June 1999

In this issue

3 Creating an MVS mini system for OS/390
15 Locating members in concatenated PDSs
24 Listing ICF catalog entries
40 JES2 checkpoint sizing
43 On-line explanation of OS/390 system messages
51 An IPL subsystem
72 MVS news

© Xephon plc 1999
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 £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.
Creating an MVS mini system for OS/390

INTRODUCTION
This article describes a minimum OS/390 system, located on one disk, which will run JES, VTAM, TSO, and ISPF. Initially, it was destined for a P/390 system running OS/390 Version 2 Release 5. I had only 590 spare cylinders of 3380 for this. So only the necessary files from my current system were copied. I had to abandon SDSF, DFSort, OpenEdition, TCP/IP, etc. Such a system should be saved on one 3480 cartridge (see the last step of the procedure) in order to be stand-alone restored. You may want to adapt it to your site. It is best fit for OS/390, RACF and JES2. Thus TSS or JES3 sites will have to change it or add steps.

WHY A MINI SYSTEM?
A mini system is mainly for safety. With a mini MVS system on disk you can re-IPL immediately in the case of a big crash on your present SYSRES (or on other important volumes) in order to be able to restore what has been damaged. Also, you avoid using DFDSS stand-alone, which is two or three times slower than DFDSS executing under MVS. If you use PR/SM (or its equivalent), you can use a mini MVS system to test a new partition quickly. When you do back-up tests on a disaster recovery site, the mini system is much appreciated. After restoring the mini MVS system and IPLing it (no loadparm is needed, just specify the disk unit address), you log-on and submit jobs to restore the remaining volumes. Eventually you can IPL on the restored system.

In my own case, as several VTAM tables were missing, I had to log-on to TSO using the command:

```
LOGON APPLID(TSO) LOGMODE(NSX32702)
```

Keeping a mini MVS system on tape preceded by the DFDSS stand-alone program (rather than on DASD) enables you to restore it to disk immediately after IPLing from the 3480 unit and entering the DFDSS stand-alone parameters. Fortunately, DFDSS stand-alone has been enhanced and its syntax is now more forgiving. For example, this is the input I had to type to restore the volume:
RESTORE FROMDEV(TAPE) FROMADDR(56Ø) TOADDR(12A) -
VFY(MINISY) FULL FILE(2)

File 2 contains the data to be restored, while file 1 is the DFDSS SA program created by the ADRDSSU BUILDSA command.

Of course, you will not create a clone of your production system, but only a small MVS to be used temporarily. All the files are the smallest possible (there are no SMF, STGINDEX, DUMPxx, or NCPLIB files). The only RACF/TSO user-id will be IBMUSER. At IPL time you must expect some messages and replies because of the lack of some system files or PARMLIB members. The very first time, you must CLPA the system and cold-start JES2. As soon as your mini system has been tested, you may change parameters to CVIO and JES2 warm-start; then the disk may be saved.

The following steps create a mini MVS system with JES2, RACF, VTAM, TSO, and ISPF on a disk called MINISY.

**Initialize**

Initialize MINISY with IPL text using ICKDSF. The disk must be off-line. You must modify the unit number (12A here).

```verbatim
//INIT1 EXEC PGM=ICKDSF
//SYSPRINT DD SYSOUT=* 
//DDI DD DSN=SYS1.SAMPLIB(IEAIPL00),DISP=SHR 
//SYSIN DD *
    INIT UNIT(12A) IPLDD/DDI/ NOVERIFY -
    VTOC(1,0,60) INDEX(0,1,14) NOCONTINUE PURGE NOCHECK
```

**Allocate**

Allocate MINI.PARMLIB, MINI.LOGREC, MINI.PROCLIB, MINI.UADS, MINI.BRODCAST, MINI.HASPCKPT, and MINI.HASPACE on it using IEFBR14.

```verbatim
//ALLOC02 EXEC PGM=IEFBR14
/*
//DD1 DD DISP=(NEW,KEEP),VOL=SER=MINISY,UNIT=SYSALLDA,
// SPACE=(CYL,(1,1,2)),DCB=(LRECL=80,BLKSIZE=9040,RECFM=FB),
// DSN=MINI.PARMLIB
//DD2 DD DISP=(NEW,KEEP),VOL=SER=MINISY,UNIT=SYSALLDA,
// DCB=DSORG=PSU,SPACE=(CYL,20),DSN=MINI.HASPACE
//DD3 DD DISP=(NEW,KEEP),VOL=SER=MINISY,UNIT=SYSALLDA,
// DCB=SYS1.LOGREC,SPACE=(TRK,2),DSN=MINI.LOGREC
//DD4 DD DISP=(NEW,KEEP),VOL=SER=MINISY,UNIT=SYSALLDA,
```
Change the high-level identifier

