN SUCCESSFULL?
BO AUD_OPEN A. YES, PROCEED
DROP R1

* SYNAD CONTROL POINT FOR PHYSICAL ERROR ON THE AUDIT DATASET *

SYN_AUD DS ØH SYNAD EXIT CODE
MVC RET_CODE,RCØØ1Ø SET THE RETURN CODE
B EXIT_PGM EXIT PROGRAM

* AUDIT DATASET IS OPEN *

AUD_OPEN DS ØH
MVI FLAG_AUD,DCBOFOPN INDICATE DATASET IS OPEN

* OPEN THE JCL OUTPUT FILE *

OPEN (SYSUT2Ø,(OUTPUT)),MODE=31

* EXAMINE THE DCB TO MAKE SURE THE FILE HAS BEEN OPENED *

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
USING IHADCB,R1               TELL THE ASSEMBLER
LA   R1,SYSU2Ø               GET @(DCB WE JUST OPENED)
TM   DCBOFGLS,DCBOFOPN       Q. OPEN SUCCESSFULL?
BO   LOP_U1Ø                 A. YES, PROCEED
DROP  R1

*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
* SYNAD CONTROL POINT FOR PHYSICAL ERROR ON THE SYSU2Ø DATASET       *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
SYN_U2Ø  DS ØH                      SYNAD EXIT CODE
MVC   RET_CODE,RCØØ1Ø         SET THE RETURN CODE
B     EXIT_PGM                EXIT PROGRAM

*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
* ALL QSAM FILES ARE OPEN, PROCESS THE DIRECTIVES FORM SYSU1Ø DATASET *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
U2Ø_OPEN DS ØH
MVI   FLAG_U2Ø,DCBOFOPN       INDICATE DATASET ID OPEN

*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
* FOLLOWING ARE VALID LINE INPUTS FROM THE DIRECTIVES FILE            *
* CARD POSITION 1...5.....Ø.....5.....Ø.....5.....Ø                        *
*                   *  = COMMENT CARD                                     *
*                   DC = DUMP CLASS                                      *
*                   DD = DUMP DATE, JULIAN                            *
*                   DO = OFFSET FROM BASE DAY, VALID VALUES ARE +Ø, +1, +2*
*                   Q1 = HLQ OF THE OUTPUT DATASET                      *
*                   NV = NUMBER OF VOLUMES TO PROCESS                 *
*                   DT = TAPE DEVICE TYPE                              *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
LOP_U1Ø  DS ØH

*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
* GET A RECORD FROM THE DIRECTIVES FILE. USE LOCATE MODE PROCESSING.  *
* REGISTER 5 WILL BE THE BASE REGISTER FOR THE INPUT RECORD           *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
GET   SYSU1Ø
LR    R5,R1                   POINT TO CURRENT RECORD

*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
* SEE IF IT IS A COMMENT CARD, IF SO WE DO NOT NEED TO DO ANYTHING    *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
CLC   L_STAR,Ø(R5)            Q. COMMENT CARD
BE    LOP_U1Ø                 A. YES, GET NEXT REORD

*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
* SEE IF IT IS A DUMP CLASS DIRECTIVE. IF SO, WE NEED TO ISOLATE      *
* THE DUMP CLASS AND SAVE IT FOR LATER USE                           *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
CLC   L_DC,Ø(R5)              Q. DC CARD?
BNE   NOT_DC                  A. NO
LA   R5,3(,R5)               BUMP THE POINTER
TRT  Ø(77,R5),TRAN_TAB       FIND THE DELIMITER
BC    8,NOT_DT                ERROR IN THE INPUT RECORD
LR    R4,R1                   PICK UP REG 1
SR    R4,R5                   R4 NOW HAS THE LENGTH
BCTR  R4,Ø                    DECREMENT IT BY ONE
STH   R4,LEN_DC               SAVE THE LENGTH
LA    R3,H_DC                 GET @(TARGET AREA)
EX R4, MOVE_PRM
   MOVE THE PARM DATA
B LOP_U1Ø
   GO GET THE NEXT RECORD

* SEE IF IT IS A DUMP DATE DIRECTIVE. IF IT IS WE NEED TO ISOLATE *
* THE DATE INFORMATION AND SAVE IT FOR LATER USE *

NOT_DC DS ØH
CLC L_DD, Ø(R5)
   Q. DD CARD?
BNE NOT_DD
   A. NO
LA R5, 3(.R5)
   BUMP THE POINTER
TRT Ø(77,R5), TRAN_TAB
   FIND THE DELIMITER
BC B NOT_DT
   ERROR IN THE INPUT RECORD
LR R4, R1
   PICK UP REG 1
SR R4, R5
   R4 NOW HAS THE LENGTH
BCTR R4, Ø
   DECREMENT IT BY ONE
STH R4, LEN_DD
   SAVE THE LENGTH
LA R3, H_DD
   GET @(TARGET AREA)
EX R4, MOVE_PRM
   MOVE THE PARM DATA
XR R5, R5
   CLEAR REG 5
IC R5, PACK_L1
   GET TARGET LENGTH
OR R4, R5
   GET SOURCE LENGTH
EX R4, EXC PACK
   PACK UP THE DATE
B LOP_U1Ø
   GO GET THE NEXT RECORD

* SEE IF IT IS A HIGH-LEVEL QUALIFIER DIRECTIVE. IF IT IS WE NEED TO *
* ISOLATE THE HIGH-LEVEL QUALIFIER AND SAVE IT FOR LATER USE *

NOT_DD DS ØH
CLC L_Q1, Ø(R5)
   Q. Q1 CARD?
BNE NOT_Q1
   A. NO
LA R5, 3(.R5)
   BUMP THE POINTER
TRT Ø(77,R5), TRAN_TAB
   FIND THE DELIMITER
BC B NOT_DT
   ERROR IN THE INPUT RECORD
LR R4, R1
   PICK UP REG 1
SR R4, R5
   R4 NOW HAS THE LENGTH
BCTR R4, Ø
   DECREMENT IT BY ONE
STH R4, LEN_Q1
   SAVE THE LENGTH
LA R3, H_Q1
   GET LENGTH OF THE STORAGE AREA
EX R4, MOVE_PRM
   MOVE THE PARM DATA
B LOP_U1Ø
   GO GET THE NEXT RECORD

* SEE IF IT IS A NUMBER OF VOLUMES DIRECTIVE. IF IT IS WE NEED TO *
* ISOLATE IT AND SAVE IT FOR LATER USE *

NOT_Q1 DS ØH
CLC L_NV, Ø(R5)
   Q. NV CARD?
BNE NOT_NV
   A. NO
LA R5, 3(.R5)
   BUMP THE POINTER
TRT Ø(77,R5), TRAN_TAB
   FIND THE DELIMITER
BC B NOT_DT
   ERROR IN THE INPUT RECORD
LR R4, R1
   PICK UP REG 1
SR R4, R5
   R4 NOW HAS THE LENGTH
LA R6, L'H_NV
   GET LENGTH OF THE STORAGE AREA
SR R6,R4                      COMPUTE THE DISPLACEMENT
STH R4,LEN_NV                 SAVE THE LENGTH
LA R3,H_NV                    GET @(TARGET AREA)
LA R3,Ø(R6,R3)                BUMP TARGET LOCATION
EX R4,MOVE_PRM                MOVE THE PARM DATA
PACK PL_NV(8),H_NV(5)         CONVERT IT TO DECIMAL
CVB R4,PL_NV                  MAKE IT BINARY
ST R4,BI_NV                   SAVE IT FOR LATER USE
B LOP_U1Ø                     GO GET THE NEXT RECORD
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*

SEE IF IT IS A DEVICE TYPE DIRECTIVE. IF IT IS WE NEED TO ISOLATE
IT AND SAVE IT FOR LATER USE

NOT_NV   DS ØH
CLC L_DT,Ø(R5)                Q. DT CARD?
BNE NOT_DO                    A. NO
LA R5,3(.R5)                  BUMP THE POINTER
TRT Ø(77,R5),TRAN_TAB         FIND THE DELIMITER
BC B,NOT_DT                   ERROR IN THE INPUT RECORD
LR R4,R1                      PICK UP REG 1
SR R4,R5                      R4 NOW HAS THE LENGTH
BCTR R4,Ø                     DECREMENT IT BY ONE
STH R4,LEN_DT                 SAVE THE LENGTH
LA R3,H_DT                    GET @(TARGET AREA)
EX R4,MOVE_PRM                MOVE THE PARM DATA
B LOP_U1Ø                     GO GET THE NEXT RECORD
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*

SEE IF IT IS A DUMP DATE OFFSET DIRECTIVE. IF IT IS WE NEED TO USE
IT TO CREATE ADDITIONAL DATES FROM THE BASE THAT HAS BEEN SPECIFIED

NOT_DO   DS ØH
CLC L_DO,Ø(R5)                Q. DO CARD
BNE NOT_DO                    A. NO
CLC L_DO₀,3(R5)               Q. ZERO OFFSET ?
BE LOP_U1Ø                    A. YES, NOTHING TO DO
CLC L_DO₁,3(R5)               Q. OFFSET OF 1 DAY ?
BNE NOT_DO₁                   A. NO
MVC PL_DD1,PL_DD              PRIME THE AREA
AP PL_DD1,PACK₁               BUMP IT UP BY A DAY
OI PL_DD1+3,X'ØF'             SET LAST 4 BITS ON
B LOP_U1Ø                     GO GET THE NEXT RECORD
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*

IF THE USER HAS ASKED FOR AN OFFSET OF TWO DAYS FROM THE BASE DATE
THEN WE NEED TO CALCULATE TWO ADDITIONAL DATES

NOT_DO₁ DS ØH
CLC L_DO₂,3(R5)               Q. OFFSET OF 2 DAYS ?
BNE LOP_U1Ø                   A. NO, BYPASS FOR NOW
MVC PL_DD1,PL_DD              PRIME THE AREA
AP PL_DD1,PACK₁               BUMP IT UP BY A DAY
OI PL_DD1+3,X'ØF'             SET LAST 4 BITS ON
MVC PL_DD2,PL_DD1             PRIME THE AREA
AP PL_DD2,PACK₁               BUMP IT UP BY A DAY
OI  PL_D2+3,X'OF'    SET LAST 4 BITS ON
B  LOP_U1Ø

* ONLY WAY WE SHOULD GET HERE IS IF THERE IS A DIRECTIVE ERROR. WE
* WILL BYPASS THE CARD, AND SET A ERROR INDICATOR IN THE RET_CODE
* FIELD

NOT_DT  DS  ØH
MVC  RET_CODE,RCØ10    SET THE RETURN CODE
B  LOP_U1Ø    GO READ ANOTHER RECORD
EOF_U1Ø  DS  ØH

* CLOSE UP THE DIRECTIVES FILE, SET THE FLAG AND CHECK FOR ERRORS

CLOSE (SYSUT1Ø),MODE=31
XC  FLAG_U1Ø,FLAG_U1Ø    INDICATE FILE CLOSED

* CHECK THE RETURN CODE FIELD TO SEE IF WE HAD ANY ERRORS PROCESSING THE
* DIRECTIVES FROM THE SYSUT1Ø FILE

ICM  R5,B'1111',RET_CODE    Q. RETURN CODE SET?
BZ  PARMS_OK    A. NO, PROCEED
PUT  MESSAGES,EM1
B  EXIT_PGM    EXIT THE PROGRAM

* NOW WE WANT TO USE THE NUMBER OF DASD VOLUMES THAT HAS BEEN SPECI-
* FIED AND GETMAIN A STORAGE AREA TO SAVE DATA INTO

PARMS_OK  DS  ØH
LA  R5,W_TEMPL    GET SIZE OF SINGLE ENTRY
ST  R5,W_GESIZE    SAVE IT FOR BXLE
L  R5,BI_NV    GET NUMBER OF ENTRIES
XR  R4,R4    MAKE SURE R4 IS CLEAR
M  R4,W_GESIZE    COMPUTE LENGTH NEEDED
ST  R5,W_GSIZE    SAVE THE SIZE FOR OBTAIN

* ACQUIRE THE NEEDED STORAGE TO SAVE INFORMATION INTO

STORAGE OBTAIN,                   +
   LENGTH=(R5),                   +
   LOC=(ANY,ANY),                 +
   COND=YES

* NOW THAT WE HAVE THE STORAGE AREA WE NEED TO PRIME IT

ST  R1,W_GADDR    SAVE THE ADDRESS
LA  RØ,W_JCL    GET @(TARGET LOCATION)
LA  R14,J1    GET @(SOURCE DATA)
LA  R1,J_LEN    GET THE LENGTH
LA  R15,J_LEN    GET THE LENGTH
MVCL  RØ,R14    MOVE THE MODEL JCL
* THE BACK-UP CONTROL DATASET IS VSAM KSDS. WE NEED TO USE AN ACB AND *
* RPL TO ACCESS THIS DATASET. WE NEED TO PRIME THESE STRUCTURES WITH *
* THE CORRECT INFORMATION BEFORE WE CAN BEGIN TO USE THEM             *

MVC   BCDS_ACB(ACB_MOLL),ACB_MODL PRIME THE ACB
MVC   BCDS_RPL(RPL_MOLL),RPL_MODL PRIME THE RPL

* GET THE APPROPRIATE INFORMATION SO WE CAN MODIFY THE RPL *

LA    R3,BCDS_RPL          GET @(RPL)
LA    R4,BCDS_ACB          GET @(ACB)
LA    R5,R_BUFF            GET @(ADDRESS OF DATA BUFFER)

* MOVE DYNAMIC INFORMATION INTO THE RPL FOR THE BCDS *

MODCB RPL=(R3),
       ACB=(R4),
       AREA=(R5),
       AREALEN=4

* PICK UP THE ADDRESS OF THE ACB AND OPEN IT UP *

LA    R5,BCDS_ACB          PRIME REGISTER 5
OPEN  ((R5)),MODE=31

* IF THE OPEN WAS ERROR FREE, WE WILL BYPASS THE SHOWCB SECTION *

LTR   R15,R15               Q. GOOD OPEN ?
BZ    OPEN_OK                A. YES, PROCEED
LA    R5,BCDS_ACB          GET @(ACB)
LA    R6,ACB_INFO          GET @(INFO FIELD)

* WE ARE ONLY COMING HERE IF WE HAD AN ERROR OPENING THE BCDS *

SHOWCB ACB=(R5),
       AREA=(R6),
       LENGTH=4,
       OBJECT=DATA,
       FIELDS=(ERROR)

* SET A RETURN CODE, AND THEN EXIT THE PROGRAM *

MVC   RET_CODE,RCØØ1Ø       SET THE RETURN CODE
PUT   MESSAGES,EM2
B     EXIT_PGM              EXIT THE PROGRAM

* BCDS IS OPEN AND AVAILABLE TO US FOR PROCESSING *

OPEN_OK  DS ØH
L    R7,W_GADDR            POINT TO STORAGE STRUCTURE
USING  W_TEMP,R7             DECLARE THE BASE
LA    R6,BCDS_RPL          GET @(RPL)
READLOOP DS ØH
GET RPL=(R6)

* DETERMINE IF THE READ WAS GOOD  *

LTR R15,R15 Q. READ SUCCESSFUL?
BZ CHECK_21 A. YES, DETERMINE RECORD TYPE
C R15,FULL_8 Q. RETURN CODE 8?
BNE EXIT_PGM A. NO, EXIT FOR NOW
CLI 15(R6),RPLDVOL Q. EOD OF FILE?
BE CLOSBCDS A. YES, GO CLOSE BCDS
BNZ EXIT_PGM A. NO, EXIT

* THE FOLLOWING CHECKS ARE USED TO SCREEN THE CURRENT RECORD TO SEE  *
* IF IT IS ONE THAT WE NEED TO PROCESS.  *
* REGISTER 2 WILL BE USED AS THE BASE FOR THE BCDS RECORD  *

CHECK_21 DS ØH
L R2,R_BUFF
USING DVL,R2 LET ASSEMBLER KNOW

* SEE IF IT IS RECORD TYPE X'21', A DVL RECORD  *

CLI DVLTYPE,BCDS_21 Q. IS IT A DVL RECORD?
* BRANCH IF NECESSARY.

