140 MVS update

May 1998

In this issue

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

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

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

Editor
Jaime Kaminski

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

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

Subscriptions and back-issues
A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £325.00 in the UK; $485.00 in the USA and Canada; £331.00 in Europe; £337.00 in Australasia and Japan; and £335.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Printed in England.

OS/390 Version 2 Releases 5 and 6

INTRODUCTION

March 27 1998 saw the commercial availability of OS/390 Version 2 Release 5. This latest incarnation of MVS is dominated by features designed to enhance hardware and software security, e-commerce enablement tools, and functionality designed to facilitate server consolidation, systems management, and application development.

VERSION 2 RELEASE 5

The new features and functionality available in OS/390 Version 2 Release 5 include:

- Firewall technologies – previously available separately as a kit with Release 4, now integrated in Release 5.
- eNetwork Communications Server – has new functionality with improved performance.
- LDAP – Lightweight Directory Access Protocol Server function in the OS/390 Security Server.
- Lotus Domino Go Webserver Version 4.6.1.
- Print Server – is a new optional and separately priced feature.
- Component Broker (in beta) is available to selected customers.

VERSION 2 RELEASE 6

Preliminary information regarding OS/390 Version 2 Release 6, with proposed availability in September 1998, suggests additional support for Java, and plans for:

- Enhancements to Parallel Sysplex.
- New functionality for Unix System Services.
- Enhancements to the eNetwork Communications Server.
- Lotus Domino Go Webserver Version 5.0 enhancements.

- Modifications to the Network File System and the Distributed File Service.
- The release of Component Broker after beta completion.

ANALYSIS

There are several themes that emerge from these announcements:

- IBM is reducing the complexity of OS/390. This is clearly seen in the improved installation times with ServerPac, and the subsystem and integration testing. The beta release of Component Broker for OS/390 also provides another pointer. By tying together disparate enterprise systems via an object framework Component Broker will mask the complexity of enterprise-class computing. While this is beneficial to the traditional large enterprise base, the underlying implication is that these elements are part of a concerted attempt to drive OS/390 penetration down into the Small to Medium Enterprise (SME) market.
- The considerable emphasis placed on electronic commerce and security shows that this is where IBM really sees the best prospects for OS/390 growth. The sheer scale of electronic commerce growth is likely to drive users towards the highly scalable and secure top-end platforms as users outstrip the capacity of the middleground systems. This is why we see the RACF security environment being complemented by Triple DES encryption, LDAP on the OS/390 Security Server, digital certificates, hardware cryptography for SET, and integrated firewall technologies. Electronic commerce enablement tools are enhanced with Net.Commerce Version 3 and eNetwork Host On-Demand Version 2.
- Most importantly for MVS users, OS/390 now has extensive facilities for server consolidation, systems management, and application development. Consolidation and integration are big issues in all sizes of organization at present, with large installations drawing LAN servers and departmental systems back into the data centres, and smaller organizations looking to reduce costs by consolidating a smaller number of departmental machines.

GTF SVC trace formatter

INTRODUCTION

GTF traces are amongst the most useful tools provided for the systems programmer to really understand the internal logic flow and functioning of an MVS system. This is especially true during the diagnosis of a problem which may not conveniently provide a dump for analysis.

IPCS enables one to process GTF traces and to extract a huge quantity of information, but there are certain things which IPCS doesn't tell you in a useful shorthand way, so I have developed a REXX EXEC which I have found to be beneficial in getting a quick handle on a problem.

The EXEC SVCFRMT reads a GTF SVC trace in standard SYS1.TRACE format created by a run of GTF specifying TRACE=SVC in the parmlib member. The JCL required to run the program is as follows:

```
//REXXJCL  EXEC PGM=IRXJCL,
//  PARM='SVCFRMT xxxxxxxx nnn'
//* PARM='SVCFRMT xxxxxxxx'
//* PARM='SVCFRMT'
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*
//GTFDAT   DD DSN=SYS1.TRACE,DISP=SHR
//SYSEXEC  DD DSN=SYS1.REXX,DISP=SHR
```

SVCFRMT accepts two optional parameters, a job name of up to 8 characters (xxxxxxxx), and an SVC number (nnn). These parameters allow one to narrow the output of the program as required. The output consists of a single line per matched SVC invocation in the trace, presenting the following fields:

- Jobname
- SVC number in decimal and hex
- Macro name as documented by IBM or third-party software vendors
- TOD Clock in format hh:mm:ss.microsecs

- CDE program calling the SVC
- R0 at call
- R1 at call
- RF at call
- Old PSW before call
- CPU executing the call
- TCB in control at call

Below is an example of the output of SVCFRMT up to the CDE field. Although the full output line fits in 133 characters, space considerations do not permit the entire detail line to be shown here:

```
No jobname specified, processing all jobs
No SVC specified, processing all SVCs

Jobname    SVC  Macro       TOD Clock      CDE

IMSREG1  124 7C TPIO     13:11:44.161702 DFSXDSP0
NET       60 3C STAE     13:11:44.162002 **IRB***
NET        3 03 EXIT     13:11:44.162240 **IRB***
IMSREG1  124 7C TPIO     13:11:44.162441 DFSXDSP0
NET       60 3C STAE     13:11:44.162630 **IRB***
NET        3 03 EXIT     13:11:44.163161 **IRB***
NET       60 3C STAE     13:11:44.163613 **IRB***
NET        3 03 EXIT     13:11:44.164069 **IRB***
TSO1       1 01 WAIT     13:11:44.164092 SVC-T2
JES2       0 00 EXCP     13:11:44.164276 HASJES20
CICSREG1   2 02 POST     13:11:44.164493 DSN2EXT3
JES2       2 02 POST     13:11:44.164517 HASJES20
CICSREG1   1 01 WAIT     13:11:44.164547 DSN2EXT3
CICSREG2   1 01 WAIT     13:11:44.165403 DFHKETCB
CICSREG2   1 01 WAIT     13:11:44.165501 DSN2EXT3
JES2     119 77 TESTAUTH 13:11:44.165526 HA$PSUBS
JES2      60 3C STAE     13:11:44.165615 HA$PSUBS
JES2      60 3C STAE     13:11:44.165915 HA$PSUBS
JES2       2 02 POST     13:11:44.166096 HA$PSUBS
CICSREG2   2 02 POST     13:11:44.166161 DSN2EXT3
JES2       1 01 WAIT     13:11:44.166166 HA$PSUBS
JES2       2 02 POST     13:11:44.166243 HASJES20
CICSREG1   2 02 POST     13:11:44.166520 DFHKETCB
CICSREG1   1 01 WAIT     13:11:44.166627 DFHKETCB
CICSREG2   1 01 WAIT     13:11:44.167315 DFHKETCB
JES2     119 77 TESTAUTH 13:11:44.167318 HA$PSUBS
CICSREG2   1 01 WAIT     13:11:44.167408 DSN2EXT3
```

```
CICSREG2   2 Ø2 POST     13:11:44.167607 DSN2EXT3
JES2     119 77 TESTAUTH 13:11:44.168100 HA$PSUBS
CICSREG1   2 Ø2 POST     13:11:44.168171 DSN2EXT3
CICSREG1   1 Ø1 WAIT     13:11:44.168220 DSN2EXT3
CICSREG2   1 Ø1 WAIT     13:11:44.168521 DFHKETCB
CICSREG2   1 Ø1 WAIT     13:11:44.168736 DSN2EXT3
JES2     119 77 TESTAUTH 13:11:44.168889 HA$PSUBS
CICSREG2   2 Ø2 POST     13:11:44.168907 DSN2EXT3
CICSREG1   2 Ø2 POST     13:11:44.169635 DFHKETCB
CICSREG1   1 Ø1 WAIT     13:11:44.169739 DFHKETCB
JES2     119 77 TESTAUTH 13:11:44.169752 HA$PSUBS
CICSREG2   1 Ø1 WAIT     13:11:44.170083 DFHKETCB
CICSREG2   1 Ø1 WAIT     13:11:44.170168 DSN2EXT3
JES2     119 77 TESTAUTH 13:11:44.170570 HA$PSUBS
CICSREG2   2 Ø2 POST     13:11:44.170658 DSN2EXT3
CICSREG1   2 Ø2 POST     13:11:44.171243 DSN2EXT3
```

A brief analysis of the above sample reveals that it is taken from a system where multiple CICS regions spend much of their time waiting on DB2 threads.

A second example follows. This focuses on a single job, an IMS region, which is processing VSAM I/O requests:

```
Processing Job IMSREG1
No SVC specified, processing all SVCs

Jobname    SVC  Macro       TOD Clock      CDE

IMSREG1  124 7C TPIO     13:11:44.161702 DFSXDSPØ
IMSREG1  124 7C TPIO     13:11:44.162441 DFSXDSPØ
IMSREG1  121 79 VSAM     13:11:44.190147 DFSXDSPØ
IMSREG1    2 Ø2 POST     13:11:44.201478 DFSXDSPØ
IMSREG1  121 79 VSAM     13:11:44.201584 DFSXDSPØ
IMSREG1  121 79 VSAM     13:11:44.294395 DFSXDSPØ
IMSREG1  114 72 EXCPVR   13:11:44.305195 DFSXDSPØ
IMSREG1  121 79 VSAM     13:11:44.306320 DFSXDSPØ
IMSREG1  121 79 VSAM     13:11:44.313855 DFSXDSPØ
IMSREG1  121 79 VSAM     13:11:44.324012 DFSXDSPØ
IMSREG1  121 79 VSAM     13:11:44.329455 DFSXDSPØ
IMSREG1  121 79 VSAM     13:11:44.332908 DFSXDSPØ
IMSREG1   47 2F STIMER   13:11:44.339176 DFSFDLDØ
IMSREG1  114 72 EXCPVR   13:11:44.347938 DFSXDSPØ
IMSREG1  121 79 VSAM     13:11:44.371560 DFSXDSPØ
IMSREG1  121 79 VSAM     13:11:44.391863 DFSXDSPØ
IMSREG1  121 79 VSAM     13:11:44.398721 DFSXDSPØ
IMSREG1  121 79 VSAM     13:11:44.403291 DFSXDSPØ
IMSREG1  121 79 VSAM     13:11:44.422693 DFSXDSPØ
IMSREG1  121 79 VSAM     13:11:44.458434 DFSXDSPØ
IMSREG1  121 79 VSAM     13:11:44.479744 DFSXDSPØ
```

```
IMSREG1  121 79 VSAM     13:11:44.486778 DFSXDSPØ
IMSREG1  121 79 VSAM     13:11:44.489556 DFSXDSPØ
```

Note that the SVC table in the program is not 100% complete, in keeping with IBM documentation. If you trace an SVC that is not in the table, the program displays '** Undef **' in the macro field, and further investigation will be required to discover the function and owner of such an SVC.

SVCFRMT REXX

```
/*———————————————————————— REXX ————————————————————————————*/
/* Function   : GTF trace analysis                              */
/*              Process a GTF trace with TRACE=SVC              */
/*——————————————————————————————

———————————————————————————————————————————*/
numeric digits 21
arg job svc
call init_svc
trecs = Ø; vrecs = Ø
say ' '
if job = '' then
  say 'No jobname specified, processing all jobs'
  else
  say 'Processing Job' job
if svc = '' then
  say 'No SVC specified, processing all SVCs'
  else
  do
  i = c2d(x2c(svc))
  say 'Processing SVC' i svc svcexp.i
  end
say ' '
say 'Jobname    SVC  Macro       TOD Clock      CDE       RØ',
    '      R1       RF         Old PSW       CPU   TCB'
say ' '
done = 'n'
do while done = 'n'
  "execio 1 diskr gtfdat"
  if rc = Ø then
    do
    parse pull gtfrec
    trecs = trecs + 1
    call proc_rec
    end
  else
```

```
    done = 'y'
end
say ' '
say 'Valid records processed =' format(vrecs,9,Ø)
say 'Total records processed =' format(trecs,9,Ø)
say ' '
exit Ø
/*———————————————————————————*/
/* Process a record                                                  */
/*———————————————————————————*/
proc_rec:
eid = substr(gtfrec,11,2)
if eid ¬= '1ØØØ'x then
  return
jn = substr(gtfrec,19,8)
if job = '' then
  nop
  else
  if jn ¬= job then
    return
svctyp = c2x(substr(gtfrec,3Ø,1))
if svc = '' then
  nop
  else
  if svctyp ¬= svc then
    return
vrecs = vrecs + 1
i = c2d(x2c(svctyp))
tod = substr(gtfrec,3,8)
call proc_tod
cpu   = c2x(substr(gtfrec,17,2))
opsw1 = c2x(substr(gtfrec,27,4))
opsw2 = c2x(substr(gtfrec,31,4))
tcbad = c2x(substr(gtfrec,35,4))
cdenm = substr(gtfrec,39,8)
svcrf = c2x(substr(gtfrec,47,4))
svcrØ = c2x(substr(gtfrec,51,4))
svcr1 = c2x(substr(gtfrec,55,4))
say jn format(i,3,Ø) svctyp justify(svcexp.i,8,' ') ttod,
    justify(cdenm,8,' ') svcrØ svcr1 svcrf opsw1 opsw2 cpu tcbad
return
/*———————————————————————————————————————————————————————————————*/
/* Process TOD                                                       */
/*———————————————————————————————————————————————————————————————*/
proc_tod:
sec = c2d(tod) / (4Ø96 * 1ØØØ * 1ØØØ)
sec = sec - 3Ø29443200
day = sec % (24 * 6Ø * 6Ø)
sec = sec - (24 * 6Ø * 6Ø * day)
day = day + 1
hr  = sec % (6Ø * 6Ø)
```

```
sec = sec - (60 * 60 * hr)
min = sec % 60
sec = sec - (60 * min)
hr  = format(hr,2,0)
min = format(min,2,0)
sec = format(sec,2,6)
ttod = hr ]] ':' ]] min ]] ':' ]] sec
ttod = translate(ttod,'0',' ')
return
/*-------------------------------------------------------------------------*/
/* Initialise SVC macro expansions                                         */
/*-------------------------------------------------------------------------*/
init_svc:
svcexp. = '** Undef **'
svcexp.0 = 'EXCP'
svcexp.1 = 'WAIT'
svcexp.2 = 'POST'
svcexp.3 = 'EXIT'
svcexp.4 = 'GETMAIN'
svcexp.5 = 'FREEMAIN'
svcexp.6 = 'LINK'
svcexp.7 = 'XCTL'
svcexp.8 = 'LOAD'
svcexp.9 = 'DELETE'
svcexp.10 = 'GETMAIN/FREEMAIN'
svcexp.11 = 'TIME'
svcexp.12 = 'SYNCH'
svcexp.13 = 'ABEND'
svcexp.14 = 'SPIE'
svcexp.15 = 'ERREXCP'
svcexp.16 = 'PURGE'
svcexp.17 = 'RESTORE'
svcexp.18 = 'BLDL'
svcexp.19 = 'OPEN'
svcexp.20 = 'CLOSE'
svcexp.21 = 'STOW'
svcexp.22 = 'OPEN'
svcexp.23 = 'CLOSE'
svcexp.24 = 'DEVTYPE'
svcexp.25 = 'TRKBAL'
svcexp.26 = 'CATALOG/INDEX/LOCATE'
svcexp.27 = 'OBTAIN'
svcexp.29 = 'SCRATCH'
svcexp.30 = 'RENAME'
svcexp.31 = 'FEOV'
svcexp.32 = 'No macro'
svcexp.33 = 'IOHALT'
svcexp.34 = 'MGCR/QEDIT'
svcexp.35 = 'WTO'
svcexp.36 = 'WTL'
```

```
svcexp.37 = 'SEGLD'
svcexp.39 = 'LABEL'
svcexp.40 = 'EXTRACT'
svcexp.41 = 'IDENTIFY'
svcexp.42 = 'ATTACH'
svcexp.43 = 'CIRB'
svcexp.44 = 'CHAP'
svcexp.45 = 'OVLYBRCH'
svcexp.46 = 'TTIMER'
svcexp.47 = 'STIMER'
svcexp.48 = 'DEQ'
svcexp.51 = 'SNAP/SDUMP'
svcexp.52 = 'RESTART'
svcexp.53 = 'RELEX'
svcexp.54 = 'DISABLE'
svcexp.55 = 'EOV'
svcexp.56 = 'ENQ/RESERVE'
svcexp.57 = 'FREEDBUF'
svcexp.58 = 'RELBUF/REQBUF'
svcexp.59 = 'OLTEP'
svcexp.60 = 'STAE'
svcexp.61 = 'IKJEGS6A'
svcexp.62 = 'DETACH'
svcexp.63 = 'CHKPT'
svcexp.64 = 'RDJFCB'
svcexp.66 = 'BTAMTEST'
svcexp.68 = 'SYNADAF'
svcexp.69 = 'BSP'
svcexp.70 = 'GSERV'
svcexp.71 = 'ASGNBFR/BUFINQ/RLSEBFR'
svcexp.72 = 'RDJFCB'
svcexp.73 = 'SPAR'
svcexp.74 = 'DAR'
svcexp.75 = 'DQUEUE'
svcexp.76 = 'No macro'
svcexp.78 = 'LSPACE'
svcexp.79 = 'STATUS'
svcexp.81 = 'SETPRT'
svcexp.83 = 'SMFWTM'
svcexp.84 = 'GRAPHICS'
svcexp.85 = 'DDRSWAP'
svcexp.86 = 'ATLAS'
svcexp.87 = 'DOM'
svcexp.91 = 'VOLSTAT'
svcexp.92 = 'TCBEXCP'
svcexp.93 = 'TGET/TPG/TPUT'
svcexp.94 = 'STCC'
svcexp.95 = 'SYSEVENT'
svcexp.96 = 'STAX'
svcexp.97 = 'IKJEGS9G'
```

```
svcexp.98 = 'PROTECT'
svcexp.99 = 'DYNALLOC'
svcexp.100 = 'IKJEFFIB'
svcexp.101 = 'QTIP'
svcexp.102 = 'AQCTL'
svcexp.103 = 'XLATE'
svcexp.104 = 'TOPCTL'
svcexp.105 = 'IMGLIB'
svcexp.107 = 'MODESET'
svcexp.109 = 'ESR type 4'
svcexp.111 = 'No macro'
svcexp.112 = 'PGRLSE'
svcexp.113 = 'PGFIX/PGFREE/PGLOAD/PGOUT/PGANY'
svcexp.114 = 'EXCPVR'
svcexp.116 = 'ESR type 1'
svcexp.117 = 'DEBCHK'
svcexp.119 = 'TESTAUTH'
svcexp.120 = 'GETMAIN/FREEMAIN'
svcexp.121 = 'VSAM'
svcexp.122 = 'ESR type 2'
svcexp.123 = 'PURGEDQ'
svcexp.124 = 'TPIO'
svcexp.125 = 'EVENST'
svcexp.130 = 'RACHECK'
svcexp.131 = 'RACINIT'
svcexp.132 = 'RACLIST'
svcexp.133 = 'RACDEF'
svcexp.137 = 'ESR type 6'
svcexp.138 = 'PGSER'
svcexp.139 = 'CVAF'
svcexp.143 = 'GENKEY/RETKEY/CIPHER/EMK'
/*-------------------------------------------------------------------*/
/* From here on the SVCs are site specific                           */
/*-------------------------------------------------------------------*/
svcexp.225 = 'CICS HP'
svcexp.226 = 'CICS'
svcexp.230 = 'User SVC 1'
svcexp.231 = 'User SVC 2'
svcexp.240 = '3rd Party SVC 1'
svcexp.241 = '3rd Party SVC 2'
return
```

Patrick Mullen
MVS Systems Consultant (Canada)

Using REFDD and LIKE JCL statements

INTRODUCTION

The REFDD and LIKE JCL statements are relatively new additions to the JCL language, and while I was preparing a JCL course recently I thought I would double-check my understanding of these two functions. As it turned out, the results of that check were more interesting than I had expected. Initially, and according to the manuals, I had gained the impression that these new statements would effectively do the same thing (ie provide a means for easily creating SMS-controlled datasets using another dataset as a reference point). However, through a very simple test, it became clear that REFDD and LIKE do not work in the same way. Hence I have put together this short article to try to clear up how these statements work for anyone else who might have the same expectations as I did.