The High level qualifier should be changed from MINI to SYS1 using IEHPROGM.

```
//RENAME3 EXEC PGM=IEHPROGM
//DD1      DD DISP=SHR,UNIT=SYSALLDA,VOL=SER=MINISY
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
RENAME DSNAME=MINI.PARMLIB,VOL=SYSALLDA=MINISY,NEWNAME=SYS1.PARMLIB
RENAME DSNAME=MINI.HASPACE,VOL=SYSALLDA=MINISY,NEWNAME=SYS1.HASPACE
RENAME DSNAME=MINI.LOGREC,VOL=SYSALLDA=MINISY,NEWNAME=SYS1.LOGREC
RENAME DSNAME=MINI.HASPCKPT,VOL=SYSALLDA=MINISY,NEWNAME=SYS1.HASPCKPT
RENAME DSNAME=MINI.PROCLIB,VOL=SYSALLDA=MINISY,NEWNAME=SYS1.PROCLIB
RENAME DSNAME=MINI.UADS,VOL=SYSALLDA=MINISY,NEWNAME=SYS1.UADS
RENAME DSNAME=MINI.BRODCAST,VOL=SYSALLDA=MINISY,NEWNAME=SYS1.BRODCAST
```

Define the future master catalog

Defining the master catalog should be done using IDCAMS. Also, define a USERCAT on MINISY. You may optionally define all the aliases of your driving system in it.

```
//DEFMCAT4 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//CATVOL DD VOL=SER=MINISY,UNIT=SYSALLDA,DISP=OLD
//SYSIN DD *
DELETE CATALOG.MINIMVS.MINISY UCAT RECOVERY
SET MAXCC = Ø
DEFINE UCAT (ICFCAT -
   NAME(CATALOG.MINIMVS.MINISY) -
   FILE(CATVOL) VOL(MINISY) CYL(1 1) SHR(3 3))
```

Copy the main libraries of your current system

You must copy all the main system libraries to MINISY. Adapt this JCL if dataset names are different at your site (dataset names here are those delivered by IBM by default).
COPY DS(INC(SYS1.LINKLIB, SYS1.LPALIB, SYS1.SVCLIB, SYS1.CMDLIB, -
SYS1.NUCLEUS, SYS1.CSSLIB, SYS1.MIGLIB, -
SYS1.VTAMLIB, SYS1.VTAMLST, SYS1.SISTCLIB, -
SYS1.SHASLINK, -
ISP.SISPEXEC, ISP.SISPLLIB, -
ISP.SISPMENU, ISP.SISPPENU, -
ISP.ISPSMENU, ISP.ISPSPTENU, -
SYS1.SISPLPA)) -
ODY(MINISY) TOL(ENQF) WAIT(Ø,Ø) SHR BYPASSACS(**)

Copy some modules to the target system
You might copy some exits that you think are of interest (IEFACTRT, the end-of-job exit), or some modules that were not located in SYS1.LINKLIB or SYS1.LPALIB (I had to reintegrate ICHRIN03 because it was outside my LPALIB and RACF will not start if it is missing).

Define three page datasets
Define PLPA (30 cylinders), COMMON (20 cylinders), and LOCAL (20 cylinders). Since I had 128MB of main storage for my system, the local page dataset is hardly used after the IPL, so a minimum size will do.

//DEFPGSP7 EXEC PGM=IDCAMS
//STECAT DD DISP=SHR, DSN=CATALOG.MINIMVS.MINISY
//SYSPRINT DD SYSOUT=* 
//SYSIN DD * 
DEF PGSPC (NAME(SYS1.PAGE.VMINISY.PLPA) -
VOL(MINISY) FILE(DD1) CYL(30) UNIQUE ) -
CAT(CATALOG.MINIMVS.MINISY)
DEF PGSPC (NAME(SYS1.PAGE.VMINISY.COMMON) -
VOL(MINISY) FILE(DD1) CYL(20) UNIQUE ) -
CAT(CATALOG.MINIMVS.MINISY)
DEF PGSPC (NAME(SYS1.PAGE.VMINISY.LOCAL) -
VOL(MINISY) FILE(DD1) CYL(20) UNIQUE ) -
CAT(CATALOG.MINIMVS.MINISY)
Catalogue in the new master catalog

Catalogue in the new master catalog all the files created in steps 2 (allocation) and 5 (copy on MINISY) using DEFINE NONVSAM. Also create an entry for the RACF database.