BL READLOOP A. LOW, GET NEXT RECORD
BH CLOSBCDS A. HIGH, DONE READING

* THIS IS A DVL RECORD. EXAMINE VARIOUS FIELDS TO SEE IF IT IS A  *
* RECORD THAT WE NEED TO PROCESS  *

CHECK_DT DS ØH

* CHECK TO SEE IF THE DEVICE TYPES MATCH  *

LA R3,DVLUNIT POINT TO THE UNIT TYPE FOR DUMP
LA R5,H_DT POINT TO REQUESTED UNIT TYPE
LH R4,LEN_DT GET THE COMPARE LENGTH
EX R4,COMP_VAL Q. UNIT TYPES MATCH?
BNE READLOOP A. NO, GET THE NEXT RECORD

* CHECK TO SEE IF THE DUMP CLASS MATCHES  *

CHECK_DC DS ØH

LA R3,DVLDCLAS POINT TO THE DUMP CLASS
LA R5,H_DC POINT TO REQUESTED DUMP CLASS
LH R4,LEN_DC GET THE COMPARE LENGTH
EX R4,COMP_DC Q. DUMP CLASS MATCH
BNE READLOOP A. NO, GET THE NEXT RECORD

* CHECK TO SEE IF THE DATE IS IN THE RANGE THAT WE ARE LOOKING FOR  *
CHECK_DD DS ØH
  CLC  DVLTSDD,PL_DD   Q. DATE WE ARE LOOKING FOR?
  BE  CHECK_FL        A. YES, CHECK FLAG SETTINGS
  CLC  NULL_VAL,PL_DD1 Q. CHECK FOR DAY +1
  BE  READLOOP        A. NO, GET THE NEXT RECORD
  CLC  DVLTSDD,PL_DD1 Q. DATE WE ARE LOOKING FOR?
  BE  CHECK_FL        A. YES, CHECK THE FLAG SETTINGS
  CLC  NULL_VAL,PL_DD2 Q. CHECK FOR DAY +2
  BE  READLOOP        A. NO, GET THE NEXT RECORD
  CLC  DVLTSDD,PL_DD2 Q. DATE WE ARE LOOKING FOR?
BNE  READLOOP        A. NO, GET THE NEXT RECORD

*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*

* CHECK VARIOUS BIT SETTINGS TO MAKE SURE THAT THE DVL RECORD IS ONE *
* THAT WE WANT TO TRY AND PROCESS. LOOK IN THE LITERAL POOL AREA TO   *
* SEE WHAT BIT SETTINGS WE ARE INTERESTED IN. THERE ARE THREE OF THEM *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*

CHECK_FL DS ØH
  CLC  FLAG_CHK,DVLFLAGS       Q. APPROPRIATE FLAG BITS ON
  BNE  READLOOP                A. NO, GET THE NEXT RECORD

*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*

* WE WANT TO TRY TO PROCESS THIS RECORD. PICK UP THE FILE SEQUENCE    *
* NUMBER, AND SET UP TO GET THE DASD INFORMATION OUT OF THE RECORD.   *
* REGISTER 9 WILL BE USED A BASE INTO A SPECIFIC AREA OF THE DVL      *
* RECORD                                                              *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*

LA  R8,FSEQ_VAL             POINT TO FILE SEQUENCE VALUES
LA  R9,DVLDGNKY             POINT TO THE DASD INFO IN DVL

NEXT_VOL DS ØH
  MVC  T_VOLSER,DVLVSN       GET THE TAPE VOLSER
  DROP R2                    DROP OFF THE BASE
  USING DVLDGNKY,R9          DECLARE THE BASE
  MVC  D_VOLSER,DVLVSN       GET THE DASD VOLSER
  MVC  T_FSEQ,Ø(R8)          MOVE IN THE FILE SEQUENCE #
  MVC  D_DATE,DVLTSDD        GET THE DUMP DATE
  MVC  D_TIME,DVLTSDT        GET THE DUMP TIME
  A  R7,W_GESIZE             BUMP THE POINTER
  LA  R8,2(,R8)              POINT TO NEXT FILE SEQ VALUE
  LA  R9,DVL_RESV-DVLDGNKY(,R9) POINT TO NEXT DASD VOL AREA
  CLC  DVLTSDD,PL_DD         Q. ANOTHER VOLUME THERE?
  BE  NEXT_VOL                A. YES, GO PROCESS THE ENTRY
  DROP R9
  B  READLOOP                 GET THE NEXT RECORD

*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*

* ALL DVL RECORDS HAVE BEEN PROCESSED. CLOSE UP THE BCDS. SET UP      *
* REGISTER 7 TO BE THE BASE FOR THE STORAGE STRUCTURE WHERE WE HAVE   *
* PLACED THE INFORMATION THAT WE OBTAINED FROM THE BCDS               *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*

CLOSBCDS DS ØH
  S   R7,W_GESIZE             ADJUST POINTER TO LAST ENTRY
  ST  R7,W_GADDRL             SAVE IT
  LA  R5,BCDS_ACB             PRIME REGISTER 5
  CLOSE ((R5)),MODE=31
  XC  FLAG_HSM,FLAG_HSM       INDICATE BCDS IS CLOSED
* WE ARE NOW READY TO PRIME VARIOUS FIELDS IN THE JCL CARDS WITH THE *
* INFORMATION THAT WILL BE CONSTANT ACROSS ALL OF THE VOLUMES         *

LA R3,S_CARD61              POINT TO STEP CARD 6
LH R4,LEN_Q1                GET THE LENGTH OF HLQ
LA R5,H_Q1                  POINT TO THE HLQ
EX R4,MOVE_PRM              PUT IT IN THE OUTPUT CARD
LA R3,1(R4,R3)              BUMP R3 INTO THE RECORD
MVC Ø(L' L_DMP,R3),L_DMP    MOVE IN LITERAL INFO
LA R3,5,(R3)                BUMP R3 INTO THE RECORD
LH R4,LEN_DC                GET LENGTH OF DUMP CLASS
LA R5,H_DC                  POINT TO THE DUMP CLASS
EX R4,MOVE_PRM              MOVE IT IN
LA R3,1(R4,R3)              BUMP R3 INTO THE RECORD
MVC Ø(L' L_DMP1,R3),L_DMP1  MOVE IN LITERAL INFO
ST R3,CARD6_@1              SAVE FIRST VARIABLE TARGET
LA R3,6,(R3)                BUMP R3 INTO RECORD
MVC Ø(L' L_DMP2,R3),L_DMP2  MOVE IN LITERAL INFO
ST R3,CARD6_@2              SAVE SECOND VARIABLE TARGET
LA R3,5,(R3)                BUMP R3 INTO RECORD
MVC Ø(L' L_DMP3,R3),L_DMP3  MOVE IN LITERAL INFO
ST R3,CARD6_@3              SAVE THIRD VARIABLE TARGET

* PRIME A COUPLE OF COUNTERS, AND SET REGISTERS 7 8 9 FOR A BXLE LOOP *

ZAP P_TVOL,PACK_Ø          CLEAR OUT COUNTER
ZAP P_DVOL,PACK_Ø          CLEAR OUT COUNTER
MVC AUDIT_R,AUDIT_M        PRIME THE AUDIT RECORD
LM R7,R9,W_GADDR           PRIME REGS FOR BXLE LOOP

* LOOP THROUGH HERE TO OUTPUT INFORMATION FOR EACH DASD VOLUME *

PUT_LOOP DS ØH

* PLUG INFORMATION INTO THE JOBCARD, AND OUTPUT THE JOBCARDS. FOR *
* OUR JOBS, WE HAVE TWO JOBCARDS *

CLC C_VOL,T_VOLSER        Q. STILL ON THE SAME VOLSER?
BE SAME_VS                A. YES, STEP CARDS ONLY
MVC C_VOL,T_VOLSER        SAVE THE CURRENT VOLSER
MVC J_CARD11,T_VOLSER     PUT VOLSER IN THE JOBCARD
PUT SYSUT20,J_CARD1       PUT VOLSER IN THE JOBCARD
PUT SYSUT20,J_CARD2
AP P_TVOL,PACK_1          INCREMENT TAPE COUNTER
SAME_VS DS ØH

* PLUG INFORMATION INTO THE STEP CARDS, AND THEN OUTPUT THEM *

MVC S_CARD11,D_VOLSER       MOVE IN DISK VOLSER
MVC S_CARD31,D_VOLSER       MOVE IN DISK VOLSER
MVC S_CARD41,T_FSEQ         MOVE IN FILE SEQUENCE
MVC S_CARD51,T_VOLSER       MOVE IN TAPE VOLSER
LM R3,R5,CARD6 @1          PICK UP TARGETS IN CARD 6
MVC Ø(L'D_VOLSER,R3),D_VOLSER MOVE IN THE DISK VOLSER
UNPK CH_DD(7),D_DATE        UNPACK THE DATE
UNPK CH_DT(7),D_TIME        UNPACK THE TIME
MVC Ø(L'CH_DDYY,R4),CH_DDYY MOVE DATE INFO TO CARD 6
MVC Ø(2,R5),CH_DTSS         MOVE TIME VALUES TO CARD 6
MVC 2(2,R5),CH_DTMH         MOVE TIME VALUES TO CARD 6
MVC 4(2,R5),CH_DTHH         MOVE TIME VALUES TO CARD 6
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
* STEP CARDS ARE READY, WRITE THEM OUT TO THE JCL FILE                *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
PUT SYSUT2Ø,S_CARD1
PUT SYSUT2Ø,S_CARD2
PUT SYSUT2Ø,S_CARD3
PUT SYSUT2Ø,S_CARD4
PUT SYSUT2Ø,S_CARD5
PUT SYSUT2Ø,S_CARD6
PUT SYSUT2Ø,S_CARD7
PUT SYSUT2Ø,S_CARD8
PUT SYSUT2Ø,S_CARD9
AP P_DVOL,PACK_1           INCREMENT DASD COUNTER
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
* FILL IN THE AUDIT RECORD, AND OUTPUT IT TO MESSAGES FILE            *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
MVC AUDIT_TV,T_VOLSER       MOVE IN THE TAPE VOLSER
MVC AUDIT_DV,D_VOLSER       MOVE IN THE DASD VOLSER
MVC AUDIT_FS,T_FSEQ         MOVE IN THE FILE SEQ
PUT AUDIT,AUDIT_R
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
* PROCESS ALL ENTRIES FROM THE TABLE. BXLE LOOP DOES THIS FOR US      *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
BXLE R7,R8,PUT_LOOP
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
* ALL ENTRIES PROCESSED. CLOSE UP THE OUTPUT JCL FILE                 *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
CLOSE (SYSUT2Ø),MODE=31
XC FLAG_U2Ø,FLAG_U2Ø       INDICATE FILE CLOSED
MVC AUDIT_R,AUDIT_T         MOVE IN THE MODEL RECORD
UNPK AUDIT_TT(5),P_TVOL(3)   UNPACK THE TOTAL TAPE VOLS
UNPK AUDIT_TD(5),P_DVOL(3)   UNPACK THE TOTAL DASD VOLS
OI AUDIT_TT+4,X'FØ'         FIX THE SIGN
OI AUDIT_TD+4,X'FØ'         FIX THE SIGN
PUT AUDIT,AUDIT_R
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
* COMMON EXIT, FREE UP STORAGE AND CHECK ALL FILES, AND CLOSE THOSE   *
* THAT MAY STILL BE OPEN                                                *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
EXIT_PGM DS ØH

L R5,W_GSIZE      PICK UP THE CHUNK SIZE
ICM R6,B'1111',W_GADDR        Q. AREA ADDRESS PRESENT
B2 EXIT_NST       A. NO NEED FOR RELEASE
STORAGE RELEASE, +
LENGTH=(R5), +
ADDR=(R6), +
COND=YES
EXIT_NST DS ØH
TM FLAG_U1Ø,DCBOFOPN Q. U1Ø FILE OPEN
BNO EXIT_U1Ø      A. NO, BYPASS THE CLOSE
CLOSE (SYSUT1Ø),MODE=31
EXIT_U1Ø DS ØH
TM FLAG_U2Ø,DCBOFOPN Q. U2Ø FILE OPEN
BNO EXIT_U2Ø      A. NO, BYPASS THE CLOSE
CLOSE (SYSUT2Ø),MODE=31
EXIT_U2Ø DS ØH
TM FLAG_HSM,DCBOFOPN Q. BCDS STILL OPEN
BNO EXIT_AUD      A. NO, BYPASS THE CLOSE
LA R5,BCDS_ACB    PRIME REGISTER 5
CLOSE ((R5)),MODE=31
EXIT_AUD DS ØH
TM FLAG_AUD,DCBOFOPN Q. AUDIT FILE STILL OPEN
BNO EXIT_PG9      A. NO, BYPASS THE CLOSE
CLOSE (AUDIT),MODE=31
CLOSE (MESSAGES),MODE=31
EXIT_PG9 DS ØH
$ESAEPI RET_CODE
TITLE 'GEN359ØJ - LITERAL POOL'
BCDS_21 EQU X'21'       USED TO TEST FOR TYPE X'21'
FULL_8 DC F'8'          USED FOR VSAM RETURN CODE TEST
DELIM DS ØF
DC XL4'ØØØØØØ4Ø'       USED TO PRIME TRANSLATE TABLE
NULL_VAL DC XL4'ØØØØØØØØ' USED TO TEST DATA FIELDS
*
* THE FOLLOWING ARE TARGETS OF EXECUTE INSTRUCTIONS
* 
MOVE_PRM MVC Ø(*-*,,R3),Ø(R5)          TARGET OF AN EXECUTE
COMP_VAL CLC Ø(*-*,,R3),Ø(R5)          TARGET OF AN EXECUTE
EXC_PACK PACK PL_DD(*-*) ,H_DD(*-*)    TARGET OF AN EXECUTE
*
* THE FOLLOWING IS USED TO CONSTRUCT LENGTH FOR EXC_PACK INSTRUCTION
PACK_L1 DC AL1((L'PL_DD-1)*16)         LENGTH OF THE TARGET AREA - 1
*
* USED TO CONSTRUCT TEST PATTERN TO SEE IF REORD SHOULD BE PROCESSED
FLAG_CHK DC AL1(DVLFWRIT+DVLFVALD+DVLFTSED)
*
* USED TO CONSTRUCT FILE SEQUENCE INFORMATION
FSEQ_VAL DC CL3Ø'010203040506070809101112131415'
PACK_Ø DC PL4'0'             USED TO ZERO COUNTERS
PACK_1 DC PL4'1'             USED FOR DATE MANIPULATION
*
* ERROR MESSAGES THAT WE MAY ISSUE
*
EM1 DC CL133'INCORRECT OR BAD PARAMETER INFORMATION SUPPLIED'
EM2 DC CL133'ERROR ENCOUNTERED OPENING THE HSM BCDS'

* MODEL JOB AND STEP CARDS THAT WE USE FOR THE RESTORE JCL *

J1 DC CL8Ø'//R??????  JOB (ACCT),DFDSS-RESTORE,CLASS=B,'
J2 DC CL8Ø'//MSGCLASS=Q,PRTY=Ø3,TYPRUN=HOLD'
S1 DC CL8Ø'//R?????? EXEC  PGM=ADRDSSU,REGION=8M'
S2 DC CL8Ø'//SYSPRINT DD SYSOUT=*
S3 DC CL8Ø'//DASDOUT DD UNIT=339Ø,DISP=OLD,Vol=SER=??????'
S4 DC CL8Ø'//TAPEIN DD DISP=OLD,UNIT=CART359Ø,LABEL=(??,SL),'
S5 DC CL8Ø'//Vol=(,RETAIN,SER=??????),'
S6 DC CL8Ø'//DSN=
S7 DC CL8Ø'//SYSPRINT DD *
S8 DC CL8Ø' RESTORE FULL INDDNAME(TAPEIN) OUTDDNAME(DASDOUT) COPYVOLID'
S9 DC CL8Ø'/'
J_LEN EQU *-J1 LET ASSEMBLR CALCULATE LENGTH

* SIMPLE LAYOUT FOR THE AUDIT RECORDS *

AUDIT_M DS ØXL133
DC CL1'
AUDIT_M1 DC CL12'TAPE VOLUME='
DC CL6'
DC CL1'
AUDIT_M2 DC CL12'DASD VOLUME='
DC CL6'
DC CL1'
AUDIT_M3 DC CL14'FILE SEQUENCE='
DC CL2'
DC (133-(*)-AUDIT_M)CL1' ' FILL IT OUT
AUDIT_T DS ØXL133
DC CL1'
AUDIT_T1 DC CL23'NUMBER OF TAPE VOLUMES='
DC CL5'
DC CL1'
AUDIT_T2 DC CL23'NUMBER OF DASD VOLUMES='
DC CL5'
DC (133-(*)-AUDIT_T)CL1' ' FILL IT OUT
L_DC DC CL3'DC=' DUMP CLASS SPECIFICATION
L_DD DC CL3'DD=' DUMP DATE, JULIAN FORMAT
L_DO DC CL3'DO=' DUMP OFFSET VALUE
L_DO_Ø DC CL2'+Ø' DUMP OFFSET OF ZERO DAYS
L_DO_1 DC CL2'+1' DUMP OFFSET OF ONE DAY
L_DO_2 DC CL2'+2' DUMP OFFSET OF TWO DAYS
L_Q1 DC CL3'01=' HLQ OF THE OUTPUT DATASET
L_NV DC CL3'NV=' NUMBER OF DASD VOLUMES TO HANDLE
L_DT DC CL3'DT=' TAPE DEVICE TYPE
L_STAR DC CL1'*' COMMENT CARD
L_DMP DC CL5'.DMP.'
L_DMP1 DC CL2'.V'
L_DMP2 DC CL2'.D'
L_DMP3 DC CL2'.'T'

* DBC EXTENDED CONTROL BLOCKS

*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*

U1Ø_DCBE DCBE RMODE31=BUFF,SYNAD=SYN_U1Ø,EODAD=EOF_U1Ø
U2Ø_DCBE DCBE RMODE31=BUFF,SYNAD=SYN_U2Ø
MSG_DCBE DCBE RMODE31=BUFF,SYNAD=SYN_MSG
AUD_DCBE DCBE RMODE31=BUFF,SYNAD=SYN_AUD

* DBC AREA FOR ALL OF THE FILES WE USED

*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*

SYSUT1Ø DCB DDNAME=SYSUT1Ø,MACRF=(GL),DSORG=PS,LRECL=8Ø, +
         DCBE=U1Ø_DCBE
SYSUT2Ø DCB DDNAME=SYSUT2Ø,MACRF=(PM),DSORG=PS,LRECL=8Ø, +
         DCBE=U2Ø_DCBE
MESSAGES DCB DDNAME=MESSAGES,MACRF=(PM),DSORG=PS,LRECL=133, +
         DCBE=MSG_DCBE
AUDIT DCB DDNAME=AUDIT,MACRF=(PM),DSORG=PS,LRECL=133, +
         DCBE=AUD_DCBE

* ACB AND THE RPL FOR THE BCDS

*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*

ACB_MODL ACB AM=VSAM, +
           DDNAME=HSMBCDS, +
           MACRF=(IN,SEQ), +
           RMODE31=ALL
ACB_MOLL EQU *-ACB_MODL
RPL_MODL RPL AM=VSAM, +
           ACB=(*-*), +
           AREA=(*-*), +
           OPTCD=LOC
RPL_MOLL EQU *-RPL_MODL

* WTO WE WILL USE IF WE CAN'T GET THE MESSAGES FILE OPENED

*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*

WTO_MSG WTO 'UNABLE TO OPEN MESSAGES FILE, GEN359ØJ TERMINATING', +
          ROUTCDE=(2), +
          MCSFLAG=(HRDCPY), +
          DESC=(6), +
          MF=L
$ESASTG