A COMPARISON OF REFDD AND LIKE

To begin with here is a small sample of JCL containing both statements to illustrate their use:

```
//jobname JOB CLASS=S,MSGLEVEL=(1,1)
//A EXEC PGM=IEFBR14
//DD1 DD DSN=userid.testdata,DISP=SHR
//B DD DSN=userid.temp1,DISP=(,CATLG),REFDD=*.DD1
//C DD DSN=userid.temp2,DISP=(,CATLG),LIKE=userid.testdata
```

Both LIKE and REFDD are supposed to use the dataset to which they refer as a means of obtaining allocation information. However, they do not necessarily give the same results! In the example above userid.temp2 will be created similarly to userid.testdata, but userid.temp1 might not. The reason for the difference is as follows:

- REFDD looks at the SMS allocation information for the dataset and allocates the new dataset according to this information along with any additionally supplied dataset defining JCL.
- LIKE meanwhile uses the physical attributes of the dataset for the allocation (along with any dataset defining JCL).

As a result, the above use of REFDD and LIKE will only result in the creation of identical datasets if the allocation of userid.testdata was done only on SMS defaults. It is therefore important to be careful how you exploit these new statements in your JCL.

One further point regarding these statements, which can easily be missed (especially regarding LIKE), is that both statements can refer to a dataset being created in the JCL. In this manner you can code all the JCL necessary for creating one particular dataset, and then use REFDD or LIKE to incorporate that JCL for another dataset. From a personal viewpoint, this is about the only time I would risk using REFDD, because I know the allocations will be consistent with the original. The rest of the time, LIKE seems a more reliable proposition.

C A Jacques
Systems Programmer (UK)

Understanding GRS

INTRODUCTION

GRS as an address space has been an integral part of MVS for many years, yet there is still a fair bit of misunderstanding on how it works and what role it performs in the management of resources. This article hopes to give you a better understanding of how and where it fits in and how to interpret the information made available by MVS commands and the GQSCAN macro. We will also briefly look at what it does in a sysplex, a few pitfalls, and how it has evolved to be used in a parallel sysplex in an OS/390 environment.

It is essential that you should understand that no resource (disk, dataset or any other 'item') in MVS really 'belongs' to any job, started task, or TSO session. It is all per convention and the convention is managed by GRS. It does not lock users out – users (access methods, utilities, etc) 'agree' to announce their intention to access a resource before doing so. Any 'item' (a dataset, a storage address, etc) can be

protected in this way and, as long as all users abide by the convention, everything will be fine. When the request is received, GRS will see if a resource is currently in use elsewhere. If it is, GRS will coordinate the allocation of the request by managing the queue for it.

The convention is implemented by means of a request for a QNAME/RNAME combination. Any QNAME of up to 8 characters and any RNAME of up to 255 characters can be used. The request is done by means of an ENQ or RESERVE macro and it is released by means of a DEQ macro. Let's say we have developed a subsystem with a control block that is being shared by several tasks. We can then select a name of our own choice, say QNAME=CBLOCK. If there are two areas that can be updated simultaneously, we can have two RNAMEs, eg PART1 and PART2. To update the first field, a task would request an ENQ with QNAME=CBLOCK and RNAME=PART1 and with an EXCLUSIVE request. To read the first field, a task should then use the same names but with a SHR request. Many tasks will be allowed to proceed with the SHR allocation at the same time. As soon as one task requests an EXCLUSIVE ENQ, however, it will have to wait until none of the other tasks have it in use. Any SHR or EXCLUSIVE requests entering the system after the EXCLUSIVE request will wait in a queue. Even other tasks with a SHR request will fall in the queue behind it. It is a strictly first-come-first-served convention. Any task that does an EXCLUSIVE ENQ for CBLOCK/PART2 will be totally unaffected by this whole process and will participate in its own queueing process with other tasks that ask for exactly the same queue names. One task can also do more than one ENQ. This should give you an idea of the potential for a deadlock situation: task A has resource 1 and waits for resource 2, task B has resource 2 and waits for resource 1. This is a more common scenario in a sysplex than one would like to have and often results in one or more tasks having to be cancelled. More about this later.

Here is an example of how to ask for EXCLUSIVE use of the names SYSDSN/MY.LOADLIB

```
MODESET MODE=SUP,KEY=ZERO
ENQ  (QNAME,RNAME,E,44,SYSTEMS),RET=USE
LTR    R15,R15          .Did we get the resource?
BZ     GOTIT                 .Yes
WTO    'MY.LOADLIB is already in use…',ROUTCDE=11
B          ……
```

```
GOTIT    WTO   'No other task can now allocate MY.LOADLIB',ROUTCDE=11
         B       ……..
RNAME    DC     CL8'SYSDSN'
QNAME    DC    CL44'MY.LOADLIB'
```

The third parameter, 'E', indicates EXCLUSIVE. This could have been SHR. The '44' is the length of the QNAME and the SYSTEMS parameter indicates that this is a sysplex-wide request. This could also have been STEP or SYSTEM. The RET parameter can have the value CHNG, HAVE, TEST, USE, or NONE. CHNG will change the status from SHR to EXCLUSIVE or the other way around, HAVE is a conditional request and the resource should only be allocated if not in use, TEST tests the availability, USE asks for the request to be granted only if the name pair is not in use, and NONE (the default) is an unconditional request: the task will wait for the names (resource) until it becomes available.

DATASET PROTECTION

Dataset protection is obviously a major user of GRS and for that purpose we will look at it in more detail.

A job will, for instance, request that it is allowed a resource controlled by the name of SYSDSN/SYS1.PARMLIB, and that it wants it EXCLUSIVEly. GRS does not know that it relates to a dataset and is not interested in it – it simply does a check against the name pair and keeps a record of assigning it. If the pair is already in use, GRS will decline the request by means of a non-zero return code on an ENQ or RESERVE macro. GRS does not actually stop another user from updating SYS1.PARMLIB directly. The users (ALLOCATION in this case) have agreed amongst themselves to use a common name and to then, based on the return code received, either proceed, wait, or terminate. You can test this for yourself – write an ENQ macro with SYSDSN/SYS1.PARMLIB and EXCLUSIVE without actually allocating the dataset in your job. Then try to edit or rename the dataset. You will definitely get a 'Dataset in use' message, even though your job is not even accessing the dataset. Unless you wish to specifically block certain accesses, it is a good idea to make sure you do not use names used by standard MVS components when you have to use GRS for your own application purposes.

Here is the crunch of the convention: if anybody writes an access method that does not stick to the rules and simply opens SYS1.PARMLIB for updating, neither MVS nor GRS will intervene on behalf of the other users using or waiting on the resource. This user then breaks the rules by not participating in the convention and can expect dire consequences for itself and others. Two different users may then end up starting I/O to the same area on disk with the one overwriting output from the other.

A common question is: why is it not possible to delete SYS1.LINKLIB on the alternate system pack? After all, it is not in use. This is actually a good way to demonstrate how the ENQ works for datasets. As mentioned, the name pair 'agreed' on by all users to be used to update SYS1.PARMLIB would be SYSDSN/SYS1.PARMLIB. Note that the volser is not part of this name. Because SYS1.LINKLIB is in LLA, MVS itself has a SYSDSN/SYS1.LINKLIB ENQ with SHR against the name. By trying to delete SYS1.LINKLIB on the alternate resvol, ALLOCATION will actually attempt an EXCLUSIVE ENQ on the same name and get a non-zero return code. It then interprets this as the dataset being in use and declines your request. (GRS does not indicate that there is a 'dataset in use' condition as such, it shows that there is already an ENQ on the QNAME/RNAME and allocation assumes from this that the dataset is in use.)

This is why most systems programmers have a ZAPOFF program lying around. It would normally ZAP the VTOC to change the name somewhere and then use a standard utility to delete the new name. Let's say we have zapped SYS1.LINKLIB to SYS#.LINKLIB. When we now try to delete it, an EXCLUSIVE ENQ is done against SYSDSN/SYS#.LINKLIB. GRS indicates that the QNAME/RNAME is not in use and ALLOCATION will then allow the file to be allocated exclusively so that it can be deleted.

Another common misconception is the management of PDSs. Most jobs will allocate a PDS with a DISP=SHR in the JCL, yet update a member. For this it actually requires EXCLUSIVE usage of the dataset. Every time a member is updated, the PDS directory changes and this is a strictly one-at-a-time activity.

So why can't two users edit the same member of the same dataset at

the same time, in particular seeing that the PDS datasets are normally allocated with DISP=SHR? For this we should thank the developers of ISPF. Rather than just relying on the ENQ done by ALLOCATION, they have added a further convention that is adhered to by all ISPF users. An additional EXCLUSIVE ENQ is done with the name ISPFEDIT/SYS1.PARMLIB MEMBERNAME. All ISPF users who try to edit this member will do the same ENQ and ISPF will block a second user from updating the same member.

So what happens if you edit a member and another user submits a job with DISP=SHR to update the same member? The worst thing possible: the job will not participate in the ISPF convention and actually update the same member as you are editing it. (Feel free to try this on a PDS, but make sure it's your own because you will no doubt destroy the data.)

What happens if you run two IEBCOPY jobs at the same time updating the same member? They will happily proceed to simultaneously update the same member. So how does one overcome this problem? Well, unless one uses PDS/E datasets that allow multiple updates at the same time, the only trustworthy way is to actually code DISP=OLD on the DD-card. This is simply not feasible for most datasets because they are permanently in use, so great care should be taken where the same PDS is updated by multiple batch jobs or a mixture of batch jobs and ISPF users.

We will now have a look at GRS in a sysplex environment. We will start off by looking at a simple configuration and proceed to what a parallel sysplex has to offer.

In a basic non-sysplex configuration we could have two systems sharing disks without any GRS connectivity between them. Datasets in this kind of configuration are very much exposed. To update or even delete a dataset each system will do a 'local' ENQ. Its GRS will confirm that the name is not in use (unless of course there are other users of the same dataset on that system) and the dataset will be updated or deleted. You could literally delete a dataset from under another system whilst it is in use. You can then allocate another dataset where this one was, all without the other system having the faintest idea of all of this going on 'behind its back', so to say. So what will

happen in a case like this? Well, if you just delete the dataset from system A whilst it is in use by system B and don't create another one in exactly the same place, nothing may happen for quite some time. System B will happily use the data on the volume until it needs to refer to the datasets' description in the VTOC, where it will encounter the error. This will occur either when the dataset is closed or when it has to take another extent. If system A, however, writes data into the replacement dataset whilst in use by system B, the latter may suddenly get an I/O error because the data's format may have changed. This whole scenario is enough to show you why some form of hand-shaking between systems like this is required.

Before we look at setting up GRS between different systems, we need to have a look at what exactly a RESERVE is. A RESERVE is an ENQ with the added ability to activate a hardware block-out of other systems. MVS will actually instruct the hardware to block out all other systems from using this volume. Here is an example of how to write a RESERVE macro. This particular example, if run, will compete with the linkage editor and put a hardware RESERVE on the disk pointed at by the device number in the UCB. (The name SYSIEWLP is used by the QNAME by the linkage editor and we should avoid using it, this is just an example.)

```
            MODESET MODE=SUP,KEY=ZERO
         RESERVE   (SYSIEWLP,ENQDSN,E,44,SYSTEMS),UCB=UCBADDR
SYSIEWLP    DC     CL8'SYSIEWLP'
ENQDSN      DC     CL44'SYS5.LINKLIB'
UCBADDR     DS     F
```

This is a very effective but extremely inefficient way of protecting a volume. It is required when changes are made to the VTOCs' contents (or a VVDS or VTOC INDEX). This way, two systems won't update critical parts at the same time. Why is this inefficient? Look at it this way: we only want to make a small modification to part of the volume (the VTOC) and for that purpose we block the *entire* volume off from other systems. This is total overkill, yet very common at many sites. Some utilities, like the linkage editor, will actually do this as well, often with catastrophic performance implications. The RESERVE is still done with a MAJOR/MINOR name, eg SYSVTOC/VOL001, but in addition to that the RESERVE is also activated in the hardware by means of the UCB address of the device. Other users on the same

system may ask GRS about SYSVTOC/VOL001 and be informed that it is in use. Systems isolated from this one will be kept out by the hardware. Note that another user on the same system merely trying to read a dataset will ask GRS about SYSDSN/SYS1.PARMLIB and get access to SYS1.PARMLIB, even if it is on the same volume. (Both the MAJOR and MINOR names must match for an ENQ to return a non-zero return code.) Users of SYSDSN/SYS1.PARMLIB on the other system will get 'permission' from their GRS to go ahead but they won't be able to start an I/O to the RESERVEd volume.

So what protection does a RESERVE offer us? Very little. It will protect a volume against a simultaneous update from two or more systems. It will, however, not stop one system from deleting a dataset in use by another system. The only way to get that kind of protection is to have the GRSs from all the systems sharing disks communicate with each other. This is done by means of a GRS ring.

The most basic GRS ring consists of separate systems sending GRS tokens between themselves via Channel-To-Channel (CTC) connectors. This facility has been available for many years, but has largely been avoided because of the performance problems it created. With the advent of sysplex, GRS has been enhanced to use XCF signalling paths to communicate. This is substantially faster. With this, the option to use CTCs is still available, but performance can be further improved by making use of coupling facility structures. The information between systems is still sent in tokens, but this is done via high-speed coupling links. This has made GRS much faster and it is in use by all sysplexes. Before any of the systems use a resource, the GRS names are sent into the ring and each of the systems will indicate if the names are in use or not. This way we get dataset protection throughout the entire sysplex. GRS names are also used for other activities, eg HSM uses several names to serialize its activities throughout the sysplex and VSAM uses a number of names to protect shared catalogs. JES2, RACF, and the LOGGER are all major users of GRS.

All of these do, however, suffer from one weakness: if one of the systems in the ring does not respond, integrity is at risk. All the systems then stop GRS activity and the operator is prompted to confirm the status of the system not responding. Should the operator

indicate that the system is ‘down’, it will be removed from the GRS ring and put into a non-restartable wait state. (In a sysplex the operator intervention can be avoided with a Sysplex Failure Management (SFM) policy, which will make automatic decisions.) The only way to get a system to rejoin the ring is to IPL it. During the time that the operators’ response is awaited, a ‘hang’ condition will be experienced throughout the sysplex. Certain automation packages use extended MCS console, which in turn compete for GRS names with normal MVS consoles. In a worst-case scenario, a total deadlock can be experienced with the operator not being able to enter a command because of a console lockout. One or more systems may then have to be IPLed to break the deadlock.

With OS/390, this has been taken one step further. Rather than having tokens passed in a GRS ring, a STAR configuration can be defined. This is done in a coupling facility and the structure in the coupling facility becomes a repository to the environment. When an ENQ or RESERVE is done, the names are no longer passed to all of the systems. A look-up is done in the structure and, if the names are not in use, an entry is made into the structure and the ENQ is allowed. So, rather than having each system ‘inspect’ the request and then forward it to the next system, a central point of control is established where the usage is recorded and can be tested against. This also saves on real storage because the GRS tables are now kept in the coupling facility. This method is not affected by the number of systems in the sysplex because all the requests are handled by the coupling facility, with the result of a saving in processor overhead.

The way GRS names are managed through a ring or STAR is controlled by means of the GRSRNLxx member in SYS1.PARMLIB. This member has to be the same for all systems in the sysplex and if a system is IPLd into the sysplex with a different RNL parameter, it will actually enter a non-restartable wait state. Planning your RNL member is scope for another article because it requires some in-depth analysis, with deadlocks and data corruption as the constant threats. The parameters essentially allow you to convert system-wide ENQs to systems-wide ENQs and *vice versa*. It also lets you convert hardware RESERVEs. When a RESERVE is CONVERTed, MVS will no longer instruct the hardware to block out other systems from accessing the disk. The MAJOR/MINOR names are now passed

throughout the GRS ring or STAR and it becomes a convention that all the users of that disk participate in. The benefit is that we no longer lock up an entire disk for a single update and other datasets on the same volume can still be accessed by other systems at the same time. RNL parameters can be modified with system commands provided that the names involved in the modification are not in use at the time. This makes it sometimes impossible to make a dynamic change, eg any change that would involve SYSDSN and SYS1.* will never work because there are always ENQs on that QNAME/RNAME combination. If a command is entered to change the RNL, the system will wait for the ENQ to disappear – something that will never happen. Fortunately the RNL change request can be deleted in a case like this.