//DEFNVSAM EXEC  PGM=IDCAMS
//STEPCAT    DD DISP=SHR,DSN=CATALOG.MINIMVS.MINISY
//SYSPRINT   DD SYSOUT=*  
//SYSSIN DD *
DEF NVSAM(NAME(SYS1.RACFMNI) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME(SYS1.LINKLIB) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME(SYS1.LPALIB) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME(SYS1.CSSSLIB) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME(SYS1.SVCLIB) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME(SYS1.SYSLIB) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME(SYS1.MIGLIB) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME(SYS1.NUCLEUS) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME(SYS1.PARMLIB) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME(SYS1.PROCLIB) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME(SYS1.UADS) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME(SYS1.BRODCAST) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME(SYS1.VTAMLIB) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME(SYS1.SISTCLIB) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME(SYS1.VTAMLST) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME( SYS1.SHASLINK       ) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME( SYS1.SISPLPA       ) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME( ISP.SISPLOAD       ) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME( ISP.SISPPENU       ) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME( ISP.SISPSLIB       ) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
DEF NVSAM(NAME( ISP.SISPMENU       ) DEVT(ØØØØ) VOL(******)) - CAT(CATALOG.MINIMVS.MINISY)
Create necessary members in the new SYS1.PARMLIB

You should adapt COMMND00 (for starting VTAM), LOAD00 (config name) and CONSOL00 (for your master console). A CLOCK00 member may be added if you want to avoid clock prompting during the IPL. Member IFAPRD00 was necessary to enable RACF.

```
//MAJPARM9 EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD SYSOUT=* 
//SYSUT2 DD DISP=SHR,DSN=SYS1.PARMLIB,UNIT=SYSALLDA,VOL=SER=MINISY
//SYSIN DD DATA,DLM=$$
./ ADD NAME=IEASYSØØ
CVIO, CLPA OR CVIO
CMB=(UNITR,COMM,GRAPH,CHRDR), ADDITIONAL CMB ENTRIES
CMD=ØØ,
CON=ØØ, SELECT CONSOLØØ
CSA=(2000,30000),
GRS=NONE, NO COORDINATION OF GRS REQUESTS
PROG=ØØ,
PROD=ØØ,
LNKAUTH=LNKLST, MVS/XA 2.1.1 DEFAULT, APFTAB IS ALT
LOGCLS=L, WILL NOT BE PRINTED BY DEFAULT
LOGLMT=999999, MUST BE 6 DIGITS, MAX WTL MESSAGES QUEUED
MAXUSER=250, (SYS TASKS + INITS + TSOUSERS)
PAGTOTL=(9,2), ALLOW ADDITION 3 PAGE D/S AND 2 SWAP D/S
OPI=YES, ALLOW OPERATOR OVERRIDE TO IEASYSØØ
PAGE=(SYS1.PAGE.VMINISY.PLPA,
    SYS1.PAGE.VMINISY.COMMON,
    SYS1.PAGE.VMINISY.LOCAL.L),
REAL=128, ALLOWS 2 64K JOBS OR 1 128K JOB TO RUN V=R
RSU=Ø, NO RECONFIG STORAGE UNITS DEFAULT
RSVSTRT=5, RESERVED ASVT ENTRIES DEFAULT
RSVNONR=5, RESERVED ASVT ENTRIES DEFAULT
SSN=ØØ,
VAL=ØØ,
SYSNAME=MINI,
VIODSN=IGNORE,
VRREGN=64 DEFAULT REAL-STORAGE REGION SIZE DEFAULT
./ ADD NAME=IFAPRDØØ
```
WHEN (HWNAME(*))
PRODUCT OWNER('IBM CORP')
  NAME(OS/390)
  ID(5647-A01)
  VERSION(*) RELEASE(*) MOD(*)
  FEATURENAME(OS/390)
  STATE(ENABLED)

PRODUCT OWNER('IBM CORP')
  NAME(OS/390)
  ID(5647-A01)
  VERSION(*) RELEASE(*) MOD(*)
  FEATURENAME(DFSMSDSS)
  STATE(ENABLED)

PRODUCT OWNER('IBM CORP')
  NAME(OS/390)
  ID(5647-A01)
  VERSION(*) RELEASE(*) MOD(*)
  FEATURENAME('SECURITY SERVER')
  STATE(ENABLED)