* DYNAMIC STORAGE AREA, REGISTER 13 IS THE BASE FOR THIS AREA

*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*

RET_CODE DS F USED FOR RETURN CODE
LEN_DC DS H LENGTH, DUMP CLASS SPECIFICATION
LEN_DD DS H LENGTH, DUMP DATE, JULIAN FORMAT
LEN_Q1 DS H LENGTH, HLQ OF THE OUTPUT DS
LEN_DT DS H LENGTH, TAPE DEVICE TYPE
LEN_NV DS H LENGTH, NUMBER OF DASD VOLUMES
C_VOL DS XL6 USED FOR VOLSER COMPARE
H_DC DS XL8 DUMP CLASS SPECIFICATION
H_DD   DS  XL8       DUMP DATE, JULIAN FORMAT
H_QI   DS  XL8       HLQ OF THE OUTPUT DATASET
H_DT   DS  XL8       TAPE DEVICE TYPE
H_NV   DS  XL5       NUMBER OF DASD VOLUMES
          DS  ØD       FORCE ALIGNMENT
PL_NV   DS  PL8      NUMBER OF DASD VOLUMES PACKED
B1_NV   DS  F        NUMBER OF DASD VOLUMES BINARY
PL_DD  DS  PL4       DUMP DATE IN PACKED FORMAT
PL_DD1 DS  PL4       DUMP DATE IN PACKED FORMAT
PL_DD2 DS  PL4       DUMP DATE IN PACKED FORMAT
CH_DD  DS  ØXL7     
          DS  XL2
CH_DDYY DS  XL5      
CH_DT   DS  ØXL7     
CH_DTHH DS  XL2      
CH_DTMM DS  XL2      
CH_DTSS DS  XL2      
          DS  XL1
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
* FLAG BYTES FOR ALL OF THE FILES THAT WE USE                         *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
FLAG_U1Ø DS  XL1    FLAG BYTE, INDICATES FILE STATUS
FLAG_U2Ø DS  XL1    FLAG BYTE, INDICATES FILE STATUS
FLAG_MSG DS  XL1    FLAG BYTE, INDICATES FILE STATUS
FLAG_HSM DS  XL1    FLAG BYTE, INDICATES FILE STATUS
FLAG_AUD DS  XL1    FLAG BYTE, INDICATES FILE STATUS
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
* COUNTERS FOR THE AUDIT INFORMATION                                  *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
P_TVOL  DS  PL3      COUNTER FOR TAPE VOLUMES
P_DVOL  DS  PL3      COUNTER FOR DASD VOLUMES
*
W_GSIZE DS  A       SIZE OF THE WORK AREA
W_GADDR DS  A       ADDRESS OF THE WORK AREA
W_GESIZE DS  A      SIZE OF AN INDIVIDUAL ENTRY
W_GADDRL DS  A      ADDRESS OF LAST ENTRY
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
* FOLLOWING ARE THE ADDRESSES OF THE OUTPUT FIELDS. THESE HAVE TO      *
* BE DETERMINED AT RUNTIME DUE TO THE VARIABLE NATURE OF SOME OF THE   *
* FIELDS IN THE DATASET NAMES                                         *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
CARD6_@1 DS  A       @(DASD VOLUME FIELD)
CARD6_@2 DS  A       @(JULIAN DATE FIELD)
CARD6_@3 DS  A       @(TIME STAMP FIELD)
*
TRAN_TAB DS  256XL1   TRANSLATE TABLE
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
* JOB AND STEP CARDS THAT WERE BUILT FROM THE MODELS. WE USE THESE     *
* FOR EACH DASD VOLUME THAT WAS OBTAINED FROM THE BCDS                 *
*——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+——+*
W_JCL   DS  (J_LEN)XL1 WORKING JCL AREA
          ORG W_JCL    ORG BACK
J_CARD1 DS  XL3      FILLER
J_CARD11 DS XL6                   TAPE VOLSER IN JOB NAME
   DS *(8Ø-(*)-J_CARD1))XL1      FILLER
J_CARD2 DS XL80                  SECOND JOB CARD
S_CARD1 DS XL03                  STEP CARD
S_CARD11 DS XL06                 LET ASSEMBLER FILL IT OUT
   DS *(8Ø-(*)-S_CARD1))XL1
S_CARD2 DS XL80
S_CARD3 DS XL42
S_CARD31 DS XL06                 LET ASSEMBLER FILL IT OUT
   DS *(8Ø-(*)-S_CARD3))XL1
S_CARD4 DS XL45
S_CARD41 DS XL02
   DS *(8Ø-(*)-S_CARD4))XL1
S_CARD5 DS XL22
S_CARD51 DS XL06                 LET ASSEMBLER FILL IT OUT
   DS *(8Ø-(*)-S_CARD5))XL1
S_CARD6 DS XL9                   STEPCARD 6
S_CARD61 DS XL1                  LET ASSEMBLER FILL IT OUT
   DS *(8Ø-(*)-S_CARD6))XL1
S_CARD7 DS XL80                  STEP CARD 7
S_CARD8 DS XL80                  STEP CARD 8
S_CARD9 DS XL80                  STEP CARD 9
L_CHECK EQU (*-W_JCL)-J_LEN      USE AS LENGTH CHECK SHOULD BE Ø

* AUDIT RECORDS THAT WE WILL OUTPUT                        *

AUDIT_R DS 0XL133                 AUDIT RECORD
   DS XL1                     FILLER
   DS *(L'AUDIT_M1)XL1        SPACE IT OUT
AUDIT_TV DS XL6                  PLACE FOR THE TAPE VOLUME
   DS XL1                     FILLER
   DS *(L'AUDIT_M2)XL1        SPACE IT OUT
AUDIT_DV DS XL6                  PLACE FOR THE DASD VOLUME
   DS XL1                     FILLER
   DS *(L'AUDIT_M3)XL1        SPACE IT OUT
AUDIT_FS DS XL2                  FILE SEQUENCE
   DS *(133-(*)-AUDIT_R))XL1  LET ASSEMBLER FILL IT IN
ORG AUDIT_R
   DS 0XL133                 AUDIT RECORD
   DS XL1                     FILLER
   DS *(L'AUDIT_T1)XL1        SPACE IT OUT
AUDIT_TT DS XL5                  PLACE FOR THE TAPE VOLUME
   DS XL1                     FILLER
   DS *(L'AUDIT_T2)XL1        SPACE IT OUT
AUDIT_TD DS XL5                  PLACE FOR THE DASD VOLUME
   DS *(133-(*)-AUDIT_R))XL1  LET ASSEMBLER FILL IT IN
ORG
   DS 0F
BCDS_ACB DS (ACB_MOLL)XL1
   DS 0F
BCDS_RPL DS (RPL_MOLL)XL1
R_BUFF DS A
ACB_INFO DS A

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
TITLE 'GEN359ØJ - MAP OUT THE WORKING STORAGE STRUCTURE'
W_TEMP   DSECT                         WORKING TEMPLATE
T_VOLSER DS   XL6                     TAPE VOLSER
T_FSEQ   DS    XL2                     FILE SEQUENCE NUMBER
D_VOLSER DS    XL6                     DASD VOLSER
D_DATE   DS    XL4                     DATE IT WAS DUMPED
D_TIME   DS    XL4                     TIME IT WAS DUMPED
W_TEMPL  EQU   *-W_TEMP                LET ASM CALCULATE THE SIZE

TITLE 'GEN359ØJ - MAP OUT THE BCDS DVL RECORD'
$SMSDVL LIST=Yes

TITLE 'GEN359ØJ - MAP OUT THE VSAM RETURN-REASON CODES'
IDARMRCD

TITLE 'GEN359ØJ - MAP OUT THE DCB AREA'
DCBD DSORG=(QS)

END   GEN359ØJ                IDENTIFY END OF PROGRAM

$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 $$$$EYEC-*(R15) BRANCH AROUND EYECATCHER
DC AL1($$$$EYEC-*)-1 EYECATCHER LENGTH
DC CL8'&LABEL' MODULE ID
DC CL3' - '
DC CL8'&SYSDATE' ASSEMBLY DATE
DC CL3' - '
DC CL8'&SYSTIME' ASSEMBLY TIME
DC CL3' ' FILLER
$$$$F1SA DC CL4'F1SA' USED FOR STACK OPERATIONS
$$$$4Ø96 DC F'4Ø96' USED TO ADJUST BASE REGS
$$$$EYEC DS ØH
BAKR R14,Ø SAVE GPRS AND ARS ON THE STACK
AIF (N'&SYSLIST EQ Ø).USER12
LAE &SYSLIST(1),Ø(R15,Ø) LOAD OUR BASE REG
USING &LABEL,&SYSLIST(1) LET THE ASSEMBLER KNOW
AGO .GNBASE
.USER12 ANOP
MNOTE *, 'NO BASE REG SPECIFIED, REGISTER 12 USED'
LAE R12,Ø(R15,Ø) LOAD OUR BASE REG
USING &LABEL,R12 LET THE ASSEMBLER KNOW
AGO .STGOB
.GNBASE ANOP
AIF (N'&SYSLIST LE 1).STGOB
&AA SETA 2
&AC SETA 4Ø96
.GNBASE1 ANOP
AIF (&AA GT N'&SYSLIST).STGOB
&AB SETA &AA-1
LR &SYSLIST(&AA),&SYSLIST(&AB) GET INITIAL BASE
A &SYSLIST(&AA),$$$$4Ø96 ADJUST NEXT BASE
USING &LABEL+&AC,&SYSLIST(&AA) LET THE ASSEMBLER KNOW
&AA SETA &AA+1
&AC SETA &AC+4Ø96
AGO .GNBASE1
.STGOB ANOP
L RØ,QLENGTH GET THE DSECT LENGTH
STORAGE OBTAIN, LENGTH=(R0), LOC=(RES, ANY)
LR    R15, R1                  GET @(OBTAINED AREA)
L     R13, QDSECT              GET DISPLACEMENT INTO AREA
LA    R13, 0(R13, R15)          GET @(OBTAINED AREA)
LR    R0, R13                  SET REG 0 = REG 13
L     R1, QLENGTH              GET THE LENGTH OF THE AREA
XR    R15, R15                 CLEAR REG 5
MVCL  R0, R14                  INITIALIZE THE AREA
MVC   4(4, R13), $$$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     R0, QLENGTH              GET THE DSECT LENGTH
STORAGE RELEASE, LENGTH=(R0), 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:
  
  XXX DC F = DEFINE ADDITIONAL STORAGE AREA
  YYY DC XL255

**********************************************************************

RCØØØ DC F'Ø'                    USED TO SET RETURN CODES
RCØØØ4 DC F'4'                    USED TO SET RETURN CODES
RCØØØ8 DC F'8'                    USED TO SET RETURN CODES
RCØØØC DC F'12'                   USED TO SET RETURN CODES
RCØØ1Ø 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

$SMSDVL MACRO

MACRO
$SMSDVL &LIST=NO

**********************************************************************
* MAP OUT THE SMS DVL RECORD
* MAPPING INFORMATION OBTAINED FROM LY27-96Ø8-Ø2
* DFHSMSHSM DIAGNOSIS REFERENCE VERSION 1 RELEASE 3

**********************************************************************

AIF ('&LIST' EQ 'YES').LDVL
PUSH PRINT
PRINT OFF
.LDVL ANOP
DVL DSECT
DVLKEY DS ØXL44 DUMP VOLUME RECORD KEY
DVLTYP DS XL1 DVL RECORD TYPE X'21'
DVLVSN DS XL6 DUMP VOLUME SERIAL
    DS XL37 RESERVED
DVLHDR DS ØXL2 DVL HEADER INFO
DVLLEN DS XL2 DVL RECORD LENGTH, SUM OF
    DVLKEY+DVLHDR+DVLDATA
DVLETYP DS XL1 SAME AS DVLTYP
    DS XL1 RESERVED
DVLTSLU DS XL8 TIME STAMP IN MICROSECONDS
* FORMAT THAT INDICATES WHEN THE DVL RECORD WAS LAST UPDATED
* DVLTSCR DS XL8 TIME STAMP IN MICROSECONDS
* FORMAT THAT INDICATES WHEN THE DVL RECORD WAS CREATED
* DVLDATA DS 0XL110 DATA PORTION OF THE DUMP RECORD
DVLUUNIT DS XL8 UNIT NAME SPECIFIED FOR VOLUME
DVLFLAGS DS XL1 FLAG BYTES
DVLFWRIT EQU B'10000000' DFSMSHSM HAS WRITTEN TO VOL
DVLFUSED EQU B'01000000' VOLUME CURRENTLY IN USE
DVLFUNAV EQU B'00100000' VOL UNAVAILABLE FOR OUTPUT
DVLFVALID EQU B'00010000' VOL PART OF VALID DUMP COPY
DVLFPSW EQU B'00001000' VOL IS PASSWORD PROTECTED
DVLFSEW EQU B'00000100' VOL IS EXPIRATION DATE PROTECTED IN HEADER LABEL
* DVLFSTRF EQU B'00000001' VOL IS RACF PROTECTED
DVLFURAC EQU B'00000001' VOL HAD ALREADY BEEN ADDED TO RACF WHEN DFSMSHSM USED IT
* DVLFFLG2 DS XL1 SECOND FLAG BYTE
DVLFUASN EQU B'10000000' ADDVOLED
DVLFVEXT EQU B'01000000' CALL TAPE VOL EXIT AT DELVOL
DVLFUCUS EQU B'00100000' VOL CONTROL UNIT CONTAINS COMPACTED DATA
* DS XL2 RESERVED
DVLUCBTY DS XL4 UCB DEVICE TYPE FOR VOLUME
DVLVOLSQ DS XL2 VOLUME SEQUENCE NUMBER,
* SIGNIFYING A VOLUME'S RELATIVE POSITION WITHIN A SET OF VOLS THAT CONSTITUTE A DUMP COPY
* DVLDEN DS XL1 VOLUMES RECORDING DENSITY
DVLDCLASS DS XL8 DUMP CLASS NAME
DVLEXPDT DS PL4 DUMP COPY EXPIRATION DATE
DVLDGNKY DS 0XL14
DVLSVSN DS XL6 SOURCE VOLUME DUMP WAS CREATED FROM
* DVLTSDT DS PL4 TIME STAMP WHEN DUMP WRITTEN
DVLTSDD DS PL4 DATE STAMP WHEN DUMP WRITTEN
DVLDID DS XL1
DVLSDEVT DS 0XL4 SOURCE VOL DEVICE OPTIONS
DS XL2
DVLSDEVC DS 0XL2
DVLSDEV DS XL1 SOURCE VOL DEVICE TYPE
DS XL1 SOURCE VOL DEVICE CODE
DVL_RESV DS XL60 RESERVED
DVL_LEN EQU *-DVL LET ASSEMBLER CALCULATE LENGTH
AIF ('&LIST' EQ 'YES').LLDVL PRINT ON
POP PRINT
.LLDVL ANOP
MEND

Enterprise Data Technologies (USA) © Xephon 1999

An advanced dataset utility

THE PROBLEM
There is no easy method for reallocating an existing partitioned or sequential dataset and changing the attributes or allocating a second dataset with the same attributes as the first dataset and optionally copying the contents.

All of us have been in situations where a dataset is too small and needs to be reallocated with a larger size or more directory blocks. Or, we have wanted to allocate a new dataset with identical or near identical characteristics to an existing dataset and copy the contents. In both of these situations, this has only been possible by flipping between several ISPF panels or running a batch job in order to accomplish the task.

A SOLUTION
In order to address this problem, we have developed a new ISPF panel modelled after option 3.2 that is called option 3.22, Dataset Utility Plus. In the 3.22 panel, a user is allowed to enlarge an existing dataset by reallocation, or allocate and copy a new dataset with similar attributes to an existing dataset. Both of these functions can be easily accomplished in one panel. If desired, after placing this dialog’s associated members in the relevant libraries, choosing option 3.22 will start the Dataset Utility Plus dialog from the existing ISPF panel ISRUTIL. The utility selection panel is shown below:

```
22,'CMD(%REALCL)'
```
The first time the panel is invoked it will appear as the example shown in Figure 1. All panel fields will initially be blank except the primary and secondary dataset names, which will be restored from previous uses of the panel via the ISPF profile. All remaining data information fields on the bottom half of the screen will be dynamically filled on the next execution of the panel. The DATASET INFO subtitle in the middle of the screen will contain either PRIMARY or SECONDARY, depending which dataset information is currently being displayed.

In order to display dataset information, leave the ‘OPTION===>’ field blank, fill in the primary dataset name, and press ‘enter’. The bottom half of the panel will then be filled in, as shown in Figure 2.
To allocate a new dataset with attributes similar to an existing dataset, type an ‘A’ in the OPTION field and then indicate whether the contents of the existing file are to be copied by entering a ‘Y’ in the next input field. An ‘N’ is the default. Next, enter the primary and secondary dataset names. Optionally, highlighted fields associated with the dataset can be changed. Then press ‘enter’. A message of successful allocation will be displayed in the upper-right corner and the dataset information for the secondary dataset will be displayed, as seen in the example in Figure 3.

If you want to reallocate an existing dataset, then put an ‘R’ in the OPTION field, and fill in the primary dataset name. Dataset attributes can optionally be changed by replacing the highlighted fields associated with the dataset. Press ‘enter’. A successful reallocation message will be displayed in the upper-right corner and the new dataset attributes will be displayed, as shown in Figure 4.

Figure 2: Inserting data into the panel
Any of the above described options may be executed as many times as desired without exiting from the panel. To exit from the panel, press PF3 (END).

CLIST REALCL is the driver for the panel. There are several notes of interest to consider:

- The CLIST determines the dataset organization of the primary dataset and calls IEBCOPY or IEBGENER accordingly.
- The primary dataset is allocated in OLD mode. If the dataset is busy, the message DATASET IN USE is issued. In our installation, we changed the CLIST to allow system programmers to allocate the dataset in SHR mode, which allows a dataset to be copied even if it is in use. However, the ‘Reallocate’ option will issue the message DATASET IN USE.
• If the secondary dataset volume is full, then the message ALLOCATION UNSUCCESSFUL is issued, or PROCESS UNSUCCESSFUL if the copy has already started and additional extents cannot be obtained.