Just a short warning on changing RNL-statements in a sysplex. Any change that involves a resource that is always in use will have to be made at a time when all of the sysplex members are down. If we take them down one at a time they will not be able to rejoin the sysplex because the new RNL will be in conflict with what is currently in use by the other members. This will lead to the inevitable wait-state in the system you are trying to IPL, and there is simply no other way around it.

There are two aspects we still have to look at – problem detection and recovery. How do we know that there is an ENQ-problem and how can we resolve it, once detected?

Finding out that we have a problem is relatively easy, provided of course that the situation is not so bad as to prevent us from entering system commands. The most useful command is 'D GRS,C' and there are also other variations of this command. This command will list all QNMAME/RNAME combinations for which there is a form of contention. Some of the names involved are not easily recognizable. For this, you should refer to the *Diagnosis Guide*, where an indication is given of what the names are used for. One can also selectively display users of a certain name. An example is 'D GRS,RES=(SYSDSN,SYS1.LINK.*)'. This particular command will display all usages of datasets of which the two high-level qualifiers are SYS1.LINK. If the system is in a delayed situation, one should look out for major MVS components that are involved in these ENQ waits. HSM tends to do a large number of ENQs and RESERVEs and can get involved in delays. It may actually put an ENQ on a name or a

RESERVE on a VTOC and then need that same name or VTOC for another of its subtasks. It is not uncommon to see HSM with one subtask waiting on another, and it is sometimes necessary to cancel it to resolve the situation. This would normally be as a result of incorrectly-specified RNL parameters. There are several guidelines in the HSM documentation on how to specify RNL rules for HSM QNAMEs and RNAMEs and these should be followed closely.

Another troublesome ENQ deadlock situation would be one that involves waits on SYSMCS. As mentioned, some automation operations packages use extended MCS consoles and could do ENQs on names always in use by MVS console services. This name is also being used by TSO users when they enter commands via SDSF. This could be a major problem at times because it prevents the systems programmer from getting commands entered into the system to correct the problem. The intention during an emergency should be to make sure that MVS operating system components do not end up in queues behind started tasks, batch jobs, or TSO users. Although the 'D GRS' command goes a long way towards displaying the necessary information, there is also a macro that can be used to obtain more information. The name of this macro is 'GQSCAN' and it is the only published program interface with GRS. It is able to give a substantial amount of detail on tasks waiting on resources. This macro can easily be called from the MVS command exit and you can develop new commands to help you resolve certain more common ENQ problems your site may experience from time to time.

GRS is the central point of control for integrity in both data and operating system serialization. It is fast and efficient in that it does not consume large amounts of resources, particularly if used with at least XCF services and even more so if used in a STAR configuration. By regularly keeping an eye on it, you will get more familiar with bottlenecks that exist in your system from time to time. As systems are grouped together into sysplexes it is essential that you should be able to correctly analyse output and act with certainty in a deadlock situation. By knowing what is involved you would be able to prevent major outages on mission-critical systems and prevent situations where you have to randomly cancel jobs to try to get the systems to move again.

Displaying a dataset's last-accessed date

THE PROBLEM

One piece of information about a dataset that is most useful to the storage administrator is its date of last access. For a dataset on DASD, this is simply obtained from its format-1 DSCB, eg via ISPF/PDF option 3.4. For a dataset that is under HSM control and which has been migrated, this information is not so easily available because the dataset no longer resides on DASD and so no longer has a format-1 DSCB. Instead, the required information is contained in the HSM MCDS MCD record for the dataset. Whilst the HLIST command can be used to display the information, it presents a rather verbose output, and requires various parameters other than the dataset name to get the desired result.

THE SOLUTION

To make life easier, I have written a simple program, LADATE, that takes a fully-qualified dataset name from its parameter field and displays a simple one-line message on the terminal, giving the dataset's location (volser) and its last-accessed date, independently of its migration status:

```
FULLY.QUALIFIED.DATASET.NAME on VOLSER was last referenced on dd-mon-yy
```

For simplicity, LADATE was not written as a TSO command processor, so is most easily executed via a simple CLIST or REXX EXEC, an example of which is presented below.

LADATE FUNCTIONALITY

In outline, LADATE extracts the dataset name, assumed to be fully qualified, from the parameter field, and issues a LOCATE macro for it.

- If the resulting volser is not 'MIGRAT', an OBTAIN macro is issued to acquire the dataset's format-1 DSCB. The last-accessed date is then extracted from the DSCB.

- If the volser is 'MIGRAT', an HSM CDS read request Management Work Element (MWE) requesting the MCDS MCD record for the dataset is built and sent to HSM via the extended router SVC. On (successful) return, the migration volser and the original last-accessed date are extracted from the returned MCD record.

In either case, a message of the form shown above is built and sent to the terminal via a TPUT macro.

LADATE is very useful on the ISPF/PDF option 3.4 screen as it (the CLIST) can be used as a line command on multiple lines without having to specify any parameters (PDF supplies the dataset names automatically), and without having to check the migration status of each dataset first.

OPERATIONAL ENVIRONMENT

LADATE has no special authorization requirements, and may be link-edited into any suitable load library. The LADATE CLIST should be placed in SYS1.CMDPROC or any other suitable library in the SYSPROC concatenation.

LADATE was originally written for use with DFHSM 2.3 on an MVS/XA 2.2.3 system, and has since been used unchanged with DFHSM 2.6 on an MVS/ESA 4.2.2 system and DFSMShsm 1.2.0 on an MVS/ESA 5.1.0 system.

LADATE CLIST

```
PROC 1 dsname
   CALL "LOAD.LIBRARY(LADATE)" "dsname"
END
```

LADATE

```
         TITLE 'LADATE - Display a Dataset's Last-Accessed Date'
*********************************************************************
*  PROGRAM LADATE
*  ———————
*  Provides the caller with the date of last  access  of  the  dataset
*  whose (fully qualified) name is specified in the parm  field.  This
*  information is sent to the terminal via a TPUT macro - this program
*  is not intended for batch use.
```

```
*
*  If the dataset is migrated, the MCD record is obtained from HSM via
*  a CDS record read request,  and  the last referenced date extracted
*  from there.
*
*  This routine has worked on HSM versions 2.3.Ø through to 2.6.Ø, and
*  DFSMShsm 1.2.Ø, under MVS/XA 2.2.3 through to MVS/ESA 5.1.Ø.
*
*  Operational requirements :
*
*    STATE    : Problem
*    KEY      : 8
*    APF      : NO
*    AMODE    : 31
*    RMODE    : 24
*    LOCATION : Private load library
*********************************************************************
*
         EJECT
LADATE   CSECT
LADATE   AMODE 31
LADATE   RMODE 24
*
RØ       EQU   Ø
R1       EQU   1                  * PARM FIELD ADDRESS ON ENTRY
R2       EQU   2                  * WORK REGISTER
R3       EQU   3                  * WORK REGISTER
R4       EQU   4                  * WORK REGISTER
R5       EQU   5                  * WORK REGISTER
R6       EQU   6                  * WORK REGISTER
R7       EQU   7                  *
R8       EQU   8                  * WORK REGISTER
R9       EQU   9                  * WORK REGISTER
R1Ø      EQU   1Ø                 * MCD RECORD ADDRESS
R11      EQU   11                 * DSCB/MWE ADDRESS
R12      EQU   12                 * BASE REGISTER
R13      EQU   13                 * OUR SAVEAREA
R14      EQU   14                 * RETURN ADDRESS
R15      EQU   15                 * ENTRY ADDRESS
*
         USING *,R15              * ADDRESSABILITY
         B     START              * BRANCH TO START OF CODE
         DC    AL1(LASTL-FIRSTL)  * LENGTH OF HEADER TEXT
FIRSTL   EQU   *
         DC    CL8'LADATE  '
LASTL    EQU   *
         DC    C' '
         DC    CL8'&SYSDATE'
         DC    C' '
         DC    CL8'&SYSTIME'
```

```
         DROP  R15                 * FINISHED WITH R15
         DS    ØF                  * ALIGN TO FULL WORD BOUNDARY
         EJECT
*
*********************************************************************
* ADDRESSABILITY AND LINKAGE
*********************************************************************
*
START    EQU   *
         STM   R14,R12,12(R13)     * SAVE REGISTERS IN CALLERS SAVEAREA
         LR    R12,R15             * LOAD BASE REGISTER
         USING LADATE,R12          * AND DEFINE ADDRESSIBILITY
*
         LR    R11,R13             * R11 = ADDRESS OF CALLERS SAVEAREA
         LA    R13,SAVEAREA        * R13 = ADDRESS OF OUR     SAVEAREA
         ST    R11,4(R13)          * STORE HSA ADDRESS
         ST    R13,8(R11)          * STORE LSA ADDRESS
*
* EXTRACT DATASET NAME FROM PARM FIELD
*
         L     R1,Ø(R1)            * R1 = ADDRESS OF PARM FIELD
         LH    R2,Ø(R1)            * R2 = LENGTH OF PARM FIELD
         LTR   R3,R2               * SAVE AND TEST VALUE
         BZ    NOPARM              * ERROR IF NO PARM SPECIFIED
         BCTR  R3,Ø                * LENGTH OF PARM FOR EXECUTE
*
         MVI   MSGDSN,C' '         * BLANK OUT ...
         MVC   MSGDSN+1(43),MSGDSN * ... DSN FIELD
         EX    R3,MOVEDSN          *     ... AND MOVE IN NON-BLANK BIT
         B     LOCATE              * JUMP OVER EXECUTED MVC
MOVEDSN  MVC   MSGDSN(1),2(R1)     * MOVE DSN
         EJECT
*
*********************************************************************
* PROCESS THE REQUEST
*********************************************************************
*
* LOCATE THE DATASET IN THE CATALOG
*
LOCATE   EQU   *
         LOCATE LOCDATA            * EXECUTE LOCATE MACRO
         LTR   R15,R15             * TEST RETURN CODE
         BNZ   LOCERR              * BRANCH IF NOT ZERO
*
         CLC   VSERIAL,MIGRAT      * IS DATASET MIGRATED ?
         BE    MIGRATED            * YES, SO SKIP THE OBTAIN BIT
         EJECT
*
*-------------------------------------
* DATASET IS ON-LINE ...
*-------------------------------------
* OBTAIN THE FORMAT-1 DSCB FROM THE VTOC ...
*
```

```
ONLINE   EQU   *
         OBTAIN  OBTDATA           * EXECUTE OBTAIN MACRO
         LTR   R15,R15             * TEST RETURN CODE
         BNZ   OBTERR              * BRANCH IF NOT ZERO
*
* ... AND EXTRACT THE LAST-ACCESSED DATE FROM THE FORMAT-1 DSCB
*
         LA    R11,WORKAREA        * DEFINE DSCB ...
         USING DS1FMTID,R11        * ... ADDRESSABILITY
*
         MVC   MSGVSN,VSERIAL      * MOVE VSN INTO MESSAGE
         SR    R5,R5               * ZERO R5 ...
         IC    R5,DS1REFD          * ... INSERT YY OF LAST REF ...
         LTR   R6,R5               *     ... AND SAVE/TEST ITS VALUE
         BNZ   GOODDATE            * NON-ZERO IS GOOD
         MVC   MSGDAY(9),NULLDATE  * IF ZERO, INSERT NULL DATE TEXT
         B     TPUTMSG             * AND GO AND WRITE MESSAGE
*
GOODDATE EQU   *
         SR    R4,R4               * ZERO R4 ...
         ICM   R4,3,DS1REFD+1      * ... AND INSERT DDDD OF LAST REF
         B     CONVDATE            * NOW GO AND SORT THE DATE OUT
*
         DROP  R11                 * FINISHED WITH DSCB
         EJECT
*-------------------------------------
* DATASET IS MIGRATED ...
*-------------------------------------
* BUILD A 'READ CDS RECORD' MWE FOR THE DATASET'S MCD RECORD ...
*
MIGRATED EQU   *
         LA    R8,MWE              * R8  = MWE ADDRESS
         L     R9,MWESIZE          * R9  = MWE LENGTH
         LR    R1Ø,R8              * R1Ø = MWE ADDRESS
         SR    R11,R11             * R11 = ZERO
         MVCL  R8,R1Ø              * CLEAR THE MWE
*
         LA    R11,MWE             * DEFINE MWE ...
         USING ARCMWE,R11          * ... ADDRESSABILITY
*
         MVC   MWELEN,MWESIZE+1    * INSERT MWE LENGTH INTO MWE
         LA    R8,1                * SET R8 TO 1 ...
         STH   R8,MWEMCNT          * ... AND SET MWE COUNT TO 1
         MVC   MWEFUNC,FCODE       * INSERT FUNCTION CODE IN MWE
         MVC   MWECETYP,CDSRTYPE   * INSERT CDS RECORD TYPE
         MVC   MWEDSN,MSGDSN       * INSERT DATASET NAME
         MVC   MWEBUFL,MCDLEN      * INSERT MCD LENGTH
*
* ... PASS THE REQUEST TO HSM SVC VIA THE EXTENDED ROUTER ...
*
         SR    RØ,RØ               * ZERO RØ ...
```

```
         IC    RØ,MWEFUNC          * ... AND INSERT FUNCTION CODE
         LR    R1,R11              * R1  = MWE ADDRESS
         LA    R15,24              * R15 = HSM ROUTING CODE
         SVC   1Ø9                 * CALL EXTENDED ROUTER
         LTR   R15,R15             * TEST SVC RETURN CODE
         BNZ   ROUTERR             * IF NOT ZERO ROUTER FAILED
         L     R15,MWERC           * GET HSM RETURN CODE
         LTR   R15,R15             * TEST VALUE
         BNZ   HSMERR              * IF NOT ZERO HSM FAILED
         DROP  R11                 * FINISHED WITH MWE
*
* ... AND EXTRACT THE MIGRATION VOLSER AND LAST ACCESSED DATE FROM THE
*     RETURNED MCD RECORD.
*
         LA    R1Ø,MCD             * DEFINE MCDS RECORD ...
         USING ARCMCD,R1Ø          * ... ADDRESSABILITY
         TM    MCDFLGS,MCDFASN     * MIGRATED DATASET EXISTS ?
         BNO   MIGERR              * NOPE
         MVC   MSGVSN,MCDVSN       * MOVE MIGRATION VSN INTO MESSAGE
         SR    R4,R4               * ZERO R4
         ST    R4,DBLWORD          * AND FIRST HALF OF DOUBLEWORD
         MVC   DBLWORD+4(4),MCDDLR * MOVE IN LAST REF DATE (ØØYYDDDS)
         CVB   R5,DBLWORD          * CONVERT IT TO BINARY YYDDD
         D     R4,F1ØØØ            * DIVIDE BY 1ØØØ -> R4=DDD, R5=YY
         LTR   R6,R5               *     ... AND SAVE/TEST ITS VALUE
         BNZ   CONVDATE            * NON-ZERO IS GOOD
         MVC   MSGDAY(9),NULLDATE  * IF ZERO, INSERT NULL DATE TEXT
         B     TPUTMSG             * AND GO AND WRITE MESSAGE
         DROP  R1Ø                 * FINISHED WITH MCDS RECORD
         EJECT
*-------------------------------------
* THE LAST ACCESSED DATE IS NOW AVAILABLE. BUILD AND ISSUE MESSAGE.
*-------------------------------------
* ADJUST DAYS-IN-MONTH TABLE TO TAKE LEAP YEAR INTO CONSIDERATION
*
CONVDATE EQU   *
         N     R5,F3               * YEAR MODULO 4
         IC    R5,FEBDAYS(R5)      * GET NUMBER OF DAYS IN FEBRUARY
         STC   R5,DAYMONTH+5       * AND SAVE IN TABLE
* FIND WHICH MONTH WE ARE IN
         LA    R5,DAYMONTH-4       * R5 = DAYMONTH TABLE POINTER
MLOOP    EQU   *
         LA    R5,4(R5)            * POINT AT NEXT MONTH IN TABLE
         SH    R4,Ø(R5)            * SUBTRACT DAYS FOR THIS MONTH
         BP    MLOOP               * LOOP WHILE DDD STILL POSITIVE
         AH    R4,Ø(R5)            * R4 = DAY
         LH    R5,2(R5)            * R5 = OFFSET IN MONTH TO MONTH NAME
* CONVERT DD, MM, AND YY TO 'DD-MMM-YY' FORMAT.
* R4 = DD, R5 = @MONTH NAME, R6 = YY
         CVD   R4,DBLWORD            * CONVERT DAY TO PACKED DECIMAL
         MVC   FULLWORD,DBLWORD+4    * STORE IN FULLWORD
```

```
         UNPK  DBLWORD,FULLWORD+2(2) * AND UNPACK IT
         OI    DBLWORD+7,X'FØ'       * TREAD ON SIGN
         MVC   MSGDAY,DBLWORD+6      * MOVE RESULT INTO MESSAGE
         LA    R7,MONTHS             * LOAD R7 WITH ADDRESS OF MONTHS
         AR    R7,R5                 * ADD OFFSET
         MVC   MSGMONTH,Ø(R7)        * AND MOVE MONTH NAME INTO MESSAGE
         CVD   R6,DBLWORD            * CONVERT YEAR TO PACKED DECIMAL
         MVC   FULLWORD,DBLWORD+4    * STORE IN FULLWORD
         UNPK  DBLWORD,FULLWORD+2(2) * AND UNPACK IT
         OI    DBLWORD+7,X'FØ'       * TREAD ON SIGN
         MVC   MSGYEAR,DBLWORD+6     * MOVE RESULT INTO MESSAGE
* SQUEEZE BLANKS OUT OF MESSAGE AND TPUT IT TO THE TERMINAL
TPUTMSG  EQU   *
         LA    R3,MESSAGE           * MESSAGE ADDRESS
         LR    R4,R3                * DITTO
         AR    R4,R2                * MOVE PAST DSN
         MVC   Ø(LMSGTXT,R4),MSGTXT * SHUFFLE MSGTXT UP TO MSGDSN
         LA    R4,LMSGTXT           * R2 = LENGTH ...
         AR    R2,R4                * ... OF COMPLETE MESSAGE
         TPUT  (R3),(R2)
         EJECT
*********************************************************************
* ALL DONE, SO STORE RETURN CODE AND RETURN
*********************************************************************
RETURN   EQU   *
         L     R13,4(R13)          * RESTORE ADDRESS OF CALLERS SA
         L     R14,12(R13)         * RESTORE RETURN ADDRESS
         SLR   R15,R15             * RETURN CODE IS ALWAYS ZERO
         LM    RØ,R12,2Ø(R13)      * RESTORE RØ - R12
         BR    R14                 * AND RETURN
         EJECT
*********************************************************************
* ERROR CONDITIONS
*********************************************************************
NOPARM   EQU   *                   * DSN LENGTH INVALID
         TPUT  NOPMSG,L'NOPMSG     * WRITE ERROR MESSAGE
         LA    R15,4               * SET RETURN CODE =  4
         B     RETURN              * AND BRANCH TO RETURN
NOPMSG   DC    C'A dataset name must be specified'
         DS    ØH
LOCERR   EQU   *                   * LOCATE ERROR
         TPUT  LOCMSG,L'LOCMSG     * WRITE ERROR MESSAGE
         LA    R15,8               * SET RETURN CODE =  8
         B     RETURN              * AND BRANCH TO RETURN
LOCMSG   DC    C'The requested dataset is not cataloged'
         DS    ØH
OBTERR   EQU   *                   * OBTAIN ERROR
         TPUT  OBTMSG,L'OBTMSG     * WRITE ERROR MESSAGE
         LA    R15,12              * SET RETURN CODE = 12
         B     RETURN              * AND BRANCH TO RETURN
OBTMSG   DC    C'The requested dataset does not exist on the volume spe+
```

```
               cified in the catalog'
          DS   ØH
ROUTERR   EQU  *                    * EXTENDED ROUTER FAILED
          TPUT ROUTMSG,L'ROUTMSG    * WRITE ERROR MESSAGE
          LA   R15,16               * SET RETURN CODE = 16
          B    RETURN               * AND BRANCH TO RETURN
ROUTMSG   DC   C'The extended router SVC failed to contact HSM'
          DS   ØH
HSMERR    EQU  *                    * HSM REQUEST REJECTED
          TPUT HSMMSG,L'HSMMSG      * WRITE ERROR MESSAGE
          LA   R15,2Ø               * SET RETURN CODE = 2Ø
          B    RETURN               * AND BRANCH TO RETURN
HSMMSG    DC   C'HSM rejected the CDS read request'
          DS   ØH
MIGERR    EQU  *                    * NO MIGRATED COPY OF D/S EXISTS
          TPUT MIGMSG,L'MIGMSG      * WRITE ERROR MESSAGE
          LA   R15,24               * SET RETURN CODE = 24
          B    RETURN               * AND BRANCH TO RETURN
MIGMSG    DC   C'Dataset is flagged as migrated but no migration copy e+
               xists'
          DS   ØH
          EJECT
*
*********************************************************************
* CONSTANTS, VARIABLES AND DATA AREAS
*********************************************************************
*
          DS   ØD
          DC   CL8'SAVEAREA'
SAVEAREA  DS   18F
F3        DC   F'3'
F1ØØØ     DC   F'1ØØØ'
MWESIZE   DC   AL4(MWEL-MWE)        * MWE LENGTH IN BYTES
MCDLEN    DC   AL2(MWEL-MCD)        * RETURNED MCDS RECORD LENGTH
FCODE     DC   XL1'Ø8'              * 'READ CONTROL DATASET RECORD'
CDSRTYPE  DC   CL1'D'                * TYPE D = MCDS DATASET RECORD
          DS   ØH
FEBDAYS   DC   X'1D1C1C1C'
DAYMONTH  DC   X'ØØ1FØØØØØØ1CØØØ3ØØ1FØØØ6ØØ1EØØØ9ØØ1FØØØCØØ1EØØØF'
          DC   X'ØØ1FØØ12ØØ1FØØ15ØØ1EØØ18ØØ1FØØ1BØØ1EØØ1EØØ1FØØ21'
MONTHS    DC   CL36'JanFebMarAprMayJunJulAugSepOctNovDec'
NULLDATE  DC   CL9'ØØ-ØØØ-ØØ'
MIGRAT    DC   CL6'MIGRAT'
MESSAGE   EQU  *
MSGDSN    DC   CL44' '
MSGTXT    DC   CL4' on '
MSGVSN    DS   CL6
          DC   CL24' was last referenced on '
MSGDAY    DS   CL2
          DC   CL1'-'
MSGMONTH  DS   CL3
          DC   CL1'-'
```

```
MSGYEAR  DS    CL2
LMSGTXT  EQU   *-MSGTXT
*
DBLWORD  DS    D
FULLWORD DS    F
*
LOCDATA  CAMLST NAME,MSGDSN,,CAMWORK
*
OBTDATA  CAMLST SEARCH,MSGDSN,VSERIAL,WORKAREA
*
CAMWORK  DS    ØD                  * CAMLST WORK AREA
         DC    265CL1' '
         ORG   CAMWORK
VCOUNT   DS    H                   * VOLUME COUNT
VDEVCODE DS    XL4                 * DEVICE CODE
VSERIAL  DS    CL6                 * VOLUME SERIAL
DSNSEQ   DS    H                   * DATASET SEQUENCE NUMBER
WORKAREA DS    CL238
DSCBTTR  DS    CL3                 * TTR OF DSCB
         ORG
         DS    ØD
         DC    CL4'MWE '
MWE      DS    72F                 * MWE
MCD      DS    12ØF                * RETURNED MCDS RECORD
MWEL     EQU   *
         EJECT
*-------------------------------------
* DSECTS
*-------------------------------------
ARCMWE   DSECT
         DS    CL9
MWELEN   DS    AL3                 * MWE LENGTH
         DS    3F
MWEFUNC  DS    XL1                 * MWE FUNCTION CODE
         DS    CL23
MWERC    DS    F                   * HSM RETURN CODE
         DS    CL88
MWEMCNT  DS    H                   * NUMBER OF MWES IN REQUEST
         DS    3H
MWEDSN   DS    CL44                * DATASET NAME
         DS    CL26
MWECETYP DS    XL1                 * CDS ENTRY TYPE FOR FUNC CODE 8
         DS    CL65
MWEBUFL  DS    H                   * CDS READ BUFFER LENGTH
MWEBUFU  DS    H                   * CDS READ BUFFER UTILISATION
*
ARCMCD   DSECT
MCK      DS    CL44                * MCDS RECORD KEY (LEVEL Ø DSNAME)
MCH      DS    XL2Ø                * CDS RECORD HEADER
MCDVSN   DS    CL6                 * MIGRATION VOLUME SERIAL NUMBER
MCDFLGS  DS    XL2                 * FLAGS
         DS    CL4
```

```
MCDDLC   DS    F                  * DATASET CREATION DATE
MCDTLR   DS    F                  * TIME LAST REFERENCED
MCDDLR   DS    F                  * DATE LAST REFERENCED
         DS    392C
*
MCDFASN  EQU   X'80'              * MIGRATED DATASET EXISTS
*
DSCB     DSECT
         IECSDSL1 (1)             * FORMAT-1 DSCB MAPPING MACRO
*
         END
```