./ ADD NAME=PROG00
APF FORMAT(DYNAMIC)
APF ADD DSNAME(SYS1.LINKLIB) VOLUME(MINISY)
APF ADD DSNAME(SYS1.SHASLINK) VOLUME(MINISY)
APF ADD DSNAME(SYS1.VTAMLIB) VOLUME(MINISY)
APF ADD DSNAME(SYS1.SISPLPA) VOLUME(MINISY)
APF ADD DSNAME(SYS1.SISTCLIB) VOLUME(MINISY)
APF ADD DSNAME(ISP.SISPLOAD) VOLUME(MINISY)
LNKLST DEFINE NAME(LNKLST00)
LNKLST ADD NAME(LNKLST00) DSN(SYS1.LINKLIB)
LNKLST ADD NAME(LNKLST00) DSN(SYS1.MIGLIB)
LNKLST ADD NAME(LNKLST00) DSN(SYS1.CSSLIB)
LNKLST ADD NAME(LNKLST00) DSN(SYS1.CMDLIB)
LNKLST ACTIVATE NAME(LNKLST00)

./ ADD NAME=LOAD00
IODF ØØ SYS1 CBIP0 ØØ
NUCLEUS 1
SYSCAT MINISY113CCATALOG.MINIMVS.MINISY
SYSPARM ØØ

./ ADD NAME=VATLST00
VATDEF IPLUSE(PRIVATE) SYSUSE(PRIVATE)
MINISY,1.0,3380 ,N STORAGE ** STORAGE FOR IBMUSER TO LOG ON

./ ADD NAME=IEFSSN00
JES2...,PRIMARY,NOSTART

./ ADD NAME=IGDFFPKG
DFSMS_OFFERING=(MINI,FULL)

./ ADD NAME=COMMND00
COM='D T'
COM='S JES2,PARM='WARM,NOREQ''
COM='S NET...,(LIST=Ø1) START VTAM FOR LOCAL TERMINALS'
COM='S TSO AUTOMATIC START OF TSO'

ADD NAME=JES2PARM

CONDEF
AUTOCMD=52, BUFNUM=300, BUFWARN=80, CONCHAR=$,
MASMSG=200, RDRCHAR=$

SMFDEF
BUFNUM=10, BUFWARN=80

CKPTDEF
CKPT=(DSN=SYS1.HASPCKPT, VOL=MINISY, INUSE=YES)

SPOOLDEF
BUFSIZE=3992, DSN=SYS1.HASPSPACE, FENCE=NO,
SPOLNUM=32, TGBPRL=10, TNUM=32576, TGSIZE=30,
TGWARN=90, TRKCELL=3, VOLUME=MINIS

INITDEF
PARTNUM=5

I1 START, NAME=A, CLASS=X
I2 START, NAME=B, CLASS=X
I3 START, NAME=C, CLASS=X
I4 START, NAME=D, CLASS=X
I5 START, NAME=E, CLASS=X

INTRDR
CLASS=B, RDINUM=25

JOBDEF
ACCTFLD=IGNORE, JCLERR=NO, NOBNUM=3000, JNUM=80,
PRTRYHIGH=10, PRTRYJECL=NO, PRTRYJOB=NO, PRTRYLOW=1,
PRTRYRATE=0, RANGE=(1-9999)

JOBCLASS(A-Y)
ACCT=NO, PGMNAME=NO, TIME=(60, 0), REGION=8M,
COMMAND=DISPLAY, BLP=YES, AUTH=ALL, MSGLEVEL=(1,1),
JOURNAL=NO

STCCLASS
TIME=(60, 0), REGION=8M, COMMAND=DISPLAY, BLP=YES,
AUTH=ALL, MSGLEVEL=(1,1), IEFJEP=NO, IEFUOS=NO,
LOG=NO, OUTPUT=YES, PERFORM=0, PROCLIB=00,
TYPE6=YES, TYPE26=YES, MSGCLASS=Z

TSUCLASS
TIME=(60, 0), REGION=8M, COMMAND=DISPLAY, BLP=YES,
AUTH=ALL, MSGLEVEL=(1,1), IEFJEP=NO, IEFUOS=NO,
LOG=NO, OUTPUT=YES, PERFORM=0, PROCLIB=00,
TYPE6=YES, TYPE26=YES, MSGCLASS=Z

OUTDEF
COPIES=30, DMNDSET=NO, JOENUM=3000, JOEWRN=80,
PRTRYHIGH=0, PRTRYLOW=0, PRTRYOUT=NO, STDFORM=STANDARD,
USERSET=NO

OUTCLASS(X)
OUTDISP=(HOLD), OUTPUT=PRINT, TRKCELL=YES

USERMAX=100.
RECONLIM=10.
BUFRSIZE=132.
HIBFREXT=6600.
LOBFREXT=3300.
CHNLN=4.