---

**Figure 4: Successful reallocation**

**REALCL**

```
REALCL

CLIST REALCL .
PROC Ø DEBUG(NO)
IF &DEBUG = YES THEN CONTROL NOFLUSH MAIN LIST CONLIST SYMLIST MSG
    ELSE CONTROL NOFLUSH MAIN NOLIST NOCONLIST NOSYMLIST NOMSG

/* ————————————————————————————————————————————————————————————————— */
SET &STATE = 1        /* USED FOR LOOP MANAGEMENT                */
SET &BASE =           /* PRIMARY  DSN NAME                       */
SET &NEWDS = KUKU     /* SECONDARY DSN NAME                      */
SET &MSG =            /* INIT MESSAGE                            */
SET L = &STR((        /* LEFT PARENTHESIS                        */
SET R = )             /* RIGHT PARENTHESIS                       */
SET &V = N            /* V= Y OR N (WITH COPY MEMBERS OR WITHOUT)*/
SET &ZALVOL =         /*                                         */
```

---

**Figure 4: Successful reallocation**

---

**REALCL**

```
REALCL

CLIST REALCL .
PROC Ø DEBUG(NO)
IF &DEBUG = YES THEN CONTROL NOFLUSH MAIN LIST CONLIST SYMLIST MSG
    ELSE CONTROL NOFLUSH MAIN NOLIST NOCONLIST NOSYMLIST NOMSG

/* ————————————————————————————————————————————————————————————————— */
SET &STATE = 1        /* USED FOR LOOP MANAGEMENT                */
SET &BASE =           /* PRIMARY  DSN NAME                       */
SET &NEWDS = KUKU     /* SECONDARY DSN NAME                      */
SET &MSG =            /* INIT MESSAGE                            */
SET L = &STR((        /* LEFT PARENTHESIS                        */
SET R = )             /* RIGHT PARENTHESIS                       */
SET &V = N            /* V= Y OR N (WITH COPY MEMBERS OR WITHOUT)*/
SET &ZALVOL =         /*                                         */
```
SET &ZALSPC = /* */
SET &ISPC = /* */
SET &PRIM = /* */
SET &SECON = /* INITIAL */
SET &ZALDIR = /* DATASET */
SET &ZALRF = /* PARAMETERS */
SET &ZALLREC = /* */
SET &ZALBLK = /* */
SET &DEVT = /* */
SET &DSORG = /* */
SET &TOTA = /* */
SET &TOTU = /* */
SET &DATE = /* */
SET &DIRU = /* */
SET &MYVAL = /* MYVAL=1 - PRIMARY,MYVAL=2 - SECONDARY */
/* DATASET INFO WILL DISPLAYED */
/* */
/* GLOBAL VARIABLES.FOR USE IN 'ALCe PROCEDURE */
/* */
NGLOBAL &PRIM,&SECON,&MSG,&V,&ZALDIR,&DATE,&ZALVOL,&MYVAL,+ &TOTA,&TOTU,&DIRU,&DEVT,&ZCMD,&ZALBLK,&ISPC
/* */
FREE FILE (SYSIN,SYSPRINT)
/* */
DO WHILE &STATE ¬= 1ØØ /* LOOP 'DIALOG' (UNENDING LOOP) */
DO WHILE &STATE ¬= 2    /* LOOP 'TEST' (TEST THE INPUT FROM A PANEL) */
  ISPEXEC DISPLAY PANEL(REALCL) MSG(&MSG)
  IF &LASTCC = 8 THEN +
    EXIT CODE (Ø) /* EXIT FROM CLIST */
  IF &DSN ¬= THEN +
    IF &SUBSTR(1,&STR(&DSN)) NE ' THEN +
      SET &DSNS = &STR('&DSN')
    ELSE SET &DSNS = &DSN
    ELSE +
      SET &DSNS = &STR('&PROJECT1..&LIBRARY1..&TYPE1')
  IF &SYSDSN(&DSNS) = OK THEN +
    SET &MSG = IGORM1Ø4
  ELSE +
    SET &STATE = 2
END
ELSE +
  SET &STATE = 2
ELSE SET &MSG = IGORM1Ø3 /* PRIMARY NOT CATALOGUED */
END                          /* END OF LOOP 'TEST' */
SET &MSG =
SET &STATE = 1
/* IF YOU PUT A BLANK IN THE 'OPTION' FIELD     OR */
/* YOU FILLED THE PRIMARY DSNAME THE FIRST TIME OR */
/* YOU CHANGED THE PRIMARY DSNAME               OR */
/* YOU SHOW THE SECONDARY DSINFO ON THE LAST SCREEN THEN */
/* YOU MUST TO REFRESH PRIMARY DATASET PARAMETERS */
/* /* ————————————————————————————————————————————————————————————————— */
/* IF &ZCMD = OR &BASE NE &DSNS OR &MYVAL = 2 THEN +
DO
LISTDSI &DSNS DIRECTORY
SET &RC = &LASTCC
IF &RC NE Ø THEN SET &MSG = &SYSMSGLVL1
ELSE +
DO
SET &ZALVOL = &SYSVOLUME
SET &ZALSPC = &SYSUNITS
SET &ISPC = &ZALSPC
IF &ZALSPC = &STR(BLOCK) THEN SET &ISPC = &ISPC&L&SYSBLKSIZE&R
SET &PRIM = &SYSPRIMARY
SET &SECON = &SYSSECONDS
SET &ZALDIR = &SYSADIRBLK
IF &ZALDIR =   THEN SET &ZALDIR = Ø
SET &ZALRF = &SYSCRECFM
SET &ZALLREC = &SYSLRECL
SET &ZALBLK = &SYSBLKSIZE
SET &DEVIT = &SYSUNIT
SET &DSORG = &SYSDSORG
SET &TOTA = &SYSALLOC
SET &TOTU = &SYSUSED
SET &DATE = &STR(&SYSCREATE)
SET &DIRU = &SYSDIRBLK
IF &DIRU =   THEN SET &DIRU = Ø
SET &MYVAL = 1
END
/* ONLY IF YOU PROCESS THE 'A' OR 'R' OPTION AND SHOW THE PROPER */
/* PRIMARY DSINFO ON THE LAST SCREEN, YOU GO TO CALL THE PROCEDURE */
END
ELSE SYSCALL ALC &DSNS &NEWDS
/* /* ————————————————————————————————————————————————————————————————— */
SET &BASE = &DSNS               /* SAVE THE PRIMARY DSNAME */
END                           /* END OF LOOP 'DIALOG' */
/* */
/* 'ALC' PROCEDURE */
/* */
/* ALC: PROC 2 BASE NEW */
IF &ZCMD ¬= R THEN +
DO                          /* PATH 1: PROCESS THE 'A' OPTION */
/* */
IF &ZALDIR = Ø THEN +
DO /* PATH 1A: SEQUENTIAL DATASET */
ALLOCATE F(SYSUT2) DA(&NEW) NEW SPACE(&PRIM,&SECON) &ISPC +
VOLUME(&ZALVOL) UNIT(&DEVT) BLKSIZE(&ZALBLK) LIKE(&BASE) CATALOG
IF &LASTCC = Ø THEN +
DO /* PATH 1AA: */
SET MSG = IGORM100
IF &V = Y THEN +
DO /* WITH COPY */
ALLOC F(SYSUT1) DA(&BASE) OLD
IF &LASTCC = Ø THEN +
DO
ALLOC FILE(SYSIN) DUMMY
ALLOC FILE(SYSPRINT) DUMMY
CALL 'SYS1.LINKLIB(IEBGENER)'
SET RCODE = &LASTCC
FREE DATASET(&NEW)
FREE FILE (SYSIN,SYSPRINT)
FREE DATASET(&BASE)
IF &RCODE NE Ø THEN +
DO
SET MSG = IGORM105 /* IEBGENER FAILED; COPY UNSUCCESSFUL*/
DELETE &NEW
END
ELSE +
DO
SET MSG = IGORM108 /* DATASET IN USE */
FREE DATASET(&NEW)
DELETE &NEW
END
ELSE FREE DATASET(&NEW)
IF &MSG = IGORM100 THEN SET &DATE = &STR(&SYSDATE)
LISTDSI &NEW
SET &RC = &LASTCC
IF &RC = Ø THEN +
DO
SET &TOTA = &SYSALLOC
SET &TOTU = &SYSUSED
SET &DEVT = &SYSUNIT
SET &MYVAL = 2
END /* END OF PATH 1AA */
END
ELSE SET MSG = IGORM106 /* PATH 1AB: ALLOCATION UNSUCCESSFUL*/
/* END OF PATH 1A */
END
ELSE +
DO /* PATH 1B: PARTITION DATASET */
ALLOCATE F(OUTPUT) DA(&NEW) NEW SPACE(&PRIM,&SECON) DIR(&ZALDIR) +
&ISPC +
VOLUME(&ZALVOL) UNIT(&DEVT) BLKSIZE(&ZALBLK) LIKE(&BASE) CATALOG

IF &LASTCC = Ø THEN +
DO                          /* PATH 1BA:                        */
SET MSG = IGORM1ØØ
IF &V = Y THEN +
DO                        /* WITH COPY MEMBERS                */
ALLOC F(INPUT) DA(&BASE) OLD
IF &LASTCC = Ø THEN +
DO
ALLOC FILE (SYSUT3) UNIT(SYSDA) SPACE(2,1) CYLINDERS NEW
ALLOC FILE (SYSUT4) UNIT(SYSDA) SPACE(2,1) CYLINDERS NEW
ALLOC FILE (SYSIN) UNIT(SYSDA) SPACE(1,Ø) TRACKS NEW +
RECFM(F B) LRECL(ØØ) BLKSIZE(ØØØ) DSORG(PS)
OPENFILE SYSIN OUTPUT
SET SYSIN = &STR(' COPY INDD=INPUT,OUTDD=OUTPUT ')
PUTFILE SYSIN
CLOSFILe SYSIN
ALLOC FILE(SYSPRINT) DUMMY
CALL 'SYS1.LINKLIB(IEBCOPY)'
SET RCODE = &LASTCC
FREE DATASET(&NEW)
FREE FILE (SYSUT3,SYSUT4)
FREE FILE (SYSIN,SYSPRINT)
FREE DATASET(&BASE)
IF &RCODE NE Ø THEN +
DO
SET MSG = IGORM1Ø5    /* IEBCOPY FAILED; COPY UNSUCCESSFUL */
DELETE &NEW
END
ELSE +
DO
SET MSG = IGORM1Ø8      /* DATASET IN USE                   */
FREE DATASET(&NEW)
DELETE &NEW
END
ELSE FREE DATASET(&NEW)
IF &MSG = IGORM1ØØ THEN SET &DATE = &STR(&SYSDATE)
LISTDSI &NEW DIRECTORY
SET &RC = &LASTCC
IF &RC = Ø THEN +
DO
SET &TOTA = &SYSALLOC
SET &TOTU = &SYSUSED
SET &DEVT = &SYSUNIT
SET &DIRU = &SYSUDIRBLK
IF &DIRU = THEN SET &DIRU = Ø
SET &MYVAL = 2
END
/* END OF PATH 1BA                        */
END
ELSE  SET MSG = IGORM1Ø6        /* PATH 1BB: ALLOCATION UNSUCCESSFUL*/  
END                      /* END OF PATH 1B */  
                         /* END OF PATH 1 */  
END
ELSE  +
DO                     /* PASH 2: PROCESS THE 'R' OPTION */
   SET &D1 = &SUBSTR(4:5,&STR(&SYSTIME))  
   SET &M1 = &SUBSTR(1:2,&STR(&SYSTIME))  
   SET &Y1 = &SUBSTR(7:8,&STR(&SYSTIME))  
   SET &Z1 = &SUBSTR(1:2,&STR(&SYSUID))  
   SET &DSNAME = &STR(&SYSUID..&Z1.&D1.&M1.&Y1)  
   SET &TWO = 2
   SET &TRANZIT = &SUBSTR(2:(&SYSINDEX(&STR('),&BASE,&TWO) - 1),&BASE)
   SET &TRANZIT = &STR('&TRANZIT..&Z1.&D1.&M1.&Y1')
   IF &ZALDIR = Ø THEN +
      DO                  /* PATH 2A: SEQUENTIAL DATASET */
         ALLOCATE F(SYSUT2) DA(&DSNAME) NEW SPACE(&PRIM,&SECON) + &ISPC +
         VOLUME(&ZALVOL) UNIT(&DEVT) BLKSIZE(&ZALBLK) LIKE(&BASE) CATALOG
         IF &LASTCC = Ø THEN +
            DO             /* PATH 2AA: */
               ALLOC F(SYSUT1) DA(&BASE) OLD
               IF &LASTCC = Ø THEN +
                  DO          /* PATH 2AAA: */
                     ALLOC FILE(SYSIN) DUMMY
                     ALLOC FILE(SYSPRINT) DUMMY
                     CALL 'SYS1.LINKLIB(IEBGENER)'
                     SET RC1 = &LASTCC
                     FREE FILE (SYSIN,SYSPRINT)
                     FREE DATASET(&BASE)
                     FREE DATASET(&DSNAME)
                     IF &RC1 = Ø THEN +
                        DO
                           REN &BASE &TRANZIT
                           REN &DSNAME &BASE
                           SET RC2 = &LASTCC
                           DELETE &TRANZIT
                           SET MSG = IGORM1Ø1
                           LISTDSI &BASE
                           SET &RC = &LASTCC
                           IF &RC = Ø THEN +
                              DO
                                 SET &TOTA = &SYSALLOC
                                 SET &TOTU = &SYSUSED
                                 SET &DEVT = &SYSUNIT
                                 END
                                 IF &RC2 NE Ø THEN +
                                 DO
                                    SET MSG = IGORM1ØB
                                    DELETE &DSNAME
                                 END
                                 ELSE  SET &DATE = &STR(&SYSDATE)
         END
END
ELSE +
   DO                  /* PROC.UNSUCCES.;IEBGENER FAILED */
      SET MSG = IGORM105
      DELETE &DSNAME
   END
/* END OF PATH 2AAA */
ELSE +
   DO                  /* PATH 2AAB:  DATASET IN USE */
      SET MSG = IGORM108
      FREE DATASET(&DSNAME)
      DELETE &DSNAME
   END
/* END OF PATH 2AA */
ELSE SET MSG = IGORM106   /* PATH 2AB:   ALLOC. UNSUCCESSFUL */
   /* END OF PATH 2A */
ELSE +
   DO                           /* PATH 2B:  PARTISHION DATASET */
      ALLOCATE F(OUTPUT) DA(&DSNAME) NEW SPACE(&PRIM,&SECON) DIR(&ZALDIR) +
      &ISPC +
      VOLUME(&ZALVOL) UNIT(&DEVT) BLKSIZE(&ZALBLK) LIKE(&BASE) CATALOG
      IF &LASTCC = Ø THEN +
      DO                         /* PATH 2BA: */
      ALLOC F(INPUT) DA(&BASE) OLD
      IF &LASTCC = Ø THEN +
      DO                          /* PATH 2BAA: */
      ALLOC FILE (SYSUT3) UNIT(SYSDA) SPACE(2,1) CYLINDERS NEW
      ALLOC FILE (SYSUT4) UNIT(SYSDA) SPACE(2,1) CYLINDERS NEW
      ALLOC FILE (SYSIN) UNIT(SYSDA) SPACE(1,Ø) TRACKS NEW +
      RECFM(F B) LRECL(800) BLKSIZE(800) DSORG(PS)
      OPENFILE SYSIN OUTPUT
      SET SYSIN = &STR(' COPY INDD=INPUT,OUTDD=OUTPUT ')
      PUTFILE SYSIN
      CLOSFILE SYSIN
      ALLOC FILE(SYSPRINT) DUMMY
      CALL 'SYS1.LINKLIB(IEBCOPY)'
      SET &RC2 = &LASTCC
      FREE FILE (SYSUT3,SYSUT4)
      FREE FILE (SYSIN,SYSPRINT)
      FREE DATASET(&BASE)
      FREE DATASET(&DSNAME)
      IF &RC2 = Ø THEN +
      DO
      REN &BASE &TRANZIT
      REN &DSNAME &BASE
      SET RC1 = &LASTCC
      DELETE &TRANZIT
      SET MSG = IGORM101
      LISTDSI &BASE DIRECTORY
SET &RC = &LASTCC
IF &RC = Ø THEN +
DO
SET &TOTA = &SYSALLOC
SET &TOTU = &SYSUSED
SET &DEVT = &SYSUNIT
SET &DIRU = &SYSUDIRBLK
  IF &DIRU = THEN SET &DIRU = Ø
END
IF &RC1 NE Ø THEN +
DO
SET MSG = IGORM1Ø8
DELETE &DSNAME
END
ELSE SET &DATE = &STR(&SYSDATE)
END
ELSE +
DO /* IEBCOPY FAILED */
SET MSG = IGORM1Ø5
DELETE &DSNAME
END
END
ELSE +
DO /* PATH 2BAB: */
SET MSG = IGORM1Ø8 /* DATASET IN USE */
FREE DATASET(&DSNAME)
DELETE &DSNAME
END
END
ELSE SET MSG = IGORM1Ø6 /* PATH 2BB: ALLOC. UNSUCCESSFUL */
END /* END OF PATH 2B */
/* ————————————————————————————————————————————————————————————————— */
END /* END OF PATH 2 */
END /* END OF 'ALC' PROCEDURE */

REALCL PANEL MEMBER
)
)ATTR
# TYPE(OUTPUT) INTENS(LOW) JUST(LEFT) CAPS(OFF)
$ TYPE(OUTPUT) INTENS(HIGH)
)
)BODY
%—– DATASET UTILITY PLUS ——–
%OPTION ===> _ZCMD
+
%
% A + Allocate secondary dataset like primary dataset,
+ with copy members?%====>_V+ (Y=YES ,N=NO)
% R + Reallocate primary dataset %blank+ Dataset information
+ PRIMARY DATASET INFO:               SECONDARY DATASET INFO:
+ PROJECT%===>_PRJØ      +          %===>_PROJECT1 +
+ GROUP %===>_LIBØ        +          %===>_LIBRARY1 +
+ TYPE %===>_TYPØ          +          %===>_TYPE1 +
+
+OTHER PRIMARY DATASET: %===>_DSN
+
+OTHER SECONDARY DATASET:%===>_ODSN
+
+$MYVALUE +DATASET INFO:
+
+ Volume serial: _ZALVOL+         Record format: #ZALRF +
+ Device type: _DEVT +            Record length: #ZALLREC +
+ Organization: #DSORG +          Block size: #ZALBLK +
+ Allocated#SPCUCØ #TOTA + Used#SPCUC1 #TOTU +
+ Alloc. dir. blocks: _ZALDIR +   Used dir. blocks: #DIRU +
+ 1st extent#SPCUC2 _prim +
+
+ Secondary#SPCUC3 _secon + Creation date: #DATE +

)INIT
  .HELP = IGRHLP1
  .CURSOR = ZCMD
  IF (&ZCMD ¬= R AND &ZCMD ¬= A)
    &ZCMD = &Z
    &SPCUCØ = TRANS (&ZALSPC CYLINDER,cylinders: TRACK,tracks:
                      BLOCK,blocks:
                      MEGABYTE,megabytes: KILOBYTE,kilobytes:
                      BYTE,bytes:)
    &MYVALUE = TRANS (&MYVAL 1,PRIMARY 2,SECONDARY)
  ks:
    IF (&ZTERM = 3278KN, 3277KN)
      &SPCUCØ = TRANS (&ZALSPC CYLINDER,CYLINDERS: TRACK,TRACKS:
                       BLOCK,BLOCKS:
                       MEGABYTE,MEGABYTES: KILOBYTE,KILOBYTES:
                       BYTE,BYTES:)