Peter Wright
Associate Consultant
Tessella Support Services (UK)

Locating a load module the easy way

INTRODUCTION

On an MVS system a load module can reside in any or all of several places – in the Link Pack Area (LPA), in SYS1.LINKLIB, or a member of the linklist concatenation, in a JOB or STEP library, or in a private load library such as a member of the ISPLLIB concatenation. If one is not sure from where a given module is being loaded and needs to know (eg if copies exist in several places), the program described here should be of some use.

THE LMOD PROGRAM

The LMOD program, written as a TSO command processor, may be invoked with just the name of the load module of interest, eg:

```
LMOD IEFUTL
```

or with the load module name and the DDNAME of a private library/ private library concatenation where it is suspected the module may be located, eg:

```
LMOD AMSP0000 DDNAME(ISPLLIB)
```

The processing performed by LMOD consists of the following steps:

1 The command is parsed via a call to IKJPARS.

2 If no DDNAME was specified, processing jumps to step 4 below.

3 If a DDNAME was specified, it is OPENed and a BLDL macro issued for the specified member name. The DDNAME is then CLOSEd, and if the module was found, processing jumps to step 8 below. If the module was not found, processing continues with the next step.

4 The active LPA queue is scanned for a Contents Directory Entry (CDE) for the module. If one is found, and it is a minor CDE (ie the module name is an alias), the corresponding major CDE is located and the true name of the module extracted. The module's Extent List (XTLST) is located and its length and start address extracted. Messages of the form:

```
MODNAME  is length bytes at address on active LPA Queue (alias of TRUENAME)
         Relocated entry point at address is AMODE am
```

are built and issued via TPUTs to the terminal. The (alias of …) text is only present if MODNAME is an alias. The entry point address comes from the module's CDE, and the AMODE (31 or 24) is deduced from the entry point address.

5 The LPA Directory is scanned for the module via a call to the LPA Directory Scan Routine, IEAVVMSR, the address of which is in CVTLPDSR. If a Link Pack Directory Entry (LPDE) is found, and is a minor (alias) LPDE, a second scan is made for the corresponding major LPDE for the module's true name. Having found the LPDE(s), messages of the form:

```
MODNAME  is length bytes at address in LPA Directory (alias of TRUENAME)
         Relocated entry point at address is AMODE am
```

are built and issued. In this case all the required information is contained in the LPDE(s).

6 If the name of the module starts 'IGC', it could be an SVC. If the name is of the form 'IGCnnn', where 'nnn' is between 000 and 255, it could be a Nucleus SVC; if the name is of the form

IGC00nnn it could be an LPA-resident SVC. In either case, the SVC table entry corresponding to the value of nnn is located and the data therein is used to build message(s) of the form:

```
MODNAME  is SVC nnn (Type n), entry point at address
MODNAME  is an alias of TRUENAME
```

The second message is issued in the case of a module that has already been found as an alias in the LPA by the previous steps, and hence whose true name is also known.

7 A linklist/STEPLIB BLDL is issued to locate any copy of the module in either of these locations. If nothing is found, processing jumps to step 9 below.

8 At this point, a BLDL entry exists, either for the linklist/STEPLIB, or for the private library if we have jumped here directly from step 3.

If the BLDL entry indicates that the module is in SYS1.LINKLIB, a message of the form:

```
MODNAME  found as a type name at TTR ttttrr in SYS1.LINKLIB
```

where type is ‘major’ if MODNAME is the module’s true name, or ‘minor’ if it is an alias, and ttttrr is the relative track/record location of the module in SYS1.LINKLIB.

If the BLDL entry indicates that the module is in a linklist dataset other than SYS1.LINKLIB, the linklist table is scanned for the appropriate entry, the name of the dataset extracted, and a LOCATE issued to get its catalog entry. If the module is not in the linklist, it must either be in the STEPLIB concatenation or the private library concatenation. In either case, the TIOT is scanned for the appropriate DDNAME and the required member of the concatenation located. The address of the dataset’s JFCB is extracted from the TIOT entry and the name of the dataset and its DASD volser extracted from the JFCB. Finally, messages of the form:

```
MODNAME  found as a type name at TTR ttttrr in libtype (nn)
         DSN=dataset.name on volser
```

are built and issued. As before, type is ‘major’ or ‘minor’ and ttttrr is the module’s location in the dataset. In addition, libtype is

either 'JOB/STEP library' or 'private library' as appropriate, and nn is the position of the dataset in the concatenation (00 is the first, 01 the second, etc).

In all cases these messages are followed by a message of the form:

```
Module length is length, RMODE rm; entry point at offset is AMODE am
```

and, if the module name is an alias, by the message :

```
MODNAME  is an alias of TRUENAME, entry point at offset
```

where rm is 'ANY' or '24', and am is '31' or '24', as indicated by the BLDL entry.

9 This completes the search for the module. If it was not found in any of the searched locations, a message of the form:

```
MODNAME  not found in LPA/LINKLIST/Private library
```

is issued.

10 The program cleans up and exits.

OPERATIONAL ENVIRONMENT

LMOD has no special authorization requirements, and should be link-edited with the RENT attribute into SYS1.CMDLIB or another suitable linklist library. LMOD was written for use on an MVS/ESA 5.1.0 system, but should work on earlier versions.