ADD NAME=SMFPRM00

NOACTIVE /*NO ACTIVE SMF RECORDING*/

NOPROMPT /*DO NOT PROMPT OPERATOR FOR OPTIONS*/

ADD NAME=Couple00

COUPLE SYSPLEX(LOCAL)

ADD NAME=CONSOL00

INIT PFK(00) MONITOR(DSN=NAME) MLIM(1500) RLIM(10) UEXIT(N)
CMDDELIM(:)

DEFAULT ROUTCODE(ALL)
CONSOLE DEVNUM(700) ALTERNATE(F07) ROUTCODE(ALL)

© 1999. Xephon UK telephone 01635 33848, fax 01635 33845. USA telephone (940) 455 7050, fax (940) 455 2492.
Update procedures in your new SYS1.PROCLIB

Update the following procedures in SYS1.PROCLIB:

- JES2 (with no user PROCLIB)
- NET (with only SYS1.VTAMLST)
- IKJS – a TSO LOGON procedure, with all ISPF files, and a temporary //ISPPROF.

```
//MAJPRC10 EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD SYSOUT=*  
//SYSUT2 DD DISP=SHR,DSN=SYS1.PROCLIB,UNIT=SYSALLDA,VOL=SER=MINISY  
//SYSIN DD DATA,DLM=$$
./  ADD NAME=TSO
//TSO PROC MBR=TSOKEYØØ
//STEP1 EXEC PGM=IKTCASØØ,TIME=144Ø  
//PARMLIB DD DSN=SYS1.PARMLIB(&MBR),DISP=SHR,FREE=CLOSE  
//PRINTOUT DD SYSOUT=*,FREE=CLOSE  
//*
./  ADD NAME=JES2
//JES2 PROC M=JES2PARM
//IEFPROC EXEC PGM=HASJES2Ø,TIME=144Ø,DPRTY=(15,15)
//HASPPLIST DD DDNAME=IEFRDER
//HASPPARM DD DSN=SYS1.PARMLIB(&M),DISP=SHR
//PROCØØ DD DSN=SYS1.PROCLIB,DISP=SHR
//STEPLIB DD DSN=SYS1.SHASLINK,DISP=SHR
./  ADD NAME=NET
//VTMLCL PROC
//VTMLCL EXEC PGM=ISTINMØ1,REGION=2Ø48K,  
//   DPRTY=(15,15),TIME=144Ø,PERFORM=8
//VTAMLST DD DSN=SYS1.VTAMLST,DISP=SHR
//VTMLIB DD DSN=SYS1.VTAMLIB,DISP=SHR
//SISTCLIB DD DSN=SYS1.SISTCLIB,DISP=SHR
//SYSABEND DD SYSOUT=*,HOLD=YES  
./  ADD NAME=IKJS
//IKJACCTN PROC
//IKJACCT EXEC PGM=IKJEFTØ1,DYNAMNBR=5Ø,REGION=60ØØK,TIME=144Ø,  
//   PARM=ISPF
//STEPLIB DD DSN=ISP.SISPLOAD,DISP=SHR
```
Transfer some procedures from your current SYS1.PROCLIB

OMVS, VLF, etc, are useless for a mini system.

Create the ICHRDSNT table (dataset names table) for RACF

Create the ICHRDSNT table (dataset names table) for RACF
ICHRDSNT CSECT
DC AL1(1)     INDICATES ONE RACF DATASET
DC CL44'SYS1.RACFMINI' PRIMARY RACF DS NAME
DC CL44''     BACK-UP RACF DS NAME
DC AL1(255)   # RESIDENT INDEX AND DATA BLOCKS
DC X'81'     UPDATES DUPLICATED ON BACK-UP DS
END

//RACF12B  EXEC PGM=IEWL,PARM='XREF.LIST'
//SYSPRINT DD SYSOUT=* 
//SYSUT1 DD UNIT=V10,SPACE=(CYL,(1,1))
//SYSLIN DD DSN=*.RACF12A.SYSLIN,DISP=(OLD,DELETE)
// DD *
NAME ICHRDSNT(R)
//SYSLIB DD DSN=SYS1.LINKLIB,DISP=SHR,UNIT=SYSALLDA,VOL=SER=MINISY
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR,UNIT=SYSALLDA,VOL=SER=MINISY