    &SPCUC1 = &SPCUCØ
    &SPCUC2 = &SPCUCØ
    &SPCUC3 = &SPCUCØ
    )PROC
    VER(&ZCMD,LIST,' ',A,R,MSG=IGORM1Ø2)
    VER(&V,LIST,N,Y,MSG=IGORM1Ø2)
    VER(&ZALDIR,NUM,MSG=IGORM1Ø7)
    VER(&prim,NUM,MSG=IGORM1Ø7)
    VER(&secon,NUM,MSG=IGORM1Ø7)
    IF ( &DSN = ' ' )
      VER(&PRJØ,NB)
      VER(&LIBØ,NB)
      VER(&TYPØ,NB)
    IF ( &DSN ¬= ' ' ) /* DSN SPECIFIED ?? @M1A*/
      &ZFC = TRUNC(&DSN,1) /* IF FIRST CHARACTER @M1A*/
© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
IF (&ZFC = '''') /* OF DSN IS """" CHECK @M1A*/
&ZREM = .TRAIL /* TO SEE IF LAST """" @M1A*/
&ZREM1 = TRUNC(&ZREM,'''') /* IS MISSING. @M1A*/
IF (&ZREM1 = &ZREM) /* IF LAST """" MISSING @M1A*/
&DSN = '&DSN&ZFC' /* ADD IT TO THE END @M1A*/
VPUT ( PRJØ LIBØ TYPØ ) PROFILE
IF (&ZCMD = A )
IF (&ODSN = ' ')
VER (&PROJECT1,NB)
VER (&LIBRARY1,NB)
VER (&TYPE1,NB)
IF (&ODSN = ' ') /* DSN SPECIFIED ?? @M1A*/
&ZFC = TRUNC(&ODSN,1) /* IF FIRST CHARACTER @M1A*/
IF (&ZFC = '''') /* OF DSN IS """" CHECK @M1A*/
&ZREM = .TRAIL /* TO SEE IF LAST """" @M1A*/
&ZREM1 = TRUNC(&ZREM,'''') /* IS MISSING. @M1A*/
IF (&ZREM1 = &ZREM) /* IF LAST """" MISSING @M1A*/
&ODSN = '&ODSN&ZFC' /* ADD IT TO THE END @M1A*/
)END

IGRHLPI PANEL MEMBER

%TUTORIAL ________ DATASET UTILITY(2) ________ TUTORIAL
%OPTION ===>_ZCMD
+
+
| UTILITIES |
| DATASET UTILITY(2) |

+ You may select the dataset utility (2) by either:
- selecting option%3.22+from the%primary option menu,+or
- selecting option%22+from the%utility selection menu.+  
The following topics are presented in sequence, or may be selected by number:
%1+- Allocating a new partitioned or sequential dataset (secondary dataset) like an existing (primary) dataset
%2+- Reallocating an entire dataset
%3+- Displaying dataset information (such as SIZE, RECFM, BLKSIZE, etc.)
)PROC
&ZSEL = TRANS(&ZCMD
1,IGRHLPI
2,IGRHLIP
3,IGRHLPQ
)
&ZUP = ISR3000
)END
To allocate a new sequential or partitioned dataset (secondary dataset) with the same parameters (DSORG, RECFM, LRECL, BLKSIZE) as you specified in a (primary) dataset, fill in the following fields of the dataset utility(2) panel:

- Enter %A in the option field.
- Enter %Y or leave %N depending on your choice (with copy members or without them)
- Enter the primary and secondary library or dataset name in the appropriate fields.

You will then be shown the same panel, on which all the fields below 'DATASET INFO:' line have been filled in. These values are associated with the primary dataset.

You may leave these values as displayed or modify highlighted fields.

To reallocate an existing sequential or partitioned dataset, fill in the following fields of the dataset utility(2) panel:

- Enter %R in the option field.
- Enter the primary library or dataset name in the appropriate fields.

You will then be shown the same panel, on which all the fields below 'DATASET INFO:' line have been filled in. These values are associated with primary dataset.

You may leave these values as displayed or modify highlighted fields.

To display information about an existing dataset, fill in the following fields of the dataset utility(2) panel:

- Enter %D in the option field.
- Enter the primary library or dataset name in the appropriate fields.

You will then be shown the same panel, on which all the fields below 'DATASET INFO:' line have been filled in. These values are associated with primary dataset.

You may leave these values as displayed or modify highlighted fields.
To display information about an existing dataset, fill in the following fields of the dataset utility(2) panel:
- Leave the option field blank.
- Enter the primary library or dataset name in the appropriate fields.

You will then be shown the same panel, on which all the fields below 'DATASET INFO:' line have been filled in. These values are associated with primary dataset.

```plaintext
)PROC
   &ZUP = IGRHLP1
)END
```

### IGORM10 MESSAGE MEMBER

<table>
<thead>
<tr>
<th>Message Code</th>
<th>Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>IGORM100</td>
<td>'DATASET ALLOCATED'</td>
</tr>
<tr>
<td></td>
<td>'&amp;DSNS ALLOCATED ON VOLUME &amp;ZALVOL'</td>
</tr>
<tr>
<td>IGORM101</td>
<td>'DATASET REALLOCATED'</td>
</tr>
<tr>
<td></td>
<td>'&amp;DSNS REALLOCATED ON VOLUME &amp;ZALVOL'</td>
</tr>
<tr>
<td>IGORM102</td>
<td>'INVALID OPTION' .ALARM = YES</td>
</tr>
<tr>
<td></td>
<td>'THE OPTION YOU ENTERED IS INVALID.'</td>
</tr>
<tr>
<td>IGORM103</td>
<td>''PRIMARY'- NOT CATALOGED' .ALARM = YES</td>
</tr>
<tr>
<td></td>
<td>''&amp;DSNS'' WAS NOT FOUND IN CATALOG.'</td>
</tr>
<tr>
<td>IGORM104</td>
<td>''SEC-Y'ALREADY CATALOGED' .ALARM=YES</td>
</tr>
<tr>
<td></td>
<td>'ENTIRE OR PARTIAL NAME CATALOGED, DATASET ''&amp;NEWDS''.'</td>
</tr>
<tr>
<td>IGORM105</td>
<td>'PROCESSING UNSUCCESSFUL' .ALARM = YES</td>
</tr>
<tr>
<td></td>
<td>'IEBCOPY FAILED '</td>
</tr>
<tr>
<td>IGORM106</td>
<td>'ALLOCATION UNSUCCESSFUL' .ALARM = YES</td>
</tr>
<tr>
<td></td>
<td>'ALLOC FAILED'</td>
</tr>
<tr>
<td>IGORM107</td>
<td>'INVALID VALUE' .ALARM = YES</td>
</tr>
<tr>
<td></td>
<td>'THE VALUE YOU ENTERED IS INVALID.'</td>
</tr>
<tr>
<td>IGORM108</td>
<td>'DATASET IN USE' .ALARM = YES</td>
</tr>
<tr>
<td></td>
<td>'&amp;BASE IN USE '</td>
</tr>
<tr>
<td>IGORM109</td>
<td>'TOO LONG' .ALARM = YES</td>
</tr>
<tr>
<td></td>
<td>'YOU ARE OVERFLOWING TO NEXT LINE'</td>
</tr>
</tbody>
</table>

---

**Igor Kosonovsky**  
**Systems Programmer (Israel)**

© Xephon 1999
RESET command performance group restrictions

INTRODUCTION
This is a follow-up to previous articles in *MVS Update* on user modifications to extend the MVS RESET command to be controlled by the specifications in the IEAICS PARMLIB member. This code is designed to handle modification to OS/390 Version 2 Release 5. The RESETPGN program can be called by the user modification. This handles WLM incompatibility; when we get around to implementing WLM mode some time in the future, I believe there will only be some minor modifications required to make this modification work with the SRVCLASS= keyword.

THE PROBLEM
This modification arose as a result of problems which I have encountered in a few installations. People, be they operators, systems programmers, or scheduling clerks, have a habit of resetting jobs to any performance group they can find. They especially like those performance groups reserved for on-line systems and major operating systems components such as JES2, GRS, ACF2, etc. This has caused problems for me in system measurement since I look at usage by performance group, not to mention the performance problems that it can cause.

A SOLUTION
IBM provides a facility to set jobs into their initial performance group by means of the IEAICS member of SYS1.PARMLIB, be they batch jobs, started tasks, or TSO users. I decided to extend the IEAICS member specifications to the MVS RESET command. Even though IBM provides command security via the OPERCMDS RACF class, that facility does not handle data at the operand level (PERFORM=nn), but rather at the command level.

The process of inserting this modification is as follows:
1 The program RESETPGN is assembled and linked into a LINKLIST library. It must be linked with the linkage editor attribute AMODE(31) since it accesses control blocks above the 16 MB line. Since the program is re-entrant, it may be placed in SYS1.LPALIB or anywhere else in the system LPALIST concatenation – I recommend, however, that it be placed in the LINKLIST concatenation.

2 The RESET command processing module (IEEMB810) is zapped with a patch, which will make it link to program RESETPGN. Note that IEEMB810 is marked as re-entrant and re-usable even though it resides in SYS1.LINKLIB. The zap, which stores in an in-line parameter list, violates this re-entrancy but will work just fine anyway. The alternative is to write RESETPGN as a user SVC, change the zap to replace the SYSEVENT SVC (SVC 95) with the user SVC call, and have RESETPGN issue the SYSEVENT and pass the result back to IEEMB810. This would simplify the zap enormously. I implemented it as I did because I like to be able to pull things in and out on the fly – that is, at any time I can replace RESETPGN with an IEFBR14 program.

3 If you are running with a dynamic BLDL facility, such as PMO or DYNABLDL from the Connecticut Bank tape, or if using the LLA facility of MVS/XA and above, do not forget to do a refresh to pick up the newly zapped version of IEEMB810.

You now have complete control over which performance groups will be used for which jobs and users. The one thing you must do is be specific in the IEAICS member as to what you want. By this I mean that if you have a performance group to swap a job out (ours is 86), it must be specified as an optional performance group on each IEAICS line entry so it will be allowed on a RESET command. If you have multiple low, high, and medium priority batch performance groups, they must all be specified as optional performance groups in addition to the control performance group.

The example code provided below shows that for subsystem JES2, the control performance group is 1, but optionally 3, 4, or 86 may be specified for jobs which do not have specific matches on name or class. Under JES2, jobs beginning with the characters IDMS will have a default performance group of 83 and optionally 86. Note that for all entries, I specify my swapout performance group (86), so any job or
user is able to be swapped out. This should be specified at both the subsystem level and the detail level. This is because the RESETPGN program will not go to check the subsystem level information if a match is found at the detail level. Also note that the order of specification of entries within the IEAICS member is relatively unimportant. Under each subsystem, entries are arranged by transaction name, user-id, class, and, lastly, accounting information. Within each of these, first full non-generic names are shown followed by generic names in descending size order.

SAMPLE IEAICS MEMBER

```
SUBSYS=JES2, PGN=1, OPGN=(3, 4, 86)
TRXNAME=IDMS(1), PGN=83, OPGN=86
TRXNAME=CICP(1), PGN=84, OPGN=86
TRXNAME=CICT(1), PGN=85, OPGN=86
TRXNAME=VIDEO(1), PGN=82, OPGN=86
TRXCLASS=8, PGN=3, OPGN=(4, 86)
SUBSYS=STC, PGN=80, OPGN=(86, 98)
TRXNAME=GRS, PGN=95
TRXNAME=OMEGAMON, PGN=60, OPGN=86
TRXNAME=JES(1), PGN=99, OPGN=86
TRXNAME=MSX(1), PGN=98, OPGN=86
TRXNAME=NET(1), PGN=97, OPGN=86
TRXNAME=VPS, PGN=97, OPGN=86
TRXNAME=TCAM(1), PGN=96, OPGN=86
TRXNAME=RMF(1), PGN=60, OPGN=86
SUBSYS=TSO, PGN=2, OPGN=(7, 11, 86)
USERID=BRUCEB, PGN=2, OPGN=(11, 80, 86)
USERID=APPL(1), PGN=5, OPGN=(80, 86)
```

This modification has been tested on OS/390 Version 2 Release 5 with JES2. It does not, however, support use of account codes as a criterion for resetting the performance group for a job since the accounting information in the MVS JCT/ACT control blocks are kept in the user’s SWA. I did not want to invest the extra coding required to use cross memory services to retrieve this information.

Since some of the SRM-related macros reside in SYS1.PVTMACS, I have hardcoded the required offsets for the necessary fields in the RESTICS program. They are preceded by a comment line containing the macro calls to invoke them. You may wish to use them if you have the macros available on your system. If you do this, be sure to read the comment block preceding label CCT in the RESETPGN program. The equates are used to create otherwise undefined symbols for the
IRARMCT macro to assemble properly. If you use the IRARMCT macro and find that some of the labels used by it are not defined, simply equate each undefined label to zero so that the assembler can find it. In this case it is a rather harmless technique so that the macro can be used to access other defined fields.

There are some other comments in the code dealing with the IBM SRM routine IRARMFPG, which is the find performance group routine. This routine uses stack areas which are similar to standard OS save areas. I have included a few stack areas in the RESETPGNG program for it to use. If the routine changes to use more stack areas the eyecatcher in the program should get overlaid and the program would issue a message detecting the error. The simple solution would be to add a few more stack areas. Additionally, in the parameter list passed to IRARMFPG, the pointer to the RRPA is zero since I do not know how to build an RRPA at this time. In the current IRARMFPG code, however, the RRPA pointer is not used; this may change in the future. The ASCB address, which is documented as being required by IRARMFPG, is also not currently used, but is filled in anyway since it is so easily accessible.

EXAMPLE ZAP

```plaintext
++USERMOD(LMÖØØ34).
++VER(ZØ38) FMID(JBB66Ø4).
++ZAP(IEEMB81Ø) /* THIS ZAP HOOKS IN IEEMB81Ø BEFORE A SYSEVENT
RESETPG IS ISSUED. IT CALLS PROGRAM RESETPGNG TO VALIDATE THE PERFORMANCE GROUP ENTERED ON
THE RESET COMMAND. IEEMB81Ø IS MARKED AS REENTRANT/REUSABLE AND THIS ZAP WILL VIOLATE
THAT REENTRANCY (SEE COMMENTS MARKED WITH AN
ASTERISK). AN ALTERNATIVE METHOD IS
REWRITE RESETICS AS AN SVC TO REPLACE THE
IEEMB81Ø SYSEVENT RESETPG */.

NAME IEEMB81Ø

* START ZAP VERIFICATION
VER ØB5E 58200010 DC X’58200010’ VFY INSTRUCTION
VER ØB50 CB50CB52CB54CB56 DC 19S(*) VFY PATCH AREA (19
HALFW
VER ØB58 CB58CB5ACB5CCB5E
VER ØB60 CB60CB62CB64CB66
VER ØB68 CB68CB6ACB6CCB6E
VER ØB70 CB70CB72CB74

* START ZAP REPLACE
REP ØB5E 47F0CB50 B NEWCODE REPLACE BRANCH
REP ØB50 41F0CB64 NEWCODE LA 15,PGMNAME LOAD R15
```

RESETPGN ASSEMBLER

TITLE 'RESETPGN-ENFORCE IEAICS CONTROL FOR MVS RESET(E) CMD'
RESETPGN AMODE 31
RESETPGN RMODE ANY
RESETPGN CSECT                    ESTABLISH CSECT
SAVE (14,12),,RESETPGN-&SYSDATE   YREGS
LR    R12,R15                   LOAD R12 W/EPA ADDRESS
USING RESETPGN,R12               ESTABLISH ADDRESSABILITY
LR    R7,R1                     SAVE R1 PLIST
USING PL,R7                     ESTABLISH ADDRESSABILITY
GETMAIN RU,LV=WORKLEN            GETMAIN WORKAREA
LR    R2,R1                     LOAD R2 W/A(AREA) FOR MVCL
LA    R3,WORKLEN                LOAD R3 W/WORKAREA LENGTH
SR    R5,R5                     CLEAR R5 FOR MVCL PAD + FROM LEN
MVCL  R2,R4                     CLEAR WORK AREA
ST    R13,(.R1)                ST CALLERS S/A ADDR IN MY S/A
ST    R1,(.R13)                 ST MY S/A ADDR IN CALLERS S/A
LR    R13,R1                    LOAD ADDR OF MY S/A IN R13
USING WORKAREA,R13               ESTABLISH ADDRESSABILITY
EJECT
TM    PL_XOPTIONS,PL_KEYUSED_PERFORM IS REQUEST PERFORM=?
BO    PLISTPGN                   YES, CONTINUE
WTO   'ICS999I PGN= KEYWORD NOT FOUND',DESC=5,ROUTECD=2
B     RETURN                    RETURN TO CALLER
PLISTPGN TM  PL_XOPTIONS,PL_KEYUSED_ASID WAS AN ASID SUPPLIED?
BO    GOTASID                   YES, GO USE IT
L     R15,CVTPTR                ELSE, LOAD A(CVT)
USING CVT,R15                   USING ASVT,R14
L     R14,CVTASVT               LOAD A(ASVT)
USING ASVT,R14
L     R1,ASVTMAXU               LOAD MAX NUM OF ASIDS
LA    R2,ASVTENTY               LOAD A(FIRST ASVT ENTRY)
ASVTLOOP TM  Ø(R2),ASVTAVAL     IS ENTRY AVAILABLE
BO    ASVTVNEXT                 YES, GO CHECK NEXT ENTRY
BCTR  R1,.Ø                    ELSE, DECREMENT ASVT ENTRY COUNT
L     R4,.Ø(R2)                 LOAD A(ASCBD)
USING ASCB,R4                   L     R3,ASCJBNS                 LOAD A(JOBNAME)
CLC  Ø(8,R3),PL_XJOBNAME       JOBNAME COMPARE
BE  FOUNDIT MATCH, GO PROCESS IT
L  R3,ASCBJBNI LOAD A(JOBNAME)
CLC  Ø(R3),PL_XJOBNAME JOBNAME COMPARE
BE  FOUNDIT MATCH, GO PROCESS IT
LA  R2,4(.R2) ELSE, LOAD A(NEXT ASVT ENTRY)
B  ASVTLOOP LOOP BACK

ASVTNEXT LA  R2,4(.R2) LOAD A(NEXT ASVT ENTRY)
BCT  R1,ASVTLOOP LOOP BACK IF MORE ENTRIES
NOTFOUND B  RETURN ELSE, GO RETURN TO CALLER

FOUNDIT MVC  ASID,ASCBASID SAVE ASID NUMBER FROM ASCB
B  SAVEPGN GO SAVE PGN
GOTASID MVC  ASID,PL_XASID SAVE ASID NUMBER FROM WLM PLIST
SAVEPGN MVC  PGNRESET,PL_XPERFORM SAVE PERFORMANCE GROUP