LMOD

```
         TITLE 'LMOD: Locate a load module'
*********************************************************************
*                                                                   *
*  LMOD                                                             *
*  ——                                                               *
*  Invoked under TSO,  this  program provides a quick + easy way of *
*  finding where a module is being loaded from - ie LPA, Linklist,  *
*  STEPLIB, or private library.                                     *
*                                                                   *
*  EG    LMOD IEFUTL                                                *
*        LMOD ALLOCATE                                              *
*   OR   LMOD AMSP0000 DDNAME(ISPLLIB)                              *
*                                                                   *
*  ENVIRONMENT                                                      *
*                                                                   *
*    State    : Problem                                            *
```

```
*    Key      : 8                                                      *
*    APF      : No                                                     *
*    AMODE    : 31                                                     *
*    RMODE    : 24                                                     *
*    Location : Linklist or STEPLIB                                    *
*                                                                      *
************************************************************************
*
         EJECT
LMOD     CSECT
LMOD     AMODE 31
LMOD     RMODE 24
*
RØ       EQU   Ø                        *
R1       EQU   1                        * @(CPPL) ON ENTRY
R2       EQU   2                        * WORK REGISTER
R3       EQU   3                        * WORK REGISTER
R4       EQU   4                        * WORK REGISTER
R5       EQU   5                        * WORK REGISTER
R6       EQU   6                        *
R7       EQU   7                        *
R8       EQU   8                        *
R9       EQU   9                        *
R1Ø      EQU   1Ø                       * @(PPL)/(@PDE)
R11      EQU   11                       * @(CPPL)/@(PDL)/@(DCB)/@(CVT)
R12      EQU   12                       * BASE REGISTER
R13      EQU   13                       * SAVEAREA/WORKAREA ADDRESS
R14      EQU   14                       * RETURN ADDRESS
R15      EQU   15                       * ENTRY ADDRESS
*
         USING *,R15                    * ADDRESSABILITY
         B     START                    * BRANCH TO START OF CODE
         DC    AL1(LASTL-FIRSTL)        * LENGTH OF HEADER TEXT
FIRSTL   EQU   *
         DC    C'LMOD    '
LASTL    EQU   *
         DC    C' '
         DC    CL8'&SYSDATE'
         DC    C' '
         DC    CL5'&SYSTIME'
         DROP  R15                      * FINISHED WITH R15
         DS    ØF                       * ALIGN TO FULL WORD BOUNDARY
*
************************************************************************
* ADDRESSABILITY AND LINKAGE - RE-ENTRANT FORM
************************************************************************
*
START    EQU   *
         STM   R14,R12,12(R13)          * SAVE REGISTERS IN HSA
         LR    R12,R15                  * LOAD BASE REGISTER
         USING LMOD,R12                 * AND DEFINE ADDRESSABILITY
```

```
*
         LR    R11,R1                    * SAVE @(CPPL)
*
         LA    RØ,LWKAREA                * REQUIRED WORKAREA LENGTH
         GETMAIN RU,LV=(RØ),BNDRY=PAGE,SP=78,LOC=(BELOW,ANY)
*
         LR    R2,R1                     * CLEAR ...
         LA    R3,LWKAREA                * ... WORK ...
         LR    R4,R2                     *     ... AREA ...
         SR    R5,R5                     *         ... TO ...
         MVCL  R2,R4                     *             ... ZEROS
*
         ST    R13,4(R1)                 * STORE HSA ADDRESS
         ST    R1,8(R13)                 * STORE LSA ADDRESS
         LR    R13,R1                    * R13 = OUR SAVEAREA ADDRESS
         USING WORKAREA,R13              * WORKAREA ADDRESSABILITY
*
         MVI   PRIVLIB,C'N'              * INITIALISE PRIVLIB FLAG
         LA    R1,4                      * INITIALISE ...
         ST    R1,RETCODE                * ... RETCODE
         MVC   DDNAME,STEPLIB            * INITIALISE DDNAME
         MVC   TRUENM,BLANKS             * INITIALISE TRUENM
         EJECT
*********************************************************************
* PARSE THE INPUT
*********************************************************************
         USING CPPL,R11                  * CPPL ADDRESSABILITY
*
         LA    R1Ø,STPPL                 * PARSE PARAMETER LIST ...
         USING PPL,R1Ø                   * ... ADDRESSABILITY
*
         L     R1,CPPLUPT                * STORE @(UPT) ...
         ST    R1,PPLUPT                 * ... IN THE PPL
*
         L     R1,CPPLECT                * STORE @(ECT) ...
         ST    R1,PPLECT                 * ... IN THE PPL
*
         LA    R1,CPECB                  * STORE @(ECB) ...
         ST    R1,PPLECB                 * ... IN THE PPL
         XC    CPECB,CPECB               * CLEAR THE ECB
*
         L     R1,PARSADDR               * STORE @(@(PCL)) ...
         ST    R1,PPLPCL                 * ... IN THE PPL
*
         LA    R1,APDL                   * STORE @(@PDL)) ...
         ST    R1,PPLANS                 * ... IN THE PPL
         XC    APDL,APDL                 * ZERO THE DSECT POINTER
*
         L     R1,CPPLCBUF               * STORE @(COMMAND BUFFER) ...
         ST    R1,PPLCBUF                * ... IN THE PPL
*
         CALLTSSR EP=IKJPARS,MF=(E,STPPL)
```

```
*
         LTR   R15,R15                  * PARSE OK ?
         BNZ   RETURN                   * OH DEAR
*
         DROP  R1Ø,R11                  * FINISHED WITH PPL, CPPL
         EJECT
*
*********************************************************************
* PROCESS THE REQUEST
*********************************************************************
*
         L     R11,APDL                 * PARSE DESCRIPTOR LIST ...
         USING IKJPARMD,R11             * ... ADDRESSABILITY
*
         LA    R1Ø,MEMPDE               * @(MEMBER NAME PDE)
         USING PDE,R1Ø                  * PDE ADDRESSABILITY
*
         MVC   NAME,BLANKS              * SET NAME TO BLANKS
         L     R1,PDEOPADR              * R1 = @(MEMBER NAME)
         LH    R2,PDEOPLEN              * R2 = L'(MEMBER NAME)
         ST    R2,LNAME                 * SAVE IT FOR LATER
         BCTR  R2,RØ                    * COPY ...
         EX    R2,MOVENAME              * ... MEMBER NAME
*
         LA    R1Ø,DDNPDE               * @(DDNAME PDE)
         TM    PDEFLAGS,PDEFPRES        * WAS DDNAME SPECIFIED ?
         BZ    NODDN                    * JUMP IF NOT
*
* A DDNAME WAS SUPPLIED - OPEN IT AND DO A BLDL FOR THE MODULE
*
         MVC   DDNAME,BLANKS            * SET DDNAME TO BLANKS
         L     R1,PDEOPADR              * R1 = @(DDNAME)
         LH    R2,PDEOPLEN              * R2 = L'DDNAME
         BCTR  R2,RØ                    * COPY ...
         EX    R2,MOVEDDN               * ... DDNAME
*
         DROP  R1Ø,R11                  * FINISHED WITH PDL, PDE
*
         MVC   DCBW(LDCB),DCB           * MOVE DCB TO WORKAREA
         LA    R11,DCBW                 * R11 = DCB ADDRESS
         USING IHADCB,R11               * DEFINE DCB ADDRESSABILITY
*
         MVC   DCBDDNAM,DDNAME          * MOVE DDNAME INTO DCB
*
         MVC   OPENW(LOPENL),OPENL              * OPEN ...
         OPEN  ((R11),INPUT),MODE=31,MF=(E,OPENW) * ... DDNAME
*
         TM    DCBOFLGS,DCBBIT3         * BIT 3 SHOULD BE 1
         BZ    OPENERR                  * ITS NOT SO AN ERROR OCCURRED
*
DOBLDL   EQU   *
```

```
         MVC   BLDLLIST(4),BLDLHEAD      * MOVE IN #ENTRIES/L'ENTRY
         MVC   BLNAME,NAME               * MOVE NAME INTO BLDL LIST
*
         LA    R3,BLDL24A                * @(24-BIT CODE)
         LA    R4,BLDL31A                * @(31-BIT RESUMPTION)
         BSM   R4,R3                     * SWITCH TO AMODE 24
*
BLDL24A  EQU   *
         BLDL  (R11),BLDLLIST            * LOOK IN DDNAME FOR MODULE
*
         BSM   Ø,R4                      * SWITCH BACK TO AMODE 31
*
BLDL31A  EQU   *
         LR    R2,R15                    * SAVE BLDL RETURN CODE
*
         MVC   CLOSEW(LCLOSEL),CLOSEL           * CLOSE ...
         CLOSE ((R11)),MODE=31,MF=(E,CLOSEW)    * ... DDNAME
*
         LTR   R2,R2                     * DID WE FIND THE MODULE ?
         BNZ   BLDLERR                   * NO, SO QUIT
         MVI   PRIVLIB,C'Y'              * YES, SET PRIVLIB FLAG
*
         DROP  R11                       * FINISHED WITH DCB
*
NODDN    EQU   *
         L     R11,CVTPTR                * CVT ...
         USING CVTMAP,R11                * ... ADDRESSABILITY
*
         CLI   PRIVLIB,C'Y'              * JUMP IF MODULE ...
         BE    LSPMOD                    * ... IN PRIVATE LIBRARY
         EJECT
*-------------------------------------
* 1) LOOK IN THE ACTIVE LPA QUEUE FOR THE MODULE
*-------------------------------------
* SCAN THE ACTIVE LPA QUEUE FOR THE MODULE
*
         L     R1Ø,CVTQLPAQ              * @(ACTIVE LPA QUEUE)
         L     R1Ø,Ø(R1Ø)                * @(FIRST ALPAQ CDE)
         USING CDENTRY,R1Ø               * CDE ADDRESSABILITY
LPAQLOOP EQU   *
         LTR   R1Ø,R1Ø                   * JUMP OUT OF LOOP ...
         BZ    LPADIR                    * ... AT END OF QUEUE
*
         CLC   NAME,CDNAME               * DO NAMES MATCH ?
         BE    LPAQCDE                   * OH, GOODY !
         L     R1Ø,CDCHAIN               * LOOP BACK ...
         B     LPAQLOOP                  * ... FOR NEXT ALPAQ CDE
*
* CDE FOUND. IF A MINOR CDE (ALIAS) GET THE MAJOR CDE
*
LPAQCDE  EQU   *
```

```
         XC    RETCODE,RETCODE           * INDICATE MODULE FOUND
         TM    CDATTR,CDMIN              * JUMP IF CDE ...
         BNO   LPAQBMSG                  * ... IS A MAJOR CDE
*
         L     R1Ø,CDXLMJP               * OTHERWISE GET @(MAJOR CDE)
         MVC   TRUENM,CDNAME             * SAVE MAJOR NAME
*
* BUILD AND ISSUE THE MESSAGE
*
LPAQBMSG EQU   *
         L     R9,CDXLMJP                * EXTENT LIST ...
         USING XTLST,R9                  * ... ADDRESSABILITY
*
         MVC   MSG(LMSG1A),MSG1          * MOVE IN MESSAGE SKELETON
         MVC   MSG+MSG1NAME(8),NAME      * MOVE IN MODULE NAME
         MVC   MSG+MSG1MLOC(19),LPAQ     * MOVE IN 'ON ALPA QUEUE'
         MVO   TMP8+4(4),XTLMSBLN        * COPY LENGTH AS PACKED
         UNPK  MSG+MSG1LEN(6),TMP8+4(4)  * UNPACK MODULE LENGTH
         NC    MSG+MSG1LEN(6),ZONEMASK   * CONVERT ZONES TO ZEROS
         TR    MSG+MSG1LEN(6),HEXTAB     * CONVERT TO EBCDIC
         MVO   TMP8+3(5),XTLMSBAD        * COPY ADDRESS AS PACKED
         UNPK  MSG+MSG1ADDR(8),TMP8+3(5) * UNPACK MODULE ADDRESS
         NC    MSG+MSG1ADDR(8),ZONEMASK  * CONVERT ZONES TO ZEROS
         TR    MSG+MSG1ADDR(8),HEXTAB    * CONVERT TO EBCDIC
*
         CLI   TRUENM,C' '               * IS THIS AN ALIAS ?
         BNE   LPAQM1A                   * JUMP IF YES
*
         TPUT  MSG,LMSG1                 * MESSAGE W/O TRUE NAME
         B     LPAQMSG2                  * JUMP
*
LPAQM1A  EQU   *
         MVC   MSG+MSG1TNAM(8),TRUENM    * MOVE IN TRUE NAME
         TPUT  MSG,LMSG1A                * MESSAGE WITH TRUE NAME
*
LPAQMSG2 EQU   *
         MVC   MSG(LMSG2),MSG2           * MOVE IN MESSAGE SKELETON
         MVO   TMP8+3(5),CDENTPT         * COPY EPA AS PACKED
         UNPK  MSG+MSG2EPA(8),TMP8+3(5)  * UNPACK EPA
         NC    MSG+MSG2EPA(8),ZONEMASK   * CONVERT ZONES TO ZEROS
         TR    MSG+MSG2EPA(8),HEXTAB     * CONVERT TO EBCDIC
         TM    CDENTPT,CDEMODE           * AMODE 31 ?
         BO    *+1Ø                      * YES, LEAVE MSG AS IS
         MVC   MSG+MSG2AMOD(2),AMODE24   * NO, CHANGE TO AMODE 24
         TPUT  MSG,LMSG2                 * ISSUE MESSAGE
*
         DROP  R9,R1Ø                    * FINISHED WITH CDE, XTLST
         EJECT
*-------------------------------------
* 2) LOOK IN THE LPA DIRECTORY FOR THE MODULE
*-------------------------------------
```

```
*
* CALL THE LPA DIRECTORY SCAN ROUTINE (IEAVVMSR). THIS REQUIRES THE
* MODULE NAME IN RØ/R1, @(CVT) IN R3, AND CHANGES R6, R8, AND R9.
*
LPADIR   EQU   *
         MVC   TRUENM,BLANKS             * CLEAR TRUE NAME
         LM    RØ,R1,NAME                * RØ/R1 = MODULE NAME
         LR    R3,R11                    * R3    = @(CVT)
*
LPADSCAN EQU   *
         L     R15,CVTLPDSR              * @(LPA SEARCH ROUTINE)
         BASR  R14,R15                   * CALL IEAVVMSR
         B     LPADLPDE                  * NORMAL RETURN : FOUND IT
         B     SVCTAB                    * ERROR RETURN  : NOT IN LPA
*
* LPDE FOUND. IF A MINOR LPDE (ALIAS) TRY AGAIN FOR THE MAJOR LPDE
*
LPADLPDE EQU   *
         LR    R1Ø,RØ                    * LPDE ...
         USING LPDE,R1Ø                  * ... ADDRESSABILITY
*
         TM    LPDEATTR,LPDEMIN          * IS THIS A MINOR LPDE ?
         BNO   LPADBMSG                  * NO, ITS A MAJOR
         MVC   TRUENM,LPDEMJNM           * SAVE THE MAJOR NAME
         LM    RØ,R1,LPDEMJNM            * AND GO AND ...
         B     LPADSCAN                  * ... GET ITS CDE
*
* BUILD AND ISSUE THE MESSAGE
*
LPADBMSG EQU   *
         XC    RETCODE,RETCODE           * INDICATE MODULE FOUND
*
         MVC   MSG(LMSG1A),MSG1          * MOVE IN MESSAGE SKELETON
         MVC   MSG+MSG1NAME(8),NAME      * MOVE IN MODULE NAME
         MVC   MSG+MSG1MLOC(16),LPAD     * MOVE IN 'IN ALPA DIRECTORY'
         MVO   TMP8+4(4),LPDEXTLN+1(3)   * COPY LENGTH AS PACKED
         UNPK  MSG+MSG1LEN(6),TMP8+4(4)  * UNPACK MODULE LENGTH
         NC    MSG+MSG1LEN(6),ZONEMASK   * CONVERT ZONES TO ZEROS
         TR    MSG+MSG1LEN(6),HEXTAB     * CONVERT TO EBCDIC
         MVO   TMP8+3(5),LPDEXTAD        * COPY ADDRESS AS PACKED
         UNPK  MSG+MSG1ADDR(8),TMP8+3(5) * UNPACK MODULE ADDRESS
         NC    MSG+MSG1ADDR(8),ZONEMASK  * CONVERT ZONES TO ZEROS
         TR    MSG+MSG1ADDR(8),HEXTAB    * CONVERT TO EBCDIC
*
         CLI   TRUENM,C' '               * SPECIFIED MEMBER AN ALIAS ?
         BNE   LPADM1A                   * YES - USE FULL MESSAGE
*
         TPUT  MSG,LMSG1                 * MESSAGE W/O TRUE NAME
         B     LPADMSG2                  * JUMP
*
LPADM1A  EQU   *
```

```
         MVC   MSG+MSG1TNAM(8),TRUENM    * MOVE IN TRUE NAME
         TPUT  MSG,LMSG1A                * MESSAGE WITH TRUE NAME
*
LPADMSG2 EQU   *
         MVC   MSG(LMSG2),MSG2           * MOVE IN MESSAGE SKELETON
         MVO   TMP8+3(5),LPDENTP         * COPY EPA AS PACKED
         UNPK  MSG+MSG2EPA(8),TMP8+3(5)  * UNPACK EPA
         NC    MSG+MSG2EPA(8),ZONEMASK   * CONVERT ZONES TO ZEROS
         TR    MSG+MSG2EPA(8),HEXTAB     * CONVERT TO EBCDIC
         TM    LPDENTP,LPDEMODE          * AMODE 31 ?
         BO    *+1Ø                      * YES, LEAVE MSG AS IS
         MVC   MSG+MSG2AMOD(2),AMODE24   * NO, CHANGE TO AMODE 24
         TPUT  MSG,LMSG2                 * ISSUE MESSAGE
*
         DROP  R1Ø                       * FINISHED WITH LPDE
         EJECT
*-------------------------------------
* 3) LOOK IN SVC TABLE IF THIS LOOKS LIKE AN SVC
*-------------------------------------
* IF THE MODULE NAME STARTS 'IGC' IT COULD BE AN SVC ...
*
SVCTAB   EQU   *
         CLC   NAME(3),IGCØØ             * COULD IT BE AN SVC?
         BNE   LNKSTPLB                  * NO, JUMP
*
* NUCLEUS SVCS ARE IGCNNN, LPA SVCS ARE IGCØØNNN ...
*
         L     R2,LNAME                  * R2 = L'(MEMBER NAME)
         CH    R2,H6                     * 6 CHARACTERS EXACTLY ?
         BNE   SVCLPA                    * IF NOT, CANNOT BE IN NUCLEUS
         MVI   NUCSVC,X'FF'              * OTHERWISE FLAG AS NUCLEUS
         LA    R1Ø,NAME+3                * POINT TO SVC NAME SUFFIX
         B     SVCCHECK                  * AND GO FOR THE SVC TABLE
*
* NAME > 6 CHARS : IF IT IS AN SVC IT MUST BE AN LPA SVC
*
SVCLPA   EQU   *
         LA    R1Ø,NAME+5                * @(SUFFIX OF SUPPLIED NAME)
         CLC   NAME(5),IGCØØ             * STILL THINK ITS AN SVC ?
         BE    SVCCHECK                  * YES ...
         LA    R1Ø,TRUENM+5              * @(SUFFIX OF TRUE NAME)
         CLC   TRUENM(5),IGCØØ           * STILL THINK ITS AN SVC ?
         BNE   LNKSTPLB                  * NO, SO GO FOR LINKLIST ETC
*
* CHECK SVC 'SUFFIX' IS VALID - MUST BETWEEN Ø AND 255 PACKED DECIMAL
*
SVCCHECK EQU   *
         CLI   Ø(R1Ø),C'Ø'               * X MUST BE ...
         BL    LNKSTPLB                  * ... BETWEEN ...
         CLI   Ø(R1Ø),C'2'               *     ... 'Ø' AND '2'
         BH    LNKSTPLB                  *         ... FOR AN SVC
```

```
         CLI   1(R1Ø),C'Ø'              * Y MUST BE ...
         BL    LNKSTPLB                 * ... BETWEEN ...
         CLI   1(R1Ø),C'9'              *     ... 'Ø' AND '9'
         BH    LNKSTPLB                 *         ... FOR AN SVC
*
         CLI   NUCSVC,X'FF'             * IS IT A NUCLEUS SVC?
         BE    SVCNUCHK                 * YES - GO CHECK LAST BYTE
*
         CLI   2(R1Ø),C'Ø'              * Z MUST BE 'Ø' ...
         BE    SVCOK                    * ... OR ...
         CLI   2(R1Ø),C'{'              *     ... BETWEEN ...
         BL    LNKSTPLB                 *         ... '{' AND ...
         CLI   2(R1Ø),C'I'              *             ... 'I' IF ...
         BH    LNKSTPLB                 *                 ... LPA SVC
         B     SVCOK                    * IT IS AN SVC
*
SVCNUCHK EQU   *
         CLI   2(R1Ø),C'Ø'              * Z MUST BE ...
         BL    LNKSTPLB                 * ... BETWEEN ...
         CLI   2(R1Ø),C'9'              *     ... 'Ø' AND '9' ...
         BH    LNKSTPLB                 *         ... FOR NUCLEUS SVC
*
* SVC SUFFIX IS VALID - LOCATE AND LIST CONTENTS OF ITS SVC TABLE ENTRY
*
SVCOK    EQU   *
         PACK  TMP8,Ø(3,R1Ø)            * PACK THE SUFFIX
         CVB   R2,TMP8                  * CONVERT TO BINARY
         CH    R2,H255                  * IF > 255 MODULE ...
         BH    LNKSTPLB                 * ... CAN'T BE AN SVC
*
         L     R1Ø,CVTABEND             * SCVT ...
         USING SCVTSECT,R1Ø             * ... ADDRESSABILITY
*
         LR    R1,R2                    * OFFSET INTO SVC TABLE ...
         SLL   R1,3                     * ... = (SVC NUMBER) * 8
         L     R1Ø,SCVTSVCT             * GET @(SVC TABLE)
         AR    R1Ø,R1                   * SVC TABLE ENTRY ...
         USING SVCENTRY,R1Ø             * ... ADDRESSABILITY
*
         MVC   MSG(LMSG3),MSG3          * MOVE IN MESSAGE SKELETON
         MVC   MSG+MSG3NAME(8),NAME     * MOVE IN MODULE NAME
         CVD   R2,TMP8                  * ... TO DECIMAL
         ED    MSG+MSG3SVCN(4),TMP8+6   * AND EDIT INTO MESSAGE
         TM    SVCATTR1,SVCTP34         * IS IT A TYPE 3 (OR 4)
         BNO   NOTTYP34                 * NOPE
         MVI   MSG+MSG3TYPE,C'3'        * MOVE IN TYPE '3' (OR 4)
         B     TYPEDONE                 * AND JUMP
NOTTYP34 EQU   *
         TM    SVCATTR1,SVCTP2          * IS IT A TYPE 2 ?
         BNO   NOTTYP2                  * NOPE
         MVI   MSG+MSG3TYPE,C'2'        * MOVE IN TYPE '2'
```

```
         B     TYPEDONE                  * AND JUMP
NOTTYP2  EQU   *
         TM    SVCATTR1,SVCTP6           * IS IT A TYPE 6
         BNO   TYPEDONE                  * IF NOT, MUST BE TYPE 1
         MVI   MSG+MSG3TYPE,C'6'         * MOVE IN TYPE '6'
TYPEDONE EQU   *
         MVO   TMP8+3(5),SVCEP           * COPY EP ADDR AS PACKED
         UNPK  MSG+MSG3ADDR(8),TMP8+3(5) * UNPACK EP ADDRESS
         NC    MSG+MSG3ADDR(8),ZONEMASK  * CONVERT ZONES TO ZEROS
         TR    MSG+MSG3ADDR(8),HEXTAB    * CONVERT TO EBCDIC
         TPUT  MSG,LMSG3                 * ISSUE MESSAGE
*
         CLI   TRUENM,C' '               * SPECIFIED MEMBER AN ALIAS ?
         BE    LNKSTPLB                  * NO THATS IT HERE
*
         MVC   MSG(LMSG6),MSG6           * MOVE IN MESSAGE SKELETON
         MVC   MSG+MSG6NAME(8),NAME      * MOVE IN MODULE NAME
         MVC   MSG+MSG6TRNM(8),TRUENM    * MOVE IN TRUE NAME
         TPUT  MSG,LMSG6                 * ISSUE MESSAGE
*
         DROP  R1Ø                       * FINISHED WITH SVC TABLE
         EJECT
*-------------------------------------
* 4) LOOK IN LINKLIST/STEPLIB
*-------------------------------------
* DO A LINKLIST/STEPLIB BLDL (NO DCB)
*
LNKSTPLB EQU   *
         MVC   BLDLLIST(4),BLDLHEAD      * MOVE IN #ENTRIES/L'ENTRY
         MVC   BLNAME,NAME               * MOVE NAME INTO BLDL LIST
         LA    R3,BLDL24B                * @(24-BIT CODE)
         LA    R4,BLDL31B                * @(31-BIT RESUMPTION)
         BSM   R4,R3                     * SWITCH TO AMODE 24
*
BLDL24B  EQU   *
         BLDL  Ø,BLDLLIST                * LOOK IN LINKLIST / STEPLIB
*
         BSM   Ø,R4                      * SWITCH BACK TO AMODE 31
*
BLDL31B  EQU   *
         LTR   R15,R15                   * FOUND ANYTHING ?
         BNZ   EXIT                      * JUMP IF NOT
*
* MODULE FOUND IN LINKLIST/STEPLIB, OR SPECIFIED PRIVATE LIBRARY
*
LSPMOD   EQU   *
         XC    RETCODE,RETCODE           * INDICATE MODULE FOUND
         MVC   MSG(LMSG4T),MSG4          * MOVE IN MESSAGE SKELETON
         MVC   MSG+MSG4NAME(8),NAME      * MOVE IN MODULE NAME
         TM    BLC,BLCALIAS              * IS THIS AN ALIAS?
         BNO   LSPTYPOK                  * NO, SO JUMP
```

```
         MVC   MSG+MSG4MTYP(5),MINOR     * YES, SO FLAG IT
*
LSPTYPOK EQU   *
         CLI   BLZ,BLZPRIV               * MODULE IN PRIVATE LIBRARY ?
         BE    LSPLIBOK                  * YES, ALL SET
         MVC   MSG+MSG4LTYP(16),JSLIB    * YES, SO FLAG IT
         CLI   BLZ,BLZLINK               * MODULE IN LINKLIST LIBRARY ?
         BH    LSPLIBOK                  * NO, ASSUMPTION WAS CORRECT
*
         CLI   BLK,X'ØØ'                 * IN SYS1.LINKLIB ITSELF ?
         BNE   LSPLIBOK                  * NOT IF CONCAT # > Ø
         MVC   MSG+MSG4LTYP(21),LINKLIB  * YES, SO FLAG IT
         B     LSPMSG                    * ALL DONE
*
LSPLIBOK EQU   *
         SR    R8,R8                     * GET ...
         IC    R8,BLK                    * ... CONCATENATION NUMBER
         LA    R1,1(R8)                  * ADD 1 (FIRST D/S IS CCAT Ø)
         CVD   R1,TMP8                   * CONVERT TO DECIMAL
         MVC   TMP8(4),I3PAT             * CONVERT CONCAT # ...
         ED    TMP8(4),TMP8+6            * ... TO EBCDIC ...
         MVC   MSG+MSG4CCAT(2),TMP8+2    *      ... AND MOVE INTO MSG
*
         CLI   BLZ,BLZLINK               * IS IT IN LINKLIST?
         BNE   LSPSTPLB                  * NO, MUST BE IN STEP/PRIV LIB
*
* MODULE IN LINKLIST - SCAN LINKLIST TABLE FOR ACTUAL DATASET
*
         MVC   MSG+MSG4LTYP(16),LLIB     * MOVE IN 'LINKLIST'
         L     R1Ø,CVTLLTA               * @(LINKLIST TABLE)
         LA    R1Ø,8(R1Ø)                * SCROLL PAST HEADER
LSPLLTLP EQU   *
         LA    R1Ø,45(R1Ø)               * MOVE DOWN TABLE ...
         BCT   R8,LSPLLTLP               * ... TO THE ENTRY WE WANT
*
         MVC   MSG+MSG4DSNM(44),1(R1Ø)   * MOVE IN LINKLIST D/S NAME
*
         MVC   CAMLSTW(LCAMLST),CAMLST   * SKELETON CAMLST TO WORKAREA
         LA    R1,MSG+MSG4DSNM           * STORE @(DSN) ...
         ST    R1,CAMLSTW+4              * ... IN CAMLST PARMLIST
         LA    R1,CAMDATA                * STORE @(CAMLST WORKAREA) ...
         ST    R1,CAMLSTW+12             * ... IN CAMLST PARMLIST
*
         LOCATE CAMLSTW                  * EXECUTE LOCATE MACRO
*
         LA    R1,MSG+MSG4DSNM+43        * @(LAST CHAR OF DATASET NAME
         LA    R2,44                     * #(CHARS IN DATASET NAME)
LSPDSLP1 EQU   *
         CLI   Ø(R1),C' '                * FOUND END OF DATASET NAME ?
         BNE   LSPDSND1                  * YES, SO JUMP OUT OF LOOP
         BCTR  R1,Ø                      * NO, SO MOVE BACK ...
```

```
         BCT   R2,LSPDSLP1              * ... AND LOOK AGAIN
         LA    R1,1(R1)                 * DSN ALL BLANKS
LSPDSND1 EQU   *
         MVC   2(2,R1),ON               * MOVE IN 'ON'
         MVC   5(6,R1),VSERIAL          * MOVE IN DATASET VOLSER
         B     LSPMSG                   * JUMP
*
* MODULE IN STEPLIB OR PRIVATE LIBRARY- SCAN TIOT FOR DDNAME
*
LSPSTPLB EQU   *
         L     R1Ø,CVTTCBP              * CVTTCP ...
         L     R1Ø,4(R1Ø)               * ... -> ACTIVE TCB ...
         L     R9,TCBTIO-TCB(R1Ø)       *     ... -> TIOT
         LA    R9,24(R9)                * POINT AT FIRST DD ENTRY
LSPTIOLP EQU   *
         CLI   Ø(R9),X'ØØ'              * JUMP OUT ...
         BE    LSPMSG                   * ... IF HIT THE END
         CLC   4(8,R9),DDNAME           * IS THIS THE ENTRY ?
         BE    LSPGOTDD                 * YES ...
         SR    R1,R1                    * GET LENGTH ...
         IC    R1,Ø(R9)                 * ... OF TIOT ENTRY
         AR    R9,R1                    * LOOP BACK ...
         B     LSPTIOLP                 * ... FOR NEXT ENTRY
*
LSPGOTDD EQU   *
         CLI   BLK,X'ØØ'                * FIRST STEPLIB ?
         BE    LSPGOTDS                 * YES - GOT IT IN ONE
LSPDSNLP EQU   *
         SR    R1,R1                    * GET LENGTH ...
         IC    R1,Ø(R9)                 * ... OF TIOT ENTRY
         AR    R9,R1                    * ADVANCE TO NEXT ENTRY
         CLI   Ø(R9),X'ØØ'              * HIT THE END ?
         BE    LSPMSG                   * YES, SO QUIT
         CLC   4(8,R9),BLANKS           * PART OF SAME CONCATENATION ?
         BNE   LSPMSG                   * NO, SO QUIT
         BCT   R8,LSPDSNLP              * OTHERWISE KEEP LOOPING
*
LSPGOTDS EQU   *
         SR    R8,R8                    * GET ...
         ICM   R8,B'Ø111',12(R9)        * ... JFCB ...
         LA    R8,16(R8)                *     ... ADDRESS
         USING INFMJFCB,R8              * JFCB ADDRESSABILITY
         MVC   MSG+MSG4DSNM(44),JFCBDSNM * MOVE IN DATASET NAME
         LA    R1,MSG+MSG4DSNM+43       * @(LAST CHAR OF DATASET NAME
         LA    R2,44                    * #(CHARS IN DATASET NAME)
LSPDSLP2 EQU   *
         CLI   Ø(R1),C' '               * FOUND END OF DATASET NAME ?
         BNE   LSPDSND2                 * YES, SO JUMP OUT OF LOOP
         BCTR  R1,Ø                     * NO, SO MOVE BACK ...
         BCT   R2,LSPDSLP2              * ... AND LOOK AGAIN
         LA    R1,1(R1)                 * DSN ALL BLANKS
```

```
LSPDSND2 EQU   *
         MVC   2(2,R1),ON                * MOVE IN 'ON'
         MVC   5(6,R1),JFCBVOLS          * MOVE IN DATASET VOLSER
         DROP  R8                        * FINISHED WITH JFCB
*
* AT LAST - ISSUE MESSAGES
*
LSPMSG   EQU   *
         MVO   TMP8+4(4),BLTTR           * COPY TTR AS PACKED
         UNPK  MSG+MSG4TTR(6),TMP8+4(4)  * UNPACK TTR
         NC    MSG+MSG4TTR(6),ZONEMASK   * CONVERT ZONES TO ZEROS
         TR    MSG+MSG4TTR(6),HEXTAB     * CONVERT TO EBCDIC
         TPUT  MSG,LMSG4                 * ISSUE MESSAGE
*
         CLI   MSG+MSG4DSNM,C' '         * DSN PRESENT ?
         BE    LSPMSG5                   * NO, SO SKIP SECOND LINE
         TPUT  MSG+LMSG4,LMSG4A          * ISSUE MESSAGE
*
LSPMSG5  EQU   *
         MVC   MSG(LMSG5),MSG5           * MOVE IN MESSAGE SKELETON
         MVO   TMP8+4(4),BLMODLN         * COPY LENGTH AS PACKED
         UNPK  MSG+MSG5LEN(6),TMP8+4(4)  * UNPACK LENGTH
         NC    MSG+MSG5LEN(6),ZONEMASK   * CONVERT ZONES TO ZEROS
         TR    MSG+MSG5LEN(6),HEXTAB     * CONVERT TO EBCDIC
         MVO   TMP8+4(4),BLEPADDR        * COPY EPA AS PACKED
         UNPK  MSG+MSG5EPA(6),TMP8+4(4)  * UNPACK EPA
         NC    MSG+MSG5EPA(6),ZONEMASK   * CONVERT ZONES TO ZEROS
         TR    MSG+MSG5EPA(6),HEXTAB     * CONVERT TO EBCDIC
         TM    BLARMODE,BLAM31           * AMODE 31 ?
         BO    *+1Ø                      * YES, LEAVE MSG AS IS
         MVC   MSG+MSG5AMOD(2),AMODE24   * NO, CHANGE TO AMODE 24
         TM    BLARMODE,BLRMANY          * RMODE ANY ?
         BO    *+1Ø                      * YES, LEAVE MSG AS IS
         MVC   MSG+MSG5RMOD(3),RMODE24   * NO, CHANGE TO RMODE 24
         TPUT  MSG,LMSG5                 * ISSUE MESSAGE
*
         TM    BLC,BLCALIAS              * IS THIS AN ALIAS?
         BZ    EXIT                      * NO, SO QUIT
         TM    BLC,X'ØF'                 * 15 HALFWORDS
         BO    EXIT                      * YES - SKIP ALIAS MESSAGE
*
         MVC   MSG(LMSG6A),MSG6          * MOVE IN MESSAGE SKELETON
         MVC   MSG+MSG6NAME(8),NAME      * MOVE IN MODULE NAME
         MVC   MSG+MSG6TRNM(8),BLMEMNAM  * MOVE IN TRUE NAME
         MVO   TMP8+4(4),BLMEMEPA        * COPY EPA AS PACKED
         UNPK  MSG+MSG6EPA(6),TMP8+4(4)  * UNPACK TRUE NAME EPA
         NC    MSG+MSG6EPA(6),ZONEMASK   * CONVERT ZONES TO ZEROS
         TR    MSG+MSG6EPA(6),HEXTAB     * CONVERT TO EBCDIC
         TPUT  MSG,LMSG6A                * ISSUE MESSAGE
         EJECT
*********************************************************************
* ALL DONE ...
```

```
*********************************************************************
EXIT     EQU   *
         OC    RETCODE,RETCODE           * DID WE FIND ANYTHING?
         BZ    RETURN                    * YEP.... ALL DONE
*
         MVC   MSG(LMSGØ),MSGØ           * MOVE MESSAGE BELOW LINE
         MVC   MSG+MSGØNAME(8),NAME      * MOVE IN MODULE NAME ...
         TPUT  MSG,LMSGØ                 * ... AND INFORM USER
*
RETURN   EQU   *
         IKJRLSA APDL                    * GET RID OF THE PDL
*
         LR    R1,R13                    * R1  = OUR SAVEAREA ADDRESS
         L     R13,4(R13)                * R13 = HSA ADDRESS
         LA    RØ,LWKAREA                * WORKAREA LENGTH
         FREEMAIN RU,LV=(RØ),A=(R1),SP=78
*
         L     R14,12(R13)               * RESTORE R14
         SR    R15,R15                   * NEVER PREVENT LOGON
         LM    RØ,R12,2Ø(R13)            * RESTORE RØ-R12
         BR    R14                       * AND RETURN
         EJECT
*
*********************************************************************
* ERROR CONDITIONS
*********************************************************************
*
OPENERR  EQU   *
         MVC   MSG(OPENMSGL),OPENMSG     * MOVE MESSAGE BELOW LINE
         MVC   MSG+MSGODDNM(8),DDNAME    * MOVE IN DDNAME
         TPUT  MSG,OPENMSGL              * TELL USER OPEN FAILED
         B     RETURN
*
BLDLERR  EQU   *
         MVC   MSG(BLDLMSGL),BLDLMSG     * MOVE MESSAGE BELOW LINE
         MVC   MSG+MSGBNAME(8),BLNAME    * MOVE IN NAME
         TPUT  MSG,BLDLMSGL              * TELL USER BLDL FAILED
         B     RETURN
         EJECT
*********************************************************************
* CONSTANTS, VARIABLES AND DATA AREAS
*********************************************************************
MOVENAME MVC   NAME(Ø),Ø(R1)
MOVEDDN  MVC   DDNAME(Ø),Ø(R1)
*
PARSADDR DC    V(PARMTAB)
H6       DC    H'6'
H255     DC    H'255'
BLDLHEAD DC    H'1'
         DC    H'76'
*
I3PAT    DC    XL4'FØ2Ø212Ø'
```

```
BLANKS   DC    CL8'        '
STEPLIB  DC    CL8'STEPLIB '
IGCØØ    DC    CL5'IGCØØ'
MINOR    DC    CL5'minor'
RMODE24  DC    CL3'24 '
AMODE24  EQU   RMODE24,2
LINKLIB  DC    CL21'SYS1.LINKLIB         '
LLIB     DC    CL16'LINKLIST library'
JSLIB    DC    CL16'JOB/STEP library'
LPAQ     DC    CL19'on active LPA Queue'
ON       EQU   LPAQ,2
LPAD     DC    CL16'in LPA Directory'
*
ZONEMASK DC    XL8'ØFØFØFØFØFØFØFØF'
HEXTAB   DC    CL16'Ø123456789ABCDEF'
*
DCB      DCB   DDNAME=DUMMY,MACRF=R,DSORG=PO
LDCB     EQU   *-DCB
*
OPENL    OPEN  (*-*),MODE=31,MF=L        * MF=L OPEN MACRO
LOPENL   EQU   *-OPENL
*
CLOSEL   CLOSE (*-*),MODE=31,MF=L        * MF=L CLOSE MACRO
LCLOSEL  EQU   *-CLOSEL
*
CAMLST   CAMLST NAME,CAMLST,,CAMLST      * SKELETON CAMLST MACRO
LCAMLST  EQU   *-CAMLST
         EJECT
*-------------------------------------
* MESSAGE SKELETONS
*-------------------------------------
MSGØ     DS    ØF
         DC    C'XXXXXXXX not found in LPA/LINKLIST/Private library'
MSGØNAME EQU   Ø,8
LMSGØ    EQU   *-MSGØ
*
MSG1     DS    ØF
         DC    C'XXXXXXXX is XXXXXX bytes at XXXXXXXX                 +
                '
MSG1NAME EQU   Ø,8
MSG1LEN  EQU   12,6
MSG1ADDR EQU   28,8
MSG1MLOC EQU   37,16
LMSG1    EQU   *-MSG1
MSG1A    DC    C' (Alias of XXXXXXXX)'
MSG1TNAM EQU   LMSG1+11,8
LMSG1A   EQU   *-MSG1
*
MSG2     DS    ØF
         DC    C'         Relocated entry point at XXXXXXXX is AMODE 31+
               '
```

```
MSG2NAME EQU   Ø,8
MSG2EPA  EQU   34,8
MSG2AMOD EQU   52,2
LMSG2    EQU   *-MSG2
*
MSG3     DS    ØF
         DC    C'XXXXXXXX is SVC ??? (Type 1), entry point at XXXXXXXX'
MSG3NAME EQU   Ø,8
MSG3SVCN EQU   15,4
MSG3TYPE EQU   26,1
MSG3ADDR EQU   45,8
LMSG3    EQU   *-MSG3
*
MSG4     DS    ØF
         DC    C'XXXXXXXX found as a major name at TTR XXXXXX in privat+
               e library  (XX)'
MSG4NAME EQU   Ø,8
MSG4MTYP EQU   2Ø,5
MSG4TTR  EQU   38,6
MSG4LTYP EQU   48,12
MSG4CCAT EQU   66,2
LMSG4    EQU   *-MSG4
MSG4A    DC    C'          DSN =                                        +
                    '
MSG4DSNM EQU   LMSG4+15,44
LMSG4A   EQU   *-MSG4A
LMSG4T   EQU   *-MSG4
*
MSG5     DS    ØF
         DC    C'          Module length is XXXXXX, RMODE ANY; entry poi+
               nt at XXXXXX is AMODE 31'
MSG5NAME EQU   Ø,8
MSG5LEN  EQU   26,6
MSG5RMOD EQU   4Ø,3
MSG5EPA  EQU   6Ø,6
MSG5AMOD EQU   76,2
LMSG5    EQU   *-MSG5
*
MSG6     DS    ØF
         DC    C'XXXXXXXX is an alias of XXXXXXXX'
MSG6NAME EQU   Ø,8
MSG6TRNM EQU   24,8
LMSG6    EQU   *-MSG6
         DC    C', entry point at XXXXXX'
MSG6EPA  EQU   LMSG6+17,6
LMSG6A   EQU   *-MSG6
*
OPENMSG  DS    ØF
         DC    C'Unable to OPEN DDname XXXXXXXX - not allocated ?'
MSGODDNM EQU   22,8
OPENMSGL EQU   *-OPENMSG
```

```
*
BLDLMSG  DS    ØF
         DC    C'BLDL failed for member XXXXXXXX'
MSGBNAME EQU   23,8
BLDLMSGL EQU   *-BLDLMSG
         EJECT
*-------------------------------------
* PARAMETER CONTROL LIST FOR IKJPARS
*-------------------------------------
PARMTAB  IKJPARM DSECT=IKJPARMD
PARMTAB  AMODE 31
PARMTAB  RMODE 24
MEMPDE   IKJIDENT 'MEMBER NAME',MAXLNTH=8,FIRST=ALPHA,                 +
               OTHER=ALPHANUM,PROMPT='NAME OF MEMBER TO LOCATE'
DDNKWD   IKJKEYWD
         IKJNAME  'DDNAME',SUBFLD=DDSUBF,ALIAS=('IN')
DDSUBF   IKJSUBF
DDNPDE   IKJIDENT 'DDNAME',MAXLNTH=8,FIRST=ALPHA,OTHER=ALPHANUM,       +
               PROMPT='DDNAME OF CONCATENATION TO BE SCANNED'
         IKJENDP
*
* PARSE DESCRIPTOR ELEMENT FOR 'MEMBER' AND DDNAME PARAMETERS
*
PDE      DSECT
PDEOPADR DS    A                        * @(OPERAND)
PDEOPLEN DS    H                        * L'OPERAND
PDEFLAGS DS    BL1                      * FLAG BYTE
PDEFPRES EQU   X'8Ø'                    *  OPERAND PRESENT FLAG
         DS    XL1                      * RESERVED
         EJECT
*-------------------------------------
* WORKAREA DSECT
*-------------------------------------
WORKAREA DSECT
*
SAVEAREA DS    18F
*
TMP8     DS    D
*
STPPL    DS    8F                       * IKJPARS PARAMETER LIST
CPECB    DS    F                        * ECB
APDL     DS    A                        * @(PDL) SET BY IKJPARS
*
         DS    ØF
DCBW     DS    CL(LDCB)                 * BLDL DCB
*
         DS    ØF
OPENW    DS    CL(LOPENL)               * MF=L OPEN MACRO
*
         DS    ØF
CLOSEW   DS    CL(LCLOSEL)              * MF=L CLOSE MACRO
*
```

```
         DS    ØF
BLDLLIST DS    H                       * NUMBER OF ENTRIES
         DS    H                       * ENTRY LENGTH
BLNAME   DS    CL8                     * MEMBER NAME OR ALIAS
BLTTR    DS    XL3                     * START TTR OF NAMED MEMBER
BLK      DS    XL1                     * CONCATENATION NUMBER
BLZ      DS    XL1                     * LIBRARY FLAG
BLZPRIV  EQU   X'ØØ'                   *  PRIVATE LIBRARY
BLZLINK  EQU   X'Ø1'                   *  LINKLIST LIBRARY
BLZJOB   EQU   X'Ø2'                   *  JOB/STEP/TASK LIBRARY
BLC      DS    XL1                     * INDICATOR BYTE
BLCALIAS EQU   X'8Ø'                   *  NAME IS AN ALIAS IF SET
*
BLTTRTX1 DS    XL3                     * TTR OF FIRST TEXT RECORD
BLZ2     DS    XL1                     * ZEROS
BLTTRNL  DS    XL3                     * TTR OF NOTE LIST (IF ANY)
BLNNOTEL DS    XL1                     * NUMBER OF NOTE LIST ENTRIES
BLMATTR1 DS    XL2                     * MODULE ATTRIBUTES (1)
BLA1RENT EQU   X'8Ø'                   *  REENTRANT
BLA1REUS EQU   X'4Ø'                   *  REUSABLE
BLA1EXEC EQU   X'Ø2'                   *  EXECUTABLE
BLA1REFR EQU   X'Ø1'                   *  REFRESHABLE
BLMODLN  DS    XL3                     * VIRTUAL STORAGE REQUIRED
BLLTX1   DS    XL2                     * LENGTH OF FIRST TEXT RECORD
BLEPADDR DS    XL3                     * MODULE ENTRY POINT
BLMATTR2 DS    XL1                     * MODULE ATTRIBUTES (2)
BLARMODE DS    XL1                     * AMODE/RMODE FLAGS
BLRMANY  EQU   X'1Ø'                   *  RMODE = ANY
BLAM31   EQU   X'Ø2'                   *  AMODE = 31
BLRLDCNT DS    XL1                     * RLD COUNT
BLBSEND  EQU   *
*
         ORG   BLBSEND
BLLSCLST DS    XL2                     * LENGTH OF SCATTER LIST
BLLTRTAB DS    XL2                     * LENGTH OF TRANSLATION TABLE
BLLESDTX DS    XL2                     * ESDID FOR FIRST TEXT RECORD
BLLESDEP DS    XL2                     * ESDID FOR ENTRY POINTS
*
         ORG   BLBSEND
BLMEMEPA DS    XL3                     * EPA OF 'REAL' MEMBER
BLMEMNAM DS    XL8                     * NAME OF 'REAL' MEMBER
         DS    XL22
*
NAME     DS    CL8
DDNAME   DS    CL8
TRUENM   DS    CL8
RETCODE  DS    F
LNAME    DS    F
PRIVLIB  DS    CL1
NUCSVC   DS    XL1
*
         DS    ØF
```

```
CAMLSTW  DS    CL(LCAMLST)               * CAMLST FOR LOCATE
*
CAMDATA  DS    ØD                        * CAMLST DATA AREA
         DS    265CL1
         ORG   CAMDATA
VCOUNT   DS    H                         * VOLUME COUNT
VDEVT    DS    XL4                       * DEVICE TYPE
VSERIAL  DS    CL6                       * VOLUME SERIAL
DSNSEQ   DS    H                         * DATASET SEQUENCE NUMBER
         ORG
*
         DS    ØF
MSG      DS    CL16Ø
*
WORKEND  EQU   *
LWKAREA  EQU   *-WORKAREA
         EJECT
*
*———————————————————————————————
* SYSTEM DSECTS
*———————————————————————————————
*
         PRINT NOGEN,NODATA
         IHAPSA LIST=NO
         CVT    DSECT=YES
         IHASCVT
         IHASVC
         IKJTCB LIST=NO
         DCBD  DSORG=PS,DEVD=DA          * DCB MAPPING MACRO
         DSECT
         IEFJFCBN                        * JFCB MAPPING MACRO
         EJECT
         PRINT GEN
         IKJCPPL
         IKJPPL
         EJECT
         IHALPDE
         IHACDE
         IHAXTLST
         END
```