L  R15,CVTPTR LOAD R15 W/A(CVT)
USING CVT,R15 ESTABLISH ADDRESSABILITY
L  R14,CVTOPCTP LOAD R14 W/A(RMCT)
USING RMCT,R14 ESTABLISH ADDRESSABILITY
ST  R11,15,RMCTICST LOAD R11 W/A(ICSC) IF ANY
BZ  RETURN NONE, THEN ALLOW RESET AS IS
ST  R4,ASCBADDR SAVE A(ASCB)
L  R15,ASCBOUCB LOAD R15 W/A(OUCB)
USING OUCB,R15 ESTABLISH ADDRESSABILITY
ST  R15,0UCBADDR SAVE A(0UCB)
LA  R15,FPGOAREA LOAD R15 W/A(PSEUDO FPGO AREA)
ST  R15,FPGOADDR SAVE A(PSEUDO FPGO AREA)

* IF SRVCLASS REQUEST, SET HIGH ORDER BIT OF FPGOPTR TO X'80'???
LA  R15,STACKS LOAD A(STACK AREAS)
LA  R1,312(.R15) LOAD A(NEXT STACK)
ST  R1,4(.R15) CHAIN STACK AREAS
LA  R1,RRPA LOAD R1 W/A(RRPA)
ST  R1,RRPAADDR SAVE A(RRPA)
MVC  RRPANAME,=C'RRPA' PUT EYECATCHER IN RRPA
LA  R15,STACKEND LOAD A(STACK AREA END)
ST  R15,RRPA_STACKEND STORE IN RRPA
LA  R1,ICSPAREA LOAD R1 W/A(PSEUDO ICSP PLIST)
ST  R1,ICSPADDR SAVE A(PSEUDO ICSP PLIST)
OI  ICSPADDR,X'80' TURN ON FULL ICSP INDICATOR
USING ICSP,R1 ESTABLISH ADDRESSABILITY
L  R15,OUCBADDR RELOAD A(0UCB)
MVI  ICSPSUBN,C' ' CLEAR SUBSYSTEM NAME
MVC  ICSPSUBN+1(L'ICSPSUBN-1),ICSPSUBN CLEAR SUBSYSTEM NAME
MVC  ICSPSUB0,0UCBSUBN MOVE SUBSYSTEM NAME
MVC  ICSPTRXN,0UCBTRXN MOVE TRANSACTION NAME
MVC  ICSPUSRD,0UCBUSRD MOVE USERID
MVC  ICSPCLS,0UCBCLS MOVE CLASS
MVI  ICSPACTL,Ø INDICATE NO ACCOUNTING INFO

* THIS CODE DOES NOT SUPPORT ACCOUNTING INFO VALIDITY CHECKING
MVC  ICSPPGN,PGNRESET MOVE REQUESTED PGN TO CHECK
L  R15,RMCTRMSB LOAD A(RMSB)
USING RMSB,R15 ESTABLISH ADDRESSABILITY
L  R15,RMSBFPG LOAD A(FIND PGN ROUTINE)
LM  RØ,R5,FPGOADDR LOAD RØ-R5 WITH PARMS FOR FPG
LR R6,R13           SAVE A(SAVEAREA)
LA R13,STACKS       SKIP SAVEAREA PL/I WORD(FOR FPG)
BALR R14,R15        INVOKE FIND PGN ROUTINE
LR R13,R6           RESTORE A(SAVEAREA)
C R15,FOUR          IS RETURN CODE GOOD
* RC = Ø IS MATCH FOUND/RPGN RETURNED
* 4 IS NO RPGN
* 8 IS NO SUBSYSTEM MATCH
BH SETDEF           NO, GO SET DEFAULT PGN
CLC PGNRESET,FPGONGP WAS REQUESTED PGN RETURNED
BE RETURN           YES, ALLOW IT
EJECT

SETDEF          MVC WTOPGN(MODLWTOL),MODLWT0 MOVE MODEL WTO TO GETMAIN AREA
MVC WTOPGN+MNAME(L'MNAME),PL_XJOBNAME MOVE JOBNAME TO MSG
LH R1,FPGONGP      LOAD R1 W/NEW PGN VALUE RETURNED
CVD R1,CNVTAREA    CONVERT PGN VALUE TO DECIMAL
OI CNVTAREA+7,X'OF' MAKE SIGN PRINTABLE
UNPK WTOPGN+MPGNN(3),CNVTAREA+6(2) MAKE IT PRINTABLE
LH R1,PGNRESET     LOAD R1 W/PGN VALUE REQUESTED
CVD R1,CNVTAREA    CONVERT PGN VALUE TO DECIMAL
OI CNVTAREA+7,X'OF' MAKE SIGN PRINTABLE
UNPK WTOPGN+MPGNO(3),CNVTAREA+6(2) MAKE IT PRINTABLE
WTO MF=(E,WTOPGN)  ISSUE MESSAGE
MVC PL_XPERFORM,FPGONGP PUT NEW PGN BACK INTO PLIST

RETURN          LR R1,R13           LOAD R1 W/A(SAVEAREA)
L R13,4,(R13)      LOAD R13 W/ADDR OF CALLERS S/A
FREEMAIN RU,LV=WORKLEN,A=(1) FREEMAIN WORKAREA
RETURN (14,12),RC=Ø RETURN TO OS WITH RETCODE=Ø
TITLE 'RESETPGN-CONSTANTS AND DATA AREAS'

FOUR DC F'4'        HIGHEST RETURN CODE TO ACCEPT

MODLWT0          WTO 'ICSØØ1I PERFORM=*** CHANGED TO *** FOR ********', X
DESC=5,ROUTCDE=2,MF=L MODEL WTO

MODLWTOL EQU *-MODLWT0 MODEL WTO LENGTH
MPGNO EQU 20,3      OFFSET FOR OLD PERFORMANCE GROUP
MPGNN EQU 35,3      OFFSET FOR NEW PERFORMANCE GROUP
MNAME EQU 43,8      OFFSET FOR TRXNAME/USERID

WORKAREA DSECT
SAVEAREA DS 18F     SHOULD BE FIRST IN WORKAREA
STACKS DC 78F'Ø'    STACK FRAME 1 -----------------|
DC 78F'Ø'           STACK FRAME 2              |
DC 78F'Ø'           STACK FRAME 3              |
STACKEND EQU *      END OF STACK FRAMES -------|

* THE ABOVE BRACKETED AREAS ARE STACK FRAMES USED BY IRARMFPG. THE
* EYECATCHER BELOW IS USED TO DETERMINE IF IRARMFPG HAS CHANGED TO USE
* MORE STACK FRAMES. IF SO, ADD MORE STACK FRAMES ABOVE.
*EYECATCH DC CL8' '  EYECATCHER
FPGOADDR DC A(Ø)    A(FPGO) -------(RØ)---------|
ICSPADDR DC A(Ø)    A(ICSP)        (R1)  |
RMCTADDR DC A(Ø)    A(RMCT)        (R2)  |
RRPAADDR DC A(Ø)    A(RRPA)        (R3)  |
OUCBADDR DC A(Ø) A(OUCB) (R4)  |
ASCBADDR DC A(Ø) A(ASCB) ----- (R5) ----
* THE ABOVE BRACKETED AREA R0 THROUGH R5 VALUES PASSED TO IRARMFPG.
CNVTAREA DS D AREA TO MAKE NEW PGN PRINTABLE
ASID DC H'Ø' ASID OF JOB BEING RESET
PGNRESET DC H'Ø' PGN REQUESTED FROM RESET CMD
RRPA DS CL48 SRM RRPA PLIST
RRPANAME EQU RRPA,4,C'C' RRPA EYECATCHER
RRPA_STACKEND EQU RRPA+4,4,C'A' A(END OF STACK FRAMES)
DS ØF
WTOPGN DS CL(MODLWTOL) AREA FOR MODEL WTO
DS ØF
ICSPAREA DS CL(ICSPLNG) PSEUDO ICSP PLIST
DS ØF ALIGN TO FULLWORD
FPGOAREA DS CL(OUCBDRFP-OUCBFPGO) PSEUDO FPGO AREA
FPGONPG EQU FPGOAREA,4 PSEUDO OUCBNPG FROM IRARMFPG
DS ØF ALIGN TO FULLWORD
WORKLEN EQU *-WORKAREA WORKAREA LENGTH
EJECT
CVT DSECT=YES, LIST=NO CVT
EJECT
IHAASVT ASVT
EJECT
IHAASCB ASCB
EJECT
IRAUCB OUCB
IRARMCT SRM CONTROL TABLE
*RMCT DSECT UNCOMMENT IF MACRO NOT FOUND--|
*RMCTRMSB EQU RMCT+44 POINTER TO RMSB |
*RMCTICST EQU RMCT+22Ø POINTER TO ICSC TABLE-------|
SPACE 2
* IRARMBS SRM VECTOR TABLE
RMSB DSECT UNCOMMENT IF MACRO NOT FOUND--|
RMSBFPG EQU RMSB+88 X'58' POINTER TO FIND PGN ROUTINE---|
SPACE 2
IRAICSP ICS PLIST
*ICSP DSECT UNCOMMENT IF MACRO NOT FOUND--|
* DS CL1Ø SIZE OF ICSP |
*ICSPSUBN EQU ICSP,8 SUBSYSTEM NAME |
*ICSPTRXN EQU ICSP+8,8 TRANSACTION NAME |
*ICSPUSRD EQU ICSP+16,8 USERID |
*ICSPCLS EQU ICSP+24,8 CLASS |
*ICSPPGN EQU ICSP+32,2 PERFORMANCE GROUP IN HEX |
*ICSPFLAG EQU ICSP+34,1 STATUS FLAGS |
*ICSPDP EQU ICSP+35,1 DISPATCHING PRTY IN HEX |
*ICSPACTL EQU ICSP+36,1 LENGTH OF ACCOUNT NUMBER |
*ICSPACTN EQU ICSP+37,143 ACCOUNT NUMBER FROM JCL |
*ICSPCLNG EQU *-ICSP LENGTH OF ICSP---------|
IWMPLIST DSECT
IWMRESET MF=(L, PL) WLM RESET PLIST
END

© Xephon 1999
Assembler instruction trace – part 5

This month we continue our look at the code for the Assembler instruction trace.

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| DC | 2AL2((CCBIT+DBLBIT+FLOABIT)*LEFT+RXBIT)
| *  | .7E-7F (AU,SU)
| OPF_8Ø | EQU (*-OPFLAGS)/2
| DC | 1AL2(FULLBIT*LEFT+RSBIT) .8Ø (SSM)
| DC | 3AL2(ILGLBIT*LEFT+SIBIT) .81-83 (?,LPSW,?)
| DC | 2AL2((BRBIT+FULLBIT)*LEFT+RSBIT) .84-85 (BRXH,BRXLE)
| DC | 2AL2((BRBIT+FULLBIT)*LEFT+RSBIT) .86-87 (BXH,BXLE)
| DC | 2AL2((FULLBIT+SHIFTBIT)*LEFT+RSBIT) .88-89 (SRL,SLL)
| DC | 2AL2((CCBIT+FULLBIT+SHIFTBIT)*LEFT+RSBIT) .8A-8B (SRA,SLA)
| OPF_9Ø | EQU (*-OPFLAGS)/2
| DC | AL2(RSBIT+LMSTMBIT) .9Ø (STM)
| DC | AL2(CCBIT*LEFT+SIBIT) .91 (TM)
| DC | AL2(SIBIT) .92 (MVI)
| DC | AL2(CCBIT*LEFT+SIBIT) .93 (TS)
| DC | 4AL2(CCBIT*LEFT+SIBIT) .94-97 (NI-XI)
| DC | AL2(RSBIT+LMSTMBIT) .98 (LM)
| DC | AL2(ILGLBIT*LEFT+SIBIT) .99 (TRACE)
| DC | 2AL2(RSBIT+ARBIT) .9A-9B (LAM-STAM)
| DC | 4AL2(ILGLBIT*LEFT+SIBIT) .9C-9F (SIO-TCH)NOTESA
| OPF_AØ | EQU (*-OPFLAGS)/2
| DC | 8AL2(ILGLBIT*LEFT+SIBIT) .A0-A7
| DC | 2AL2((CCBIT+DBLBIT)*LEFT+RSBIT) .A8-A9 (MVCLE,CLCLE)
| DC | 2AL2(ILGLBIT*LEFT+SIBIT) .AA-AB
| DC | AL2(SIBIT) .AC (STNSM)
| DC | AL2(SIBIT) .AD (STOSM)
| DC | AL2(CCBIT*LEFT+RSBIT) .AE (SIGP)
| DC | AL2(SIBIT) .AF (MC)
| OPF_BØ | EQU (*-OPFLAGS)/2
| DC | 1AL2(ILGLBIT*LEFT+SIBIT) .BØ
| DC | 1AL2(CCBIT*LEFT+RXBIT) .B1 (LRA)
| DC | 4AL2(ILGLBIT*LEFT+SIBIT) .B2-B5
| DC | 2AL2(RSBIT+LMSTMBIT) .B6-B7 (STCTL,LCTL)
| DC | 2AL2(ILGLBIT*LEFT+SIBIT) .B8-B9
| DC | AL2((CCBIT+FULLBIT)*LEFT+RSBIT) .BA (CS)
| DC | AL2((CCBIT+DBLBIT)*LEFT+RSBIT+LMSTMBIT) .BB (CDS)
| *  | .BB (CDS)
| * ACTUALLY, A JIPPO, SINCE 4 REGS + DBLWORD MUST BE DISPLAYED,
| * SO WE WILL DISPLAY THE DBL WORD STORAGE, AND DUMP ALL REGS
| DC | AL2(ILGLBIT*LEFT+SIBIT) .BC
| DC | 3AL2((CCBIT+FULLBIT)*LEFT+RSBIT) .BD-BF (CLM-ICM)
| OPF_CØ | EQU (*-OPFLAGS)/2
```
DC 17AL2(ILGLBIT*LEFT+SSBIT) .C0-D0

OPF_D1 EQU (*-OPFLAGS)/2
DC 3AL2(SSBIT) .D1-D3 (MVN-MVZ)
DC 4AL2(CCBIT*LEFT+SSBIT) .D4-D7 (NC-XC)
DC AL2(ILGLBIT*LEFT+SSBIT) .DB
DC 3AL2(CCBIT*LEFT+SSBIT+LMSTMBIT) .D9-DB (MVCK-MVCS)
DC AL2(CCBIT*LEFT+SSBIT) .DC (TR)
DC AL2((CCBIT+DBLBIT)*LEFT+SSBIT) .DD (TRT)
DC 2AL2(CCBIT*LEFT+SSBIT) .DE-DF (ED,EDMK)

OPF_EØ EQU (*-OPFLAGS)/2
DC 8AL2(ILGLBIT*LEFT+SSBIT) .E0-E7
DC AL2(SSBIT) .E8 (MVCIN)
DC 5AL2(ILGLBIT*LEFT+SSBIT) .E9-EE
DC 1AL2(CCBIT*LEFT+SSBIT) .EE (PLO)
DC 1AL2(ILGLBIT*LEFT+SSBIT) .EF

OPF_FØ EQU (*-OPFLAGS)/2
DC AL2(CCBIT*LEFT+SSBIT) .FØ (SRP)
DC 3AL2(SSBIT) .F1-F3 (MVO-UNPK)
DC 4AL2(ILGLBIT*LEFT+SSBIT) .F4-F7
DC 4AL2(CCBIT*LEFT+SSBIT) .F8-FB (ZAP-AP)
DC 2AL2(SSBIT) .FC-FD (MP,DP)
DC 2AL2(ILGLBIT*LEFT+SSBIT) .FE-FF

OPF_1ØØ EQU (*-OPFLAGS)/2
TITLE '****************** OP-CODE NAMES ******************'

BCDOP DS ØF
DC 'C' SPM BALR BCTR BCR ' Ø0-Ø7 ØØ
DC 'C'SSK ISK SVC BSM BASSMBASR MVCL CLCL ' Ø8-ØF Ø2Ø
DC 'C'LP R LNR LTR LCR NR CLR OR XR ' 10-16 Ø4Ø
DC 'C'LR CR AR SR MR DR ALR SLR ' 18-1F 06Ø
DC 'C'LPDR LNDR LDR LCDR HDR LSALR XMR ' 20-27 08Ø
DC 'C'LDR CDR ADR SDR MDR DDR AWR SWR ' 28-2F 0AØ
DC 'C'LPER LNER LTER LCER HER LRER AXR SXR ' 30-37 0CØ
DC 'C'LER CER AER SER MER DER AUR SUR ' 38-3F 0EØ
DC 'C'STH LA STC IC EX BAL BCT BC ' 40-47 10Ø
DC 'C'LL CH AH SH MH BAS CVD CVB ' 48-4F 12Ø
DC 'C'STH LAE N CL O X ' 50-57 14Ø
DC 'C'L C A S M D AL SL ' 58-5F 16Ø
DC 'C'STD MXD ' 60-67 18Ø
DC 'C'LD CD AD SD MD DD AW SW ' 68-6F 1AØ
DC 'C'STE MS ' 70-77 1CØ
DC 'C'LE CE AE SE ME DE AU SU ' 78-7F 1EØ
DC 'C'LSM LPSW BRXH BRXLEBXH BXL ' 80-87 20Ø
DC 'C'SRL SLL SRA SLA SRDL SLDL SRDA SDL ' 88-8F 22Ø
DC 'C'STM TM MVI TS NI CLI OI XI ' 90-97 24Ø
DC 'C'LM RCLEAM STAM SIO TIO HIO TCH ' 98-9F 26Ø
DC 'C' 'AØ-A7 28Ø
DC 'C'MVCLECLCLE STNSMSTOSMSIGP MC ' A8-AF 2AØ
DC 'C' LRA S** STCTLLCTL ' B0-B7 2CØ
DC 'C' CS CDS CLM STCM ICM ' BB-BF
DC 'C' 'CØ-C7 30Ø
DC 'C' 'CB-CF 32Ø
DC 'C' MVN MVC MVZ NC CLC OC XC ' Ø0-Ø7 34Ø
DC 'C' MVCK MVCP MVCS TR TRT ED EDMK ' DB-DF 36Ø

B2FLAGS
DS 0H

DC 2AL2(ILGLBIT*LEFT+RSBIT) .B200-B201
DC AL2(DBLBIT*LEFT+B2STGBIT) .B202 (STIDP)
DC AL2(ILGLBIT*LEFT+RSBIT) .B203
DC 2AL2((CCBIT+DBLBIT)*LEFT+B2STGBIT)  B204-05 (SCK-STCK)
DC AL2(FULLBIT*LEFT+B2ADRBIT) .B20A (SPKA)
DC AL2(ILGLBIT*LEFT+RSBIT) .B20B (IPK)
DC AL2(ILGLBIT*LEFT+RSBIT) .B20C
DC AL2(ILGLBIT*LEFT+RSBIT) .B20D (PTLB)
DC 2AL2(ILGLBIT*LEFT+RSBIT) .B20E-B20F
DC 4AL2(DBLBIT*LEFT+B2STGBIT) .B210-11 (SPX,STPX)
DC AL2(HALFBIT*LEFT+B2STGBIT) .B212 (STAP)
DC 5AL2(ILGLBIT*LEFT+RSBIT) .B213-17
DC AL2(FULLBIT*LEFT+B2STGBIT+LMSTMBIT+ARBIT) .B218 (PC)
DC AL2(FULLBIT*LEFT+B2ADRBIT) .B219 (SAC)
DC AL2(((CCBIT+FULLBIT)*LEFT+B2ADRBIT+LMSTMBIT)
DC 6AL2(ILGLBIT*LEFT+RSBIT) .B21B-1F
DC AL2(FULLBIT*LEFT+B2RBIT+B2R2BIT) .B221 (IPTE)
DC AL2(FULLBIT*LEFT+B2RBIT) .B222 (IPM)
DC AL2(FULLBIT*LEFT+B2RBIT+B2R2BIT) .B223 (IVSK)
DC AL2((CCBIT+FULLBIT)*LEFT+B2RBIT) .B224 (IAC)
DC 2AL2((CCBIT+FULLBIT)*LEFT+B2RBIT+B2R2BIT)
DC 3AL2(FULLBIT*LEFT+B2RBIT) .B225-27 (SSAR-ESAR)
DC 4AL2((CCBIT+FULLBIT)*LEFT+B2RBIT+B2R2BIT) .B228-2B (PT-SSKE)
DC 5AL2((CCBIT+FULLBIT)*LEFT+B2RBIT+B2R2BIT)
DC AL2(((CCBIT+DBLBIT+FLOATBIT)*LEFT+B2RBIT+B2R2BIT)  B22C (TB)
DC 2AL2(ILGLBIT*LEFT+RSBIT) .B22D (DXR)
DC 2AL2((CCBIT+FULLBIT)*LEFT+B2R1BIT)  B230-31(CSCH,HSCH)
DC 4AL2(((CCBIT+FULLBIT)*LEFT+B2R1BIT+B2STGBIT)
DC AL2((CCBIT+FULLBIT)*LEFT+B2STGBIT) .B236 (TPI)
DC AL2(FULLBIT*LEFT+B2R1BIT) .B237 (SAL)
DC AL2((CCBIT+FULLBIT)*LEFT+B2R1BIT) .B238 (RSCH)
DC 2AL2(((CCBIT+FULLBIT)*LEFT+B2R1BIT+B2STGBIT)
DC 3AL2((ILGLBIT+FULLBIT)*LEFT+B2R1BIT+B2STGBIT)
DC AL2((CCBIT+FULLBIT)*LEFT+B2R1BIT) .B239-3A (STRCW,STCPS)
DC 2AL2((CCBIT+FULLBIT)*LEFT+B2R1BIT) .B23B (RCHP)
DC AL2(DBLBIT*LEFT+B2R1BIT) .B23C (SCM)
DC 3AL2((ILGLBIT+FULLBIT)*LEFT)  B23D-B23F
DC AL2((FULLBIT+RBRBIT)*LEFT+B2RBIT+B2R2BIT+LMSTMBIT+ARBIT)
DC 5AL2((ILGLBIT+FULLBIT)*LEFT)  B240 (BAKR)
DC 2AL2((CCBIT+FULLBIT)*LEFT)  B241-245
DC 2AL2((ILGLBIT+FULLBIT)*LEFT)  B246 (STURA)
DC 2AL2(DBLBIT*LEFT+B2RBIT)  B247 (MSTA)
DC AL2((ILGLBIT+FULLBIT)*LEFT)  B248 (PALB)
```
* DC    AL2(FULLBIT*LEFT+B2RBIT+B2R2BIT+LMSTMBIT+ARBIT) .B249 (EREG)
* DC    AL2(DBLBIT*LEFT+B2RBIT+B2R2BIT)   .B24A (ESTA)
* DC    AL2((ILGLBIT+FULLBIT)*LEFT)       .B24B (LURA)
* DC    AL2((ILGLBIT+FULLBIT)*LEFT)       .B24C (TAR)
* DC    AL2(FULLBIT*LEFT+B2RBIT+B2R2BIT)  .B24D (CPYA)
* DC    AL2(FULLBIT*LEFT+B2RBIT+B2R2BIT)  .B24E (SAR)
* DC    AL2(FULLBIT*LEFT+B2RBIT+B2R2BIT)  .B24F (EAR)
* DC    AL2((CCBIT+FULLBIT)*LEFT+B2RBIT+B2R2BIT) .B250-B251
* DC    AL2((ILGLBIT+FULLBIT)*LEFT)       .B252 (MSR)
* DC    AL2(FULLBIT*LEFT+B2RBIT+B2R2BIT)  .B253
* DC    AL2(((CCBIT+DBLBIT)*LEFT)+B2RBIT+B2R2BIT+B2RØBIT) .B254 (MVPG)
* DC    AL2(((CCBIT+DBLBIT)*LEFT)+B2RBIT+B2R2BIT+B2RØBIT) .B255 (MVST)
* DC    AL2((ILGLBIT+FULLBIT)*LEFT)       .B256
* DC    AL2(2AL2((ILGLBIT+FULLBIT)*LEFT)) .B25F-B278
* DC    AL2(FULLBIT*LEFT+B2ADRBIT)        .B279 (SACF)