Peter Wright
Associate Consultant
Tessella Support Services (UK)

Table conversion from TSO to Word

THE PROBLEM

Have you ever tried to import a table produced in TSO into a Microsoft Word document? One of the problems I have found is the spacing between words in the TSO format. What looks great in a TSO table does not convert very well into Word format.

THE SOLUTION

A method which I have found works is as follows:

- Produce your table in TSO format with a delimiter character already inserted. It doesn't matter about spaces, but ensure that the delimiter character doesn't appear in the table in its own right!
- Then run the edit macro below (JOINR), which will remove all the blanks in each line of the TSO dataset.
- You can then upload this dataset to your PC in standard text format, and then import it into your Word document.
- It is then just a case of 'Table' and 'Convert text to table' and choosing your character in the 'separate text at' box, and your table will look perfectly formatted.

If your TSO dataset does not contain a delimiter character, then you can specify one as a parameter on the JOINR macro (ie JOINR £).

JOINR

```
/* REXX*/
 trace n
 /* To remove blanks in a line.                          */
 "ISREDIT MACRO (vrip)"
 If(vrip = ' ') then Do
   symb = ''
 End /* If(vrip = ' ') then Do */
 Else Do
   symb = vrip
 End /* If(vrip = ' ') then Do */
 iflag1 = 0
```

```
"ISREDIT (VAR2) = LINENUM .ZLAST"
say 'Number of rows to be processed is:' var2
row = Ø
col = 1
Do until row = var2
   row = row + 1
  "ISREDIT CURSOR = (row,col)"
  "ISREDIT (STM) = LINE .ZCSR"
  temp = stm
  xt = Ø
  Do until temp = ''
    xt = xt + 1
    parse var temp valt.xt temp
  End /* Do until temp = '' */
  outl = ''
  /* Build the line back up again.               */
  Do jk= 1 to xt
    If(symb = ' ') then Do
      outl = outl || valt.jk
    End /* If(symb = ' ') then Do */
    Else Do
      outl = outl || valt.jk || symb
    End /* If(symb = ' ') then Do */
  End /* Do jk= 1 to xt */
  /* Delete and write out the line.              */
  "ISREDIT CURSOR = (row,col)"
  val1 = "'ISREDIT LINE_AFTER .ZCSR ="
  val2 = outl || ' " '
  interpret  val1 ' "'val2  " ' "
  "ISREDIT DELETE " row
End /* Do until row = var2 */
```


Copying files from 3380s to 3390s

INTRODUCTION

This article considers how to obtain extra functionality from the ADRDSSU copy facility. At some point in the life of an organization, the processing of copying files from DASD to DASD comes under review. In our case, we have changed DASD from 3380 to 3390. You can use the command:


```
dfhsm hmigrate an hrecall
```

for each file but this job is very easy to use. The job EMPTYDA is used to copy during the day.

EMPTYDA

```
//ADCOP8Ø6 JOB SYS,'PSY',CLASS=Y,MSGCLASS=Ø,
//         NOTIFY=&SYSUID,MSGLEVEL=(1,1)
//*
//*************************************************
//* Ø1 : COPY WITHOUT REBLOCK           * LYO8Ø3 *
//*************************************************
//*
//DFDSSØ1  EXEC PGM=ADRDSSU,REGION=4M,PARM='LINECNT=66,UTILMSG=YES'
//*=========================
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD    *
 COPY DATASET(INCLUDE(SAS.**,CICS%%%%.**,NCP%%%.** -
              *.LOG.* OPCE.**) ) -
      ALLEXCP -
      ALLMULTI -
      CANCELERROR -
      DELETE -
      LOGINDYNAM( LYO8Ø6 ) -
      OUTDYNAM( (LYO915) (LYO917) (LYO918) (LYO919) (LYO91A) ) -
      PERCENTUTILIZED( 8Ø ) -
      PURGE -
      RECATALOG(*) -
      SPHERE -
      TGTALLOC(SOURCE) -
      WAIT(2,2)
/*
//*************************************************
//* Ø1 : COPY WITH REBLOCK              * LYO8Ø3 *
//*************************************************
//DFDSSØ2  EXEC PGM=ADRDSSU,REGION=4M,PARM='LINECNT=66,UTILMSG=YES'
//*=========================
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD    *
 COPY DATASET(INCLUDE(**)) -
      ALLEXCP -
      ALLMULTI -
      CANCELERROR -
      DELETE -
      LOGINDYNAM( LYO8Ø6 ) -
      OUTDYNAM( (LYO915) (LYO917) (LYO918) (LYO919) (LYO91A) ) -
      PERCENTUTILIZED( 8Ø ) -
```

```
      PURGE -
      REBLOCK(**) -
      RECATALOG(*) -
      SPHERE -
      TGTALLOC(SOURCE) -
      WAIT(2,2)
/*
```

Claude Dunand (France)

Improving ISPF productivity

INTRODUCTION

The following edit macros were developed to make ISPF edit and browse processing more productive.

CB, CE, AND CM

The first three macros, CB,CE, and CM, allow browsing, editing, or dataset list utility processing of any dataset name shown on the current edit screen.

This is achieved by simply entering CB or CE or CM on the command line and then locating the cursor anywhere on the dsname, before hitting enter. For example, while editing JCL, the contents of a SYSIN member may be browsed or edited without recourse to the normal process of entering EDIT on the command line and then having to remember and enter the DSN(member) details on the resultant ISREDM03 (Edit Command – Entry) panel.

CUT AND PASTE

The macros CUT and PASTE allow the ISPF user to have access to functions not too dissimilar to the very useful PC functions of CUT, COPY, and PASTE. As the HELP information contained in each macro shows, CUTting will either copy lines selected by CC..CC line

commands or cut lines selected by MM...MM line commands. The resultant block of lines is then PASTEd either B(efore) or A(fter) the designated line in the target dataset.

CUT lines are held as variables in the user ISPF profile pool. Any consecutive uses of CUT without an intervening PASTE will result in the loss of the lines copied or moved from the first CUT.

OPERATIONAL REQUIREMENTS

All macros have been developed and run under ISPF/PDF 4.2.1 for OS/390 1.3 but should be downwardly compatible to at least ISFP 3.2.

All macros can of course be assigned to a spare PF key, something which has proved very useful with CUT and PASTE. (For example, PF16 = ;UT, PF17 = ;PASTE.)

CURSOR-BASED BROWSE MACRO

The cursor-based browse macro is defined in the ISPF ISPEXEC library as CB.

```
 /***REXX*************************************************************/
ADDRESS ISREDIT
'MACRO PROCESS'

ADDRESS ISPEXEC 'CONTROL ERRORS RETURN'

'(cl,cc) = CURSOR'
IF cc = Ø THEN EXIT

'(LINE) = LINE' cl
UPPER line

valid = XRANGE('A','Z') XRANGE('Ø','9') '@#$£(.)'

end = LENGTH(line)
start = end - VERIFY(REVERSE(line) ' ',valid,'N',end-cc+1) + 2
len = VERIFY(line' ',valid,'N',cc) - start

IF len < 1 THEN EXIT

dsn = SUBSTR(line,start,len)

IF VERIFY(dsn,'.','M') = Ø THEN DO
   '(dataset) = DATASET'
```

```
   dsn = dataset'('dsn')'
END

ADDRESS ISPEXEC BROWSE "DATASET('"dsn"')"
IF rc > Ø THEN
   DO
     retcode = rc
     SIGNAL Error_check
   END
ELSE EXIT

Error_check:
IF retcode = 12 THEN
   DO
     ADDRESS ISPEXEC "SETMSG MSG(CMSGSØØ7)"
     EXIT
   END
ELSE
   DO
     IF retcode = 14 THEN
        DO
          ADDRESS ISPEXEC "SETMSG MSG(CMSGSØØ8)"
          EXIT
        END
     ELSE
        IF retcode = 16 THEN
           DO
             ADDRESS ISPEXEC "SETMSG MSG(CMSGSØØ9)"
             EXIT
           END
        ELSE
           DO
             ADDRESS ISPEXEC "SETMSG MSG(CMSGSØØØ)"
           END
   END
EXIT
```

CURSOR-BASED EDIT MACRO (CE)

The cursor-based browse macro is defined in the ISPF ISPEXEC library as CE.

```
/***REXX*************************************************************/
ADDRESS ISREDIT
'MACRO PROCESS'

ADDRESS ISPEXEC 'CONTROL ERRORS RETURN'

'(cl,cc) = CURSOR'
```

```
IF cc = Ø THEN EXIT

'(LINE) = LINE' cl
UPPER line

valid = XRANGE('A','Z') XRANGE('Ø','9') '@#$£(.)'

end = LENGTH(line)
start = end - VERIFY(REVERSE(line) ' ',valid,'N',end-cc+1) + 2
len = VERIFY(line' ',valid,'N',cc) - start

IF len < 1 THEN EXIT

dsn = SUBSTR(line,start,len)

IF VERIFY(dsn,'.','M') = Ø THEN DO
   '(dataset) = DATASET'
   dsn = dataset'('dsn')'
END

ADDRESS ISPEXEC EDIT "DATASET('"dsn"')"
IF rc > Ø THEN
   DO
     retcode = rc
     SIGNAL Error_check
   END
ELSE EXIT

Error_check:
IF retcode = 4 THEN
   DO
     ADDRESS ISPEXEC "SETMSG MSG(CMSGSØØ4)"
     EXIT
   END
ELSE
   DO
     IF retcode = 14 THEN
        DO
          ADDRESS ISPEXEC "SETMSG MSG(CMSGSØØ5)"
          EXIT
        END
     ELSE
        IF retcode = 16 THEN
           DO
             ADDRESS ISPEXEC "SETMSG MSG(CMSGSØØ6)"
             EXIT
           END
        ELSE
           DO
             ADDRESS ISPEXEC "SETMSG MSG(CMSGSØØØ)"
```

```
          END
   END
EXIT
```

CURSOR-BASED DATASET LIST UTILITY MACRO (CM)