SPACE 3

B2NAMES DS ØH
*     0  1  2  3  4  5  6  7
*    8  9  A  B  C  D  E  F
DC C' STIDP SCK STCK SCKC STCKC'   B200
DC C'SPT STPT SPKA IPK PTLB   '   B208
DC C'SPX STPX STAP SIE   '   B210
DC C'PC SAC CFC   '   B218
DC C' IPTE IPM IVSK IAC SSAR EPAR ESAR   '   B220
DC C'PT ISKE RBRE SSKE TB DXR   '   B228
DC C'CSCH HSCH MSCH SSCH STSCHTCH TPI SAL   '   B230
DC C'RSCH STCWSCTCSRCHP SCHM   '   B238
DC C'BAKR CKSM SQDR SQER STURAMSTA   '   B240
DC C'PALB EREG ESTA LURA TAR CPYA SAR EAR   '   B248
DC C' MSR MVPG MVST CUSE   '   B250
DC C' CLST SRST   '   B258
DC C'   '   B260
DC C'   '   B268
DC C'   '   B270
DC C' SACF   '   B278

A7NAMES DS ØH
*     0  1  2  3  4  5  6  7
*    8  9  A  B  C  D  E  F
DC C'TMH TML BRM BRNS BRCT   '   TMH,TML
DC C'LI AHI MHI CHI   '   TMH,TML

A7FLAGS DS ØH
DC 2AL2((CCBIT+FULLBIT)*LEFT+SIBIT) .TMH,TML
DC 2AL2((ILGLBIT+FULLBIT)*LEFT+SIBIT)
```

DC 3AL2((FULLBIT+BRBIT*LEFT)+SIBIT)       .BRC-BRCT
DC AL2((ILGLBIT*LEFT)+SIBIT)
DC AL2((FULLBIT*LEFT)+SIBIT)             .LHI
DC AL2((ILGLBIT*LEFT)+SIBIT)
DC AL2((FULLBIT+CCBIT)*LEFT+SIBIT)       .AHI
DC AL2((ILGLBIT*LEFT)+SIBIT)
DC AL2((FULLBIT*LEFT)+SIBIT)             .MHI
DC AL2((ILGLBIT*LEFT)+SIBIT)
DC AL2((FULLBIT+CCBIT)*LEFT+SIBIT)       .CHI
E5FLAGS  DS ØH
DC 2AL2((CCBIT+DBLBIT)*LEFT+SIBIT)       .LASP, TPROT
DC 12AL2(ILGLBIT*LEFT+SSBIT)             .E5Ø1-E5ØD
DC 2AL2(SSBIT)                           .MVCSK, MVCDK
E5NAMES  DS ØH
*                Ø    1    2    3    4    5    6    7
*                8    9    A    B    C    D    E    F
DC C'LASP TPROT 'E5ØØ
DC C' MVCSKMVCDK' E5Ø8
DATAEND  DS ØH     END OF TRACE ROUTINE DATA AREA
TITLE '*** CONSTANTS TO BE COPIED TO RELOCATABLE STORAGE ***'
MODELS  DS ØD
OPEN Ø,MF=L,MODE=31
CLOSE Ø,MF=L,MODE=31
BLDVRP MF=L,MODE=31,BUFFERS=(4Ø96(3))
WTO 'R7 = ........ ',MF=L
MODELSZ  EQU *-MODELS
TITLE '**************** PC DESCRIPTIONS ******************'
LXLIST  DS ØF
DC A((LXØ0,LXØ1,LXØ2,LXØ3,LXØ4,LXØ5,LXØ6,LXØ7,LXØ8,LXØ9)
DC A((LXØA,LXØB,LXØC,LXØD,LXØE,LXØF)
DC A((LX1Ø,LX11,LX12,LX13,LX14)
HI_LX    EQU (*)-LXLIST)/4
LXØ0    DC A((LX01-*)/30)
DC CL3Ø'LXRES'         .Ø0
DC CL3Ø'LXFRE'         .Ø1
DC CL3Ø'ETCRE'         .Ø2
DC CL3Ø'ETDES'         .Ø3
DC CL3Ø'ETCON'         .Ø4
DC CL3Ø'ETDIS'         .Ø5
DC CL3Ø'AXRES'         .Ø6
DC CL3Ø'AXFRE'         .Ø7
DC CL3Ø'AXEXT'         .Ø8
DC CL3Ø'AXSET'         .Ø9
DC CL3Ø'ATSET'         .ØA
DC CL3Ø'PC/AUTH RES. MAN.' .ØB
DC CL3Ø'** RESERVED **'  .ØC
DC CL3Ø'ALESERV ADD/ADDPASN' .ØD
DC CL3Ø'ALESERV DELETE'  .ØE
DC CL3Ø'ALESERV EXTRACT(H)' .ØF
LXØ1    DC A((LX02-*)/30)
DC CL3Ø'ENQ/DEQ/RESERVE' .Ø0
DC CL3Ø'ENQ/DEQ/RESERVE REDRIVE' .Ø1
DC CL3Ø'ENQ/DEQ/RESERVE RTM' .Ø2

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
DC CL3Ø'GRS DUMP SERVICES' .03
DC CL3Ø'QSCAN SCOPE=STEP/SYSTEM(S)' .04
DC CL3Ø'GRS STG. MGMT. SERVICE' .05
DC CL3Ø'QSCAN SCOPE=LOCAL/GLOBAL' .06
DC CL3Ø'DEQUEUE FAST PATH' .07
DC CL3Ø'ENQUEUE FAST PATH' .08
DC CL3Ø'GRS MAINLINE ESTAE' .09
DC CL3Ø'FRR FOR ENQ/DEQ/RESERVE' .0A
DC CL3Ø'XMS ENQ SERVICE' .0B
DC CL3Ø'XMS ENQ SERVICE' .0C
DC CL3Ø'GRS LATCH CREATE' .0D
DC CL3Ø'XMS ENQ SERVICE' .0E
DC CL3Ø'GRS LATCH PURGE' .0F

LXØ2 DC A((LXØ3-*)/3Ø)

DCL LXØ3 DC A((LXØ4-*)/3Ø)

DC CL3Ø'VSM CPOOL BUILD' .00
DC CL3Ø'VSM CPOOL EXPAND' .01
DC CL3Ø'VSM CPOOL DELETE' .02
DC CL3Ø'VSMLIST' .03
DC CL3Ø'VSMLOC' .04
DC CL3Ø'CPUTIMER' .05
DC CL3Ø'VIRTUAL FETCH CSVVFORK' .06
DC CL3Ø'DATA-IN-VIRTUAL' .07
DC CL3Ø'SYMPOTM RECORDS' .08
DC CL3Ø'LSEXPPAND' .09
DC CL3Ø'LOCASCB STOKEN=' .0A
DC CL3Ø'STOREAGE OBTAIN' .0B
DC CL3Ø'RTM DYN. RESOURCE MGR' .0C
DC CL3Ø'WAIT LINKAGE=SYSTEM' .0D
DC CL3Ø'POST LINKAGE=SYSTEM' .0E
DC CL3Ø'PC-ESTAE' .0F
DC CL3Ø'ASCRE/ASDES/ASEX'T .10
DC CL3Ø'STOREAGE RELEASE' .11
DC CL3Ø'TCBTOKEN SERVICE' .12
DC CL3Ø'TESTART SERVICE' .13
DC CL3Ø'CSVQUERY' .14
DC CL3Ø'** RESERVED **' .15
DC CL3Ø'TIMEUSED' .16
DC CL3Ø'SRB SUSPEND WITH TOKEN' .17
DC CL3Ø'SRB RESUME WITH TOKEN' .18
DC CL3Ø'SRB PURGE WITH TOKEN' .19
DC CL3Ø'LLACOPY' .1A
DC CL3Ø'RCFSTAT' .1B
DC CL3Ø'RCFCONF' .1C
DC CL3Ø'AFFINITY SERVICE' .1D
DC CL3Ø'SDOM CONNECT' .1E
DC CL3Ø'SDOM DISCONNECT' .1F
DC CL3Ø'CTRACEWR - WRITE' .20
DC CL3Ø'PC TIME' .21
DC CL3Ø'UCB SERVICE - AUTH' .22
DC CL3Ø'UCB SERVICE - UNAUTH' .23
DC CL3Ø'CONFIG. CHANGE MGR' .24

DC   CL3Ø'UNIT VERIFICATION SRV.'       .25
DC   CL3Ø'NAME/TOKEN SRV'               .26
DC   CL3Ø'NAME/TOKEN SRV'               .27
DC   CL3Ø'CONVTOD'                      .28
DC   CL3Ø'DYNAMIC APF'                  .29
DC   CL3Ø'APPC'                         .2A
DC   CL3Ø'** UNDEFINED **'             .2B
DC   CL3Ø'CSRL16J'                     .2C
DC   CL3Ø'SCHEDIRB'                    .2D
DC   CL3Ø'IOS SUPPORT'                 .2E
DC   CL3Ø'HCD S/39Ø MICRO.PROC. SUPP'  .2F
DC   CL3Ø'** UNDEFINED **'             .30
DC   CL3Ø'** UNDEFINED **'             .31
DC   CL3Ø'** UNDEFINED **'             .32
DC   CL3Ø'** UNDEFINED **'             .33
DC   CL3Ø'HCD SYSPLEX SRV'             .34
LX04  DC   A((LX05-*)/3Ø)                .00
DC   CL3Ø'WTO SERVICE'                 .00
LX05  DC   A((LX06-*)/3Ø)                .00
DC   CL3Ø'SYSTEM TRACE SERVICES'       .00
LX06  DC   A((LX07-*)/3Ø)                .00
DC   CL3Ø'VIRTUAL FETCH CSVVFSCH'       .00
LX07  DC   A((LX08-*)/3Ø)                .00
DC   CL3Ø'SMF BUFFERING'                .00
LX08  DC   A((LX09-*)/3Ø)                .00
DC   CL3Ø'LIBRARY LOOKASIDE (LLA)'     .00
LX09  DC   A((LX0A-*)/3Ø)                .00
DC   CL3Ø'VLF'                         .00
LX0A  DC   A((LX0B-*)/3Ø)                .00
DC   CL3Ø'YLF'                         .00
LX0B  DC   A((LX0C-*)/3Ø)                .00
DC   CL3Ø'** RESERVED **'              .00
LX0C  DC   A((LX0D-*)/3Ø)                .00
DC   CL3Ø'** RESERVED FOR DFP **'      .00
LX0D  DC   A((LX0E-*)/3Ø)                .00
DC   CL3Ø'** RESERVED FOR DFP **'      .00
LX0E  DC   A((LX0F-*)/3Ø)                .00
DC   CL3Ø'LLACOPY'                     .00
LX0F  DC   A((LX10-*)/3Ø)                .00
DC   CL3Ø'SDOM'                        .00
LX10  DC   A((LX11-*)/3Ø)                .00
DC   CL3Ø'MVS MESSAGE SERVICE'         .00
LX11  DC   A((LX12-*)/3Ø)                .00
DC   CL3Ø'** RESERVED **'              .00
LX12  DC   A((LX13-*)/3Ø)                .00
DC   CL3Ø'OE SPACE SWITCH'             .00
LX13  DC   A((LX14-*)/3Ø)                .00
DC   CL3Ø'OE NONSPACE SWITCH'          .00
DC   CL3Ø'OE AUTH SPACE SWITCH'        .00
DC   CL3Ø'OE SP. SW. 4 SPEC. CALLBLE SRV'. .03
LX14  DC   A((LAST_LX-*)/3Ø)             .00
DC   CL3Ø'** RESERVED **'              .00
DC   CL3Ø'PERF. BLK. CREATE'           .01
DC   CL3Ø'PERF. BLK. DELETE'           .02
DC  CL3Ø'PERF. BLK. RELATE'           .Ø3
DC  CL3Ø'WORKLOAD RPT. ICS/IPS CHANGE' .Ø4
DC  CL3Ø'PERF. BLK. SWITCH'           .Ø5
DC  CL3Ø'PERF. BLK. DISCONNECT'       .Ø6
DC  CL3Ø'PERF. BLK. CONNECT'          .Ø7
DC  CL3Ø'WLM QUERY'                   .Ø8
DC  CL3Ø'POLICY MGMT. READ'           .Ø9
DC  CL3Ø'POLICY MGMT. VARY'           .ØA
DC  CL3Ø'POLICY MGMT. INSTALL SVDEF'  .ØB
DC  CL3Ø'POLICY MGMT. READ SVDEF'     .ØC
DC  CL3Ø'ADMIN. APPL. AUTH'           .ØD
DC  CL3Ø'WLM REPORTING COLLECT'       .ØE
DC  CL3Ø'WLM REPORTING QUERY'         .ØF
DC  CL3Ø'POLICY MGMT. CDS CHANGE'     .10
DC  CL3Ø'WLM LOCK'                    .11
DC  CL3Ø'OPS. DISPLAY WLM'            .12
DC  CL3Ø'WLM QUERY'                   .13
DC  CL3Ø'GENERIC RESOURCE REGISTRATION' .14
DC  CL3Ø'GENERIC RESOURCE SELECTION'  .15
DC  CL3Ø'RECOV&DUMP SDATA(WLM)'       .16
DC  CL3Ø'WLM RPT. RESMGR'             .17
DC  CL3Ø'ENCLAVE CREATE'              .18
DC  CL3Ø'ENCLAVE DELETE'              .19
DC  CL3Ø'ENCLAVE CLASSIFICATION QUERY' .1A
DC  CL3Ø'SYSTEM CAPACITY QUERY'       .1B
DC  CL3Ø'SYSPLEX ROUTING REGISTRATION' .1C
DC  CL3Ø'SYSPLEX ROUTING DEREG.'      .1D
DC  CL3Ø'SYSPLEX ROUTING SELECTION'   .1E
DC  CL3Ø'SERVICE DEFINITION INSTALL'  .1F
DC  CL3Ø'SERVICE DEFINITION EXTRACT'  .20
DC  CL3Ø'RETURN ACT. CLASSIFIC. RULES' .21
DC  CL3Ø'POLICY ACTIVATION EXTERNAL'  .22
DC  CL3Ø'WLM MODIFY CONNECT'          .23
LAST_LX  EQU   *
TITLE '**************** SVC DESCRIPTIONS ******************'
SVCNAMES DS    ØH
SVCØØØ   DC    CL4Ø'EXCP/XDAP'
SVCØØ1   DC    CL4Ø'WAIT/WAITR/PRTOV'
SVCØØ2   DC    CL4Ø'POST'
SVCØØ3   DC    CL4Ø'EXIT'
SVCØØ4   DC    CL4Ø'GETMAIN'
SVCØØ5   DC    CL4Ø'FREEMAIN'
SVCØØ6   DC    CL4Ø'LINK'
SVCØØ7   DC    CL4Ø'XCTLL'
SVCØØ8   DC    CL4Ø'LOAD'
SVCØØ9   DC    CL4Ø'DELETE'
SVCØØ10  DC    CL4Ø'GETMAIN/FREEMAIN (R-FORM)'
SVCØØ11  DC    CL4Ø'TIME'
SVCØØ12  DC    CL4Ø'SYNCH'
SVCØØ13  DC    CL4Ø'ABEND'
SVCØØ14  DC    CL4Ø'SPIE'
SVCØØ15  DC    CL4Ø'ERREXCP'

SVC016  DC  CL40 'PURGE'
SVC017  DC  CL40 'RESTORE'
SVC018  DC  CL40 'BLDL/FIND (D-TYPE)'
SVC019  DC  CL40 'OPEN'
SVC020  DC  CL40 'CLOSE'
SVC021  DC  CL40 'STOW'
SVC022  DC  CL40 'OPEN TYPE=J'
SVC023  DC  CL40 'CLOSE TYPE=T'
SVC024  DC  CL40 'DEVTYPE'
SVC025  DC  CL40 'TRKBAL'
SVC026  DC  CL40 'CATALOG/INDEX/LOCATE'
SVC027  DC  CL40 'OBTAIN'
SVC028  DC  CL40 '?????????????'
SVC029  DC  CL40 'SCRATCH'
SVC030  DC  CL40 'RENAME'
SVC031  DC  CL40 'FEOV'
SVC032  DC  CL40 'ALLOC'
SVC033  DC  CL40 'IOHALT'
SVC034  DC  CL40 'MCGR/QEDIT'
SVC035  DC  CL40 'WTO/WTOR'
SVC036  DC  CL40 'WTL'
SVC037  DC  CL40 'SEGLD/SEGWT'
SVC038  DC  CL40 '?????????????'
SVC039  DC  CL40 'LABEL'
SVC040  DC  CL40 'EXTRACT'
SVC041  DC  CL40 'IDENTIFY'
SVC042  DC  CL40 'ATTACH(X)'
SVC043  DC  CL40 'CIRB'
SVC044  DC  CL40 'CHAP'
SVC045  DC  CL40 'OVLYBRCH'
SVC046  DC  CL40 'TTIMER'
SVC047  DC  CL40 'STIMER'
SVC048  DC  CL40 'DEQ'
SVC049  DC  CL40 '?????????????'
SVC050  DC  CL40 '?????????????'
SVC051  DC  CL40 'SNAP/DUMP'
SVC052  DC  CL40 'RESTART'
SVC053  DC  CL40 'RELEX'
SVC054  DC  CL40 'DISABLE'
SVC055  DC  CL40 'EOV'
SVC056  DC  CL40 'ENO/RESERVE'
SVC057  DC  CL40 'FREEDBUF'
SVC058  DC  CL40 'RELBUF/REQBUF'
SVC059  DC  CL40 'OLTEP'
SVC060  DC  CL40 'STAE/STAI-ESTAE/ESTAI'
SVC061  DC  CL40 'IKJEG56A'
SVC062  DC  CL40 'DETACH'
SVC063  DC  CL40 'CHKPT'
SVC064  DC  CL40 'RDJFCB'
SVC065  DC  CL40 '?????????????'
SVC066  DC  CL40 'BTAMTEST'
SVC067  DC  CL40 '?????????????'
| SVC068 | DC | CL40'SYNADAF/SYNADRLS' |
| SVC069 | DC | CL40'BSP' |
| SVC070 | DC | CL40'GSERV' |
| SVC071 | DC | CL40'ASGNBFR/BUFINFO/RLSEBFR' |
| SVC072 | DC | CL40'CALL TO IEAVVCTR' |
| SVC073 | DC | CL40'SPAR' |
| SVC074 | DC | CL40'DAR' |
| SVC075 | DC | CL40'DQUEUE' |
| SVC076 | DC | CL40'IFBSTAT' |
| SVC077 | DC | CL40'????????????' |
| SVC078 | DC | CL40'LSPACE' |
| SVC079 | DC | CL40'STATUS' |
| SVC080 | DC | CL40'????????????' |
| SVC081 | DC | CL40'SETPRT/SETDEV' |
| SVC082 | DC | CL40'????????????' |
| SVC083 | DC | CL40'SMFWTM/SMFEMT,BRANCH=NO' |
| SVC084 | DC | CL40'GRAPHICS' |
| SVC085 | DC | CL40'DDRSWAP' |
| SVC086 | DC | CL40'ATLAS' |
| SVC087 | DC | CL40'DOM' |
| SVC088 | DC | CL40'????????????' |
| SVC089 | DC | CL40'????????????' |
| SVC090 | DC | CL40'????????????' |
| SVC091 | DC | CL40'V0LSTAT' |
| SVC092 | DC | CL40'TCBEXCP' |
| SVC093 | DC | CL40'TGET/TPUT/TPG' |
| SVC094 | DC | CL40'STCC' |
| SVC095 | DC | CL40'SYSEVENT' |
| SVC096 | DC | CL40'STAX' |
| SVC097 | DC | CL40'IKJEGS9G' |
| SVC098 | DC | CL40'PROTECT' |
| SVC099 | DC | CL40'DYNALLOC' |
| SVC100 | DC | CL40'IKJEFFIB' |
| SVC101 | DC | CL40'QTIP' |
| SVC102 | DC | CL40'AQCTL' |
| SVC103 | DC | CL40'XLATE' |
| SVC104 | DC | CL40'TOPCTL' |
| SVC105 | DC | CL40'IMGLIB' |
| SVC106 | DC | CL40'????????????' |
| SVC107 | DC | CL40'MODESET' |
| SVC108 | DC | CL40'????????????' |
| SVC109 | DC | CL40'ESR TYPE 4' |
| SVC110 | DC | CL40'????????????' |
| SVC111 | DC | CL40'CALL TO 1GC111' |
| SVC112 | DC | CL40'PGRLSE' |
| SVC113 | DC | CL40'PGFIX/PGFREE/PGLOAD/PGOUT' |
| SVC114 | DC | CL40'EXCPVR' |
| SVC115 | DC | CL40'????????????' |
| SVC116 | DC | CL40'ESR TYPE 1' |
| SVC117 | DC | CL40'DEBCHK' |
| SVC118 | DC | CL40'????????????' |
| SVC119 | DC | CL40'TESTAUTH' |
| SVC120 | DC | CL40'GETMAIN/FREEMAIN' |
SVC121 DC CL40'VSAM'
SVC122 DC CL40'ESR TYPE 2'
SVC123 DC CL40'PURGEDQ'
SVC124 DC CL40'TPIO'
SVC125 DC CL40'EVENTS'
SVC126 DC CL40'????????????'
SVC127 DC CL40'????????????'
SVC128 DC CL40'????????????'
SVC129 DC CL40'????????????'
SVC130 DC CL40'RACHECk'
SVC131 DC CL40'RACINIT'
SVC132 DC CL40'RACLST'
SVC133 DC CL40'RACDEF'
SVC134 DC CL40'????????????'
SVC135 DC CL40'????????????'
SVC136 DC CL40'????????????'
SVC137 DC CL40'ESR TYPE 6'
SVC138 DC CL40'PGSER'
SVC139 DC CL40'CVAFDIR/CVAFDSM/CVAFSEQ/CVAFVOL/CVAFVRF'
SVC140 DC CL40'????????????'
SVC141 DC CL40'????????????'
SVC142 DC CL40'????????????'
SVC143 DC CL40'GENKEY/RETKEY/CIPHER/EMK'
SVC144 DC CL40'OPENMVS PTRACE'

TITLE '************ RELOCATABLE WORKING STORAGE ************'

TEMPWK DSECT
TEMP_R0 DS F
TEMP_R1 DS F
TEMP_R2 DS F
TEMP_R3 DS F
TEMP_R4 DS F
TEMP_R5 DS F
TEMP_R6 DS F
TEMP_R7 DS F
TEMP_R8 DS F
TEMP_R9 DS F
TEMP_R10 DS F
TEMP_R11 DS F
TEMP_R12 DS F
TEMP_R13 DS F
TEMP_R14 DS F
TEMP_R15 DS F
PR_SAVE DS 16F

WRKSTOR DSECT
MYSAVE DS 9D
DUB DS D
REGTBL DS 16F
OLDREGS DS 16F
AR_SAVE DS 16F
AR_OLD DS 16F
AR_WORK DS 16F
TEMPREGS DS 16F
EREGSAVE DS 16F
FLTREGS DS 4D
FLTRØ EQU FLTREGS,8
FLTR2 EQU FLTREGS+8,8
FLTR4 EQU FLTREGS+16,8
FLTR6 EQU FLTREGS+24,8
NEW_IPTR DS F
AR_FLAG DS A
XMSSTAT DS 3F
@ACB DS A
@RPL DS A
FLAGS DS H
XCELL DS CL6
PSFLAGS DS H
PSXCELL DS CL6
CODEFLD DS CL6
REALCC DS X
EXD_LINE DS CL133
AR_LINE DS CL133
PRTLINE DS CL133
OFFSET EQU PRTLINE,4
I_PTR EQU PRTLINE+5,8
HEXOP EQU PRTLINE+15,12
CC EQU PRTLINE+29
OPCODE EQU PRTLINE+32,5
FIELDS EQU PRTLINE+38
GR_1 EQU PRTLINE+67              FIRST OPERAND REGISTER
DR1 EQU GR_1+9                  2ND HALF OF 1ST DOUBLE REGISTER
SS_EFA1 EQU GR_1,3Ø              .1ST OPERAND FOR SS-INSTRUCTIONS
SS_EFA2 EQU SS_EFA1+34,3Ø       .2ND SS-OPERAND
SS_EFA3 EQU PRTLINE+122,8       .A(2ND OP) FOR MVC, TR, TRT
DR2A EQU DR1+12                 2ND DOUBLE REGISTER
DR2B EQU DR2A+9                 2ND HALF OF 2ND DOUBLE REGISTER
GR_2 EQU DR2A                    SECOND SINGLE REGISTER
EFA1 EQU GR_2                    1ST EFFECTIVE ADDRESS (RX INST)
EFA2 EQU PRTLINE+111             2ND EFFECTIVE ADDRESS (RX INST)
FR2 EQU GR_1+33                 .2ND FLTPT REG (1ST = R1)
CALLPARM DS 20F
OPENLST OPEN 0,MF=L,MODE=31
CLOSELST CLOSE 0,MF=L,MODE=31
DLVRP BLDVRP MF=L,MODE=31,BUFFERS=(4096(3))
WTO1 WTO 'ASMTRACE - FLAGS=XXXX ',MF=L
*                      .-....1
ACB ACB DDNAME=SYSTRACE,MACRF=(ADR,OUT),RMODE31=ALL
ACB_SIZE EQU -*ACB
RPL RPL ACB=0,AREA=0,AAREALEN=133,RECLEN=133,OPTCD=(ADR,MVE)
RPL_SIZE EQU -*RPL
RETSTACK
XMS_WRK DS CL256
PR_STACK DS 40CL20
CUR_PR DS A
EPSTACK@ DS A

CUREP DS F
DS 35CL(L’EPSTACK)
EPSTACK DS 40CL110
WSLEN EQU -*-WRKSTOR
REGEOU
TITLE '****************************************************************************** D S E C T S **********************************************************************'
IHAPSA
IKJTCB
DCBD DSORG=PS,DEVD=DA
IHAASCB
TIOT DSECT
IEFTIOT1
END

MACRO
MACRO
GETCC &COND
GBLA &PF_CCVAL
LCLC &LWK1
AIF ('&COND'(1,1) LT 'Ø' OR '&COND'(1,1) GT '9').NOTNUM
&PF_CCVAL =ETA &COND
\MEXIT
.NOTNUM AIF (K’&COND NE 1).TWOCHAR
&LWK1 =ETC 'COND'
AGO .CALCC
.TWOCHAR AIF (K’&COND NE 2).INVCOND
AIF ('&COND'(1,1) NE 'N').OTHERMN
&LWK1 =ETC '&COND'(2,1)
AGO .CALCC
.OTHERMN AIF ('&COND' EQ 'EQ').BC8
AIF ('&COND' EQ 'LT').BC4
AIF ('&COND' NE 'LE').TRYGT
&PF_CCVAL =ETA 13
\MEXIT
.TRYGT AIF ('&COND' EQ 'GT').BC2
AIF ('&COND' NE 'GE').INVCOND
&PF_CCVAL =ETA 11
\MEXIT
.CALCC AIF ('&LWK1' NE 'O').TRYH
&PF_CCVAL =ETA 1
AGO .TSTN
.TRYH AIF ('&LWK1' EQ 'P' OR '&LWK1' EQ 'H').BC2
AIF ('&LWK1' EQ 'L' OR '&LWK1' EQ 'M').BC4
AIF ('&LWK1' EQ 'E' OR '&LWK1' EQ 'Z').BC8
AGO .INVCOND
.BC8 ANOP
&PF_CCVAL =ETA 8
AGO .TSTN
.BC4 ANOP
&PF_CCVAL =ETA 4
MACRO
  POPINS &P
  COPY PPFGBLCØ
  LCLA &W
&W SETA &P
AGO .TEST
.UNSTACK ANOP
  AIF ('&PF_IIND3(&W)' EQ '').ONEOP
  AIF ('&PF_IIND4(&W)' NE '').THREEOP
&PF_IIND5(&W) &PF_IIND1(&W) &PF_IIND2(&W),&PF_IIND3(&W)
AGO .INCTR
.THREEOP ANOP
&PF_IIND5(&W) &PF_IIND1(&W) &PF_IIND2(&W),&PF_IIND3(&W),&PF_IIND4(&W)
AGO .INCTR
.ONEOP ANOP
&PF_IIND5(&W) &PF_IIND1(&W) &PF_IIND2(&W)
.INCTR ANOP
&W SETA &W+1
.TEST AIF (&W LE &PF_II).UNSTACK
&PF_II SETA &P-1
  AIF ('&PF_NEST(&PF_NI)'(3,1) NE ' ' OR
    '&PF_NEST(&PF_NI)'(4,0071007
  1) EQ ').NEQ
&PF_IIND5(&PF_II) &PF_IIND1(&PF_II) &PF_IIND2(&PF_II)
.NEQ AIF (&PF_II GT Ø OR (&PF_II EQ Ø AND
    '&PF_NEST(&PF_NI)'(5,4)+00740007
EQ 'IF')).END
MNOTE 8,'NEGATIVE INSTRUCTION STACK PTR. EXPANSION INVALID.'
.END MEND
**********************************************************************

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

Pieter Wiid
Advisory Systems Engineer
Persetel (South Africa) © Xephon 1999
Sterling Software has announced Version 3.0 of its VM:Webgateway Web-to-host software for using legacy applications from a Web browser while maintaining end-to-end security.

Users can Web-enable and Web-enhance all existing mainframe applications on OS/390, MVS, VM, and VSE and include full-screen applications. It uses Secure Sockets Layer technology to encrypt data transmitted between Web browsers and the mainframe and it uses client and server certificates that authenticate Web browser users.

There is new support for multi-tier security standards, and trusted third-party Certificate Authorities, such as VeriSign, will soon offer standard, digital certificates that use multi-tier certificate chaining for additional security. This will let VM:Webgateway users implement the new multi-tier encryption technology. Version 3.0 also apparently uses 20% less CPU resources, and it now supports HTTP 1.1, the latest version, for persistent connections.

For further information contact:
Sterling Software, 1800 Alexander Bell Drive, Reston, VA 22091, USA.
Tel: (703) 264 8000.
Sterling Software, Sterling Court, Eastworth Road, Chertsey, Surrey, KT16 8DF, UK.
Tel: (01932) 587000.

Xephon will be holding its annual MVS '99 conference at the Chelsea Hotel in London, 19-20 May 1999. MVS '99 is designed specifically for technical managers, systems programmers, strategic planners, and other system specialists at MVS/ESA and OS/390 installations, and provides a thorough analysis of new facilities and products in the MVS world, and a full update on the latest technical hints and tips for MVS administrators.

With e-commerce growing, unpredictable future capacity needs, and Year 2000 and euro compliance issues looming, users must exploit OS/390 functionality to the full. Furthermore, for MVS technical staff the required skill-set is gradually changing, with application integration and Web skills gaining prominence. Sites now need to plan ahead, as never before, to align business and IT strategy.


The attendance fee for MVS Update subscribers is £555.00 plus £64.75 VAT. For further information, please telephone the registrar, Angela Scott, on (01635) 33823.

***