The cursor-based dataset list utility macro is defined in the ISPF ISPEXEC library as CM.

```
/***REXX*************************************************************/
ADDRESS ISREDIT
'MACRO PROCESS'

ADDRESS ISPEXEC 'CONTROL ERRORS RETURN'

'(cl,cc) = CURSOR'
IF cc = Ø THEN EXIT

'(LINE) = LINE' cl
UPPER line

valid = XRANGE('A','Z') XRANGE('Ø','9') '@#$£(.)'

end = LENGTH(line)
start = end - VERIFY(REVERSE(line) ' ',valid,'N',end-cc+1) + 2
len = VERIFY(line' ',valid,'N',cc) - start

IF len < 1 THEN EXIT

dsn = SUBSTR(line,start,len)

IF VERIFY(dsn,'.','M') = Ø THEN DO
   '(dataset) = DATASET'
   dsn = dataset'('dsn')'
END

ADDRESS ISPEXEC "VGET ZDLDSNLV"
Saved_zdldsnlv = ZDLDSNLV
ZDLDSNLV = dsn
ADDRESS ISPEXEC "LMINIT DATAID(dataid) DATASET('"dsn"')"
ADDRESS ISPEXEC "LMOPEN DATAID("dataid") OPTION(INPUT)"
ADDRESS ISPEXEC "LMMLIST DATAID("dataid")"
IF rc > Ø THEN
   DO
     retcode = rc
     ADDRESS ISPEXEC "LMFREE DATAID("dataid")"
     SIGNAL Error_check
   END
```

```
ADDRESS ISPEXEC "VPUT ZDLDSNLV"
ADDRESS ISPEXEC "CONTROL NONDISPL ENTER"
ADDRESS ISPEXEC "SETMSG MSG(CMSGS001)"
ADDRESS ISPEXEC "SELECT PGM(ISRUDL) PARM(ISRUDLP)"

SIGNAL Endit

Error_check:
IF retcode = 4 THEN
   DO
     ADDRESS ISPEXEC "SETMSG MSG(CMSGS002)"
     SIGNAL Endit
   END
ELSE
   DO
     IF retcode = 12 THEN
        DO
          ADDRESS ISPEXEC "SETMSG MSG(CMSGS003)"
          SIGNAL Endit
        END
     ELSE
        IF retcode > 15 THEN
           DO
             ADDRESS ISPEXEC "SETMSG MSG(CMSGS000)"
             EXIT
           END
   END
Endit:
ZDLDSNLV = Saved_zdldsnlv
ADDRESS ISPEXEC "VPUT ZDLDSNLV"
EXIT
```

CURSOR-BASED MACROs

The following messages are defined in the ISPF ISPMLIB library as CMSGS00.

```
CMSGS000 '' .ALARM=YES .WINDOW=RESP
'Unspecified error accessing &dsn - may not exist'
CMSGS001 '' .ALARM=YES .WINDOW=RESP
'Enter line command M for member list processing'
CMSGS002 '' .ALARM=YES .WINDOW=RESP
'&dsn contains no members'
CMSGS003 '' .ALARM=YES .WINDOW=RESP
'&dsn not a PDS'
CMSGS004 '' .ALARM=YES .WINDOW=RESP
'&dsn no data changed/saved'
CMSGS005 '' .ALARM=YES .WINDOW=RESP
'&dsn in use'
```

```
CMSGSØØ6 '' .ALARM=YES .WINDOW=RESP
'Specified member not found or no members in &dsn'
CMSGSØØ7 '' .ALARM=YES .WINDOW=RESP
'Zero length data set, empty dataset or empty member'
CMSGSØØ8 '' .ALARM=YES .WINDOW=RESP
'Specified member not found'
CMSGSØØ9 '' .ALARM=YES .WINDOW=RESP
'No members in &dsn'
```

CUT

The cursor macro is defined in the ISPF ISPEXEC library as CUT.

```
/**REXX*************************************************************/
ADDRESS ISPEXEC
'ISREDIT MACRO (PARM1) NOPROCESS'
linestocut = Ø
IF parm1 = '?' THEN
   DO
     CALL help
     Exit
   END

'ISREDIT PROCESS RANGE C M'
SELECT
  WHEN rc = Ø THEN
    DO
      'ISREDIT (CMD) = RANGE_CMD'
      'ISREDIT (LINE1) = LINENUM .ZFRANGE'
      'ISREDIT (LINE2) = LINENUM .ZLRANGE'
      linestocut = line2 - line1 + 1
    END
  WHEN rc <= 4 THEN
    DO
      zcmd = ' '
      zedsmsg = 'Enter "C"/"M" line cmd'
      zedlmsg = 'CUT requires a "C" or "M" line command'
      'SETMSG MSG(ISRZØØ1)'
      Exit 12
    END
  OTHERWISE
    Exit 12
END

cutcnt = Ø

DO i = line1 to line2
   cutcnt = cutcnt + 1
   'ISREDIT (CL'cutcnt') = LINE' i
```

```
   INTERPRET "CL"cutcnt"= STRIP(CL"cutcnt",'T')"
   'VPUT (CL'cutcnt') PROFILE'
END

'VPUT (CUTCNT) PROFILE'

IF cmd = 'M' THEN
   DO
     'ISREDIT DELETE 'line1 line2
      zcmd = ' '
      zedsmsg = linestocut 'lines cut and deleted'
      msg = 'lines were cut and deleted from the current file'
      zedlmsg = linestocut msg
     'SETMSG MSG(ISRZ000)'
   END
ELSE
   DO
     zcmd = ' '
     zedsmsg = linestocut 'lines cut'
     zedlmsg = linestocut 'lines were cut from the current file'
     'SETMSG MSG(ISRZ000)'
   END

Exit

Help:
Say '
'
Say ' ISPF/PDF edit macro to write lines from a file to the users   '
Say ' PROFILE pool for later inclusion by the PASTE macro.       '
Say '                                                             '
Say ' To run:                                                     '
Say ' Enter CUT at the COMMAND ===> prompt and use C or M line    '
Say ' commands to select the lines to be cut.                     '
Say '                                                             '
Say ' If the M line command is used, the selected lines will be   '
Say ' deleted from this dataset.                                  '
Say '                                                             '
RETURN
```

PASTE

The paste macro is defined in the ISPF ISPEXEC library as PASTE.

```
/**REXX*************************************************************/
ADDRESS ISPEXEC
'ISREDIT MACRO (PARM1) NOPROCESS'
linestocut = 0
trunccnt = 0
```

```
line1 = Ø
lptr = Ø
IF parm1 = '?' THEN
   DO
     CALL help
     Exit
   END
'CONTROL ERRORS RETURN'
'ISREDIT PROCESS DEST'
SELECT
  WHEN rc = Ø THEN
    DO
      'ISREDIT (lptr) = LINENUM .ZDEST'
    END
  WHEN rc <= 8 THEN
    DO
      zedsmsg = 'Enter "A"/"B" line command'
      zedlmsg = 'PASTE requires an "A" or "B" line command'
      'SETMSG MSG(ISRZØØ1)'
      Exit 12
    END
  WHEN rc <= 2Ø THEN
    EXIT 12
  WHEN rc = 2Ø THEN
    lptr = Ø
  OTHERWISE
    Exit 12
END
'CONTROL ERRORS CANCEL'
'VGET (CUTCNT) PROFILE'
IF rc ª= Ø THEN
   DO
      zedsmsg = 'Use CUT before PASTE'
      zedlmsg = 'No data has been stored via CUT macro'
      'SETMSG MSG(ISRZØØ1)'
      Exit 12
   END
IF cutcnt = Ø THEN
   DO
      zedsmsg = 'Use CUT before PASTE'
      zedlmsg = 'No data has been stored via CUT macro'
      'SETMSG MSG(ISRZØØ1)'
      Exit 12
   END

trunccnt = Ø
cutcount = cutcnt

DO i = cutcnt to 1 by -1
   'VGET (CL'i') PROFILE'
   'ISREDIT LINE_AFTER 'lptr' = DATALINE (CL'i')'
```

```
   IF rc = 4 THEN
      DO
        trunccnt = trunccnt + 1
      END
END

'VGET (ZENVIR) SHARED'
DO i = 1 to cutcnt by 1
   'VERASE (CL'i') PROFILE'
END
cutcnt = Ø
'VPUT (CUTCNT) PROFILE'

IF trunccnt > Ø THEN
   DO
      zedsmsg = trunccnt ' lines truncated'
      msg = 'Current rec.len. shorter than origin'
      zedlmsg = msg '-' trunccnt ' of ' cutcount ' recs truncated'
      'SETMSG MSG(ISRZØØ1)'
   END

line1 = lptr + 1
'ISREDIT CURSOR = 'line1 Ø
Exit

Help:
Say '                                                                '
Say ' ISPF/PDF edit macro to write lines from the user PROFILE pool '
Say ' into the current file. This macro is used in conjunction with '
Say ' the CUT macro.                                                 '
Say '                                                                '
Say ' To run:                                                        '
Say ' Enter PASTE at the COMMAND ===> prompt and use A or B line     '
Say ' commands to select the location at which the lines will be     '
Say ' pasted.                                                        '
Say '                                                                '
RETURN
```

D L Mathews
ACF Systems Ltd (UK)

Year 2000 aid: generation of edit macros

INTRODUCTION

The enclosed Assembler modules (YEAR2KE and YEAR2KER) produce ISPF edit macros by providing the STDEF macros as defined in the descriptions of programs YEAR2K and YEAR2KR, respectively.

After assembling, the Assembler output from SYSPUNCH is moved to an appropriate ISPF library. Suggested member names are YEAR2K and YEAR2KR, respectively.

When a source module is edited and the above macro commands are executed the result is very similar to the effects of executing the batch programs YEAR2K and YEAR2KR, respectively. It is suggested that the YEAR2KR macro be used to test the effects of string replacements prior to using the YEAR2KR program, thus reducing the risks of unintentioned results.

YEAR2KE

```
*  THE ASSEMBLY OF THIS PROGRAM PUNCHES AN ISPF EDIT MACRO TO
*  REPLACE STRINGS AS PRESCRIBED BY ITS 'STDEF' MACROS.  IT IS
*  EXPECTED THAT THESE MACROS WILL EITHER BE EXTRACTED FROM
*  PROGRAM 'YEAR2K' OR SIMILAR MACROS WILL BE MANUALLY CODED.
*
*  THE RESULTING EDIT MACRO SHOULD BE PLACED IN AN APPROPRIATE
*  ISPF LIBRARY.  SUGGESTED NAME IS YEAR2K.
*
         MACRO
&NAME    DOQ   &A
.*
.*  IF THE OPERAND BEGINS WITH A SINGLE QUOTE (') THE BEGINNING AND
.*  ENDING CHARACTERS ARE REMOVED.
.*
.*  THE RESULT IS ENCLOSED IN DOUBLE QUOTES (") AND LACED IN THE
.*  GLOBAL SYMBOL &UNQ.
.*
         GBLC  &UNQ
.*
&UNQ     SETC  '&A'
         AIF   ('&A'(1,1) NE '''').NOTQ
&UNQ     SETC  '&A'(2,K'&A-2)
```

```
.*
.NOTQ    ANOP
&UNQ     SETC  '"&UNQ"'
         MEND
*
         MACRO
.*
.* THIS MACRO CREATES AN ISPF EDIT MACRO TO REPLACE STRINGS AS
.* SPECIFIED FROM THE STDEF MACROS EXTRACTED FROM PROGRAM YEAR2K.
.*
&NAME    STDEF &A,&B,&C,&D
.*
         GBLC  &UNQ
         LCLA  &I
         LCLC  &T
.*
         AIF   (T'&A NE 'O').NOTNULL
         MNOTE 8,'NULL STRING NOT ALLOWED'
         MEXIT
.*
.NOTNULL ANOP
.*
         DOQ   &A
&T       SETC  '&UNQ'
.*
&I       SETA  73-K'&T
.*
         AIF   ('&B' NE 'P' AND '&C' NE 'P' AND '&D' NE 'P').NOTP
         PUNCH ' ISREDIT FIND 1 &I ALL PREFIX &T'
.*
.NOTP    AIF   ('&B' NE 'S' AND '&C' NE 'S' AND '&D' NE 'S').NOTS
         PUNCH ' ISREDIT FIND 1 &I ALL SUFFIX &T'
.*
.NOTS    AIF   ('&B' NE 'W' AND '&C' NE 'W' AND '&D' NE 'W').NOTW
         PUNCH ' ISREDIT FIND 1 &I ALL WORD &T'
.*
.NOTW    AIF   (T'&B NE 'O').END
         PUNCH ' ISREDIT FIND 1 &I ALL &T'
.END     MEND
*
         PUNCH 'ISREDIT MACRO (HELP) NOPROCESS'
         PUNCH 'ISPEXEC CONTROL ERRORS RETURN'
         PUNCH ' IF &&HELP = ? THEN DO '
         PUNCH '      ISPEXEC DISPLAY PANEL(YEAR2KP)'
         PUNCH '      EXIT'
         PUNCH '    END'
         PUNCH '  ISREDIT EXCLUDE ALL'
*********************************************************************
***                                                               ***
***   THE 'STDEF' MACRO INSTRUCTIONS, BELOW, WERE CUT FROM SOURCE   ***
```

```
***   'YEAR2K' AND PASTED THERE.                                        ***
***                                                                     ***
************************************************************************
*
         STDEF AGE,W,P
         STDEF BIRTH,W,P
         STDEF CALENDAR
         STDEF CENTURY
         STDEF CSADAT
         STDEF CSAEID
         STDEF CSAJYD
         STDEF DATE,W,P
         STDEF DMY
         STDEF GREGJUL
         STDEF GREGORIAN
         STDEF JULGREG
         STDEF JULIAN
         STDEF MDY
         STDEF MMDDYY
         STDEF SCHEDULE
         STDEF TODAY,W
         STDEF YEAR
*        STDEF YD,P,S,W
         STDEF YDD
         STDEF YM,P,S,W
         STDEF YMD
         STDEF YY
************************************************************************
***                                                                   ***
***   THE 'STDEF' MACRO INSTRUCTIONS, ABOVE, WERE CUT FROM SOURCE     ***
***   'YEAR2K' AND PASTED THERE.                                      ***
***                                                                   ***
************************************************************************
         PUNCH '  ISREDIT EXCLUDE ALL DATE-WRITTEN.'
         PUNCH '  ISREDIT EXCLUDE ALL DATE-COMPILED.'
         PUNCH '  EXIT CODE(Ø)'
         END
```

YEAR2KER

```
*
*  THE ASSEMBLY OF THIS PROGRAM PUNCHES AN ISPF EDIT MACRO TO
*  REPLACE STRINGS AS PRESCRIBED BY ITS 'STDEF' MACROS.  IT IS
*  EXPECTED THAT THESE MACROS WILL EITHER BE EXTRACTED FROM
*  PROGRAM 'YEAR2KR' OR SIMILAR MACROS WILL BE MANUALLY CODED.
*
*  THE RESULTING EDIT MACRO SHOULD BE PLACED IN AN APPROPRIATE\
*  ISPF LIBRARY.  SUGGESTED NAME IS YEAR2KR.
*
```

```
         MACRO
&NAME    DOQ   &A
.*
.*  IF THE OPERAND BEGINS WITH A SINGLE QUOTE (') THE BEGINNING AND
.*  ENDING CHARACTERS ARE REMOVED.
.*
.*  THE RESULT IS ENCLOSED IN DOUBLE QUOTES (") AND LACED IN THE
.*  GLOBAL SYMBOL &UNQ.
.*
         GBLC  &UNQ
.*
&UNQ     SETC  '&A'
         AIF   ('&A'(1,1) NE '''').NOTQ
&UNQ     SETC  '&A'(2,K'&A-2)
.*
.NOTQ    ANOP
&UNQ     SETC  '"&UNQ"'
         MEND
*
         MACRO
.*
.* THIS MACRO CREATES AN ISPF EDIT MACRO TO SEARCH FOR STRINGS AS
.* SPECIFIED FROM THE STDEF MACROS EXTRACTED FROM PROGRAM YEAR2K.
.*
&NAME    STDEF &A,&R,&B,&C,&D,&OPTION=
.*
         GBLC  &UNQ
         LCLA  &K,&I
         LCLC  &T,&O
.*
         AIF   (T'&A NE 'O').NOTNULL
         MNOTE 8,'NULL TARGET STRING NOT ALLOWED'
         MEXIT
.*
.NOTNULL AIF   (T'&R NE 'O').NOTNULR
         MNOTE 8,'NULL OBJECT STRING NOT ALLOWED'
         MEXIT
.NOTNULR ANOP
.*
```

Editor's note: this article will be continued next month when the rest of the code will be published.

Keith H Nicaise
Technical Services Manager
Touro Infirmary (USA)

MVS news

Neon Systems has launched the ShadowWebserver component of its ShadowDirect middleware family, providing direct access from Web browsers to any MVS data source or transactional application.

Built on the ShadowDirect architecture, the product is claimed to scale across thousands of users and has in-built systems management. It combines multiple levels of security, including support for SSL and native MVS security packages such as RACF, with an automated control system for imposing data centre policy on a Web-connected user community, and taking appropriate action automatically.

For further information contact:
Neon Systems Inc, 14141 Southwest Freeway, Suite 6200, Sugar Land, TX 77478, USA.
Tel: (281) 491 4200
Fax: (281) 242 3880 or
Neon Systems UK Ltd, Third Floor, Sovereign House, 26-30 London Road, Twickenham, Middlesex, TW1 3RW, UK.
Tel: (0181) 607 9911
Fax: (0181) 607 9933.

* * *

Intersolv has announced Version 4.0 of its APS MVS designed for speeding up preparation of legacy systems for Y2K compliance. It checks values of the current date that can cause errors beyond 31 December 1999, and predictably handles date manipulations including calculations branching, format, and storage. It also logically handles dates entered or imported without explicit century identification, and stores and displays four-digit years in interfaces. The product uses COBOL for MVS as the runtime environment. Date edit facilities now allow a choice of input, internal storage, and output formats, which users of versions prior to 3.0.1 can't take advantage of. There's also an option to use a set of callable functions available with COBOL for MVS to retrieve dates in four-digit formats.

For further information contact:
Intersolv Inc, 9420 Key West Avenue, Rockville, MD 20850, USA.
Tel: 301 230 3200
Fax: 301 231 7813 or
Intersolv Plc, Abbey View, Everard Close, St Albans, AL1 2PS, UK.
Tel: 01727 812812
Fax: 01727 869804.

* * *

IBM has announced Release 2 of SmartBatch for OS/390, both an enhancement to, and a replacement for, BatchPipes/MVS. It's geared to both reducing the time needed for batch jobstreams and helping to migrate them to take advantage of Parallel Sysplex environments. Specific enhancements include Batch Accelerator, which splits batch jobs into units that can be executed in parallel on multiple OS/390 and MVS/ESA systems. During execution, it manages the I/O piping and/or data sharing and merges the results back into a single job image.There's now support for jobs processed by JES3.

Contact your local IBM representative for further information.

* * *