Create and initialize the target RACF database
Create and initialize the target RACF database:

//RACF13  EXEC PGM=IRRMINØØ,PARM='NEW'
//SYSPRINT DD SYSOUT=* 
//SYSTEMP DD DSN=SYS1.MODGEN(IRRTEMP1),DISP=SHR
//SYSRACF DD DSN=SYS1.RACFMINI,DISP=(NEW,KEEP,DELETE)
// DD *
NAME ICHRDSNT(R)
//SYSLIB DD DSN=SYS1.LINKLIB,DISP=SHR,UNIT=SYSALLDA,VOL=SER=MINISY
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR,UNIT=SYSALLDA,VOL=SER=MINISY

Verify module IGC0001C CSECT IEAVTRML
Verify module IGC0001C CSECT IEAVTRML (memory termination table) in your new SYS1.LPALIB. Several products (eg RMF, IMS, NetView) ZAP it to indicate the name of their own routines. If these routines are not in SYS1.LPALIB but rather in LPALST libraries, you should ZAP it back to binary zeros to avoid an S806 abend at IPL time. Make sure you have no other case that would prevent the IPL from completing.

//ZAP14  EXEC PGM=AMASPZAP
//SYSPRINT DD SYSOUT=* 
//SYSLIB DD DISP=SHR,DSN=SYS1.LPALIB,UNIT=SYSALLDA,VOL=SER=MINISY
//SYSSIN DD *
NAME IGC0001C IEAVTRML
REP 0000 0000000,0000000,0000000,0000000,0000000,0000000

Create the TSO user-id IBMUSER

Creating the TSO user-id IBMUSER should be done with the following batch TSO job:

```plaintext
//UADS15   EXEC PGM=IKJEFT01
//SYSTSIN DD   *
ACCOUNT
  A (IBMUSER S A IKJS) SIZE(4000) JCL OPER NOMOUNT ACCT UNIT(SYSALLDA)
  L (IBMUSER)
END
//SYSTSPRT DD SYSSOUT=*
//SYSUADS DD DISP=SHR,DSN=SYS1.UADS,UNIT=SYSALLDA,VOL=SER=MINISY
//SYSLBC DD DISP=SHR,DSN=SYS1.BRODCAST,UNIT=SYSALLDA,VOL=SER=MINISY
```

Since IBMUSER has no TSO segment but is known to RACF, its initial password will be “SYS1”.

Initialize the new SYS1.LOGREC

Initializing the new SYS1.LOGREC should be done using IFCDIP00.

```plaintext
//LOGREC16 EXEC PGM=IFCDIP00
//SERERDS DD DSN=SYS1.LOGREC,UNIT=SYSALLDA,DISP=SHR,VOL=SER=MINISY
//FRAMES DD DDNAME=IEFRDER
```

Copy your current IODF

Copy your current IODF (we suppose here its name is SYS1.IODF04. It is downloaded to a sequential file and then REPROEd to SYS1.IODF00):

```plaintext
//IODF17   EXEC  PGM=IDCAMS
//SYSPRINT DD  SYSOUT=* 
//IODFOLD DD  DISP=SHR,DSN=SYS1.IODF04.CLUSTER 
//OUT DD  DSN=&OUT,DISP=(NEW,PASS),SPACE=(CYL,2,2) 
//    UNIT=SYSALLDA,DSORG=PS,LRECL=4096,RECFM=F 
//SYSIN DD *
  REPRO INFILE(IODFOLD)    OUTFILE(OUT)
/**  IODF18 : CREATE IODF00 FROM CURRENT IODF04 **/
/**  IODF18 EXEC PGM=IDCAMS **/
//STEP CAT DD  DISP=SHR,DSN=CATALOG.MINIMVS.MINISY 
//SYSPRINT DD  SYSOUT=* 
//OUT DD  DSN=&OUT,DISP=(OLD,DELETE)
```
If you intend to keep your mini system on tape, add the following:

```plaintext
//SYSIN DD *
DEL SYS1.IODF00.CLUSTER -
  CAT(CATALOG.MINIMVS.MINISY)
DEF CL(NAME(SYS1.IODF00.CLUSTER) LINEAR TRACKS(8 1) VOLUME(MINISY)) -
  DATA(NAME(SYS1.IODF00)) -
  CAT(CATALOG.MINIMVS.MINISY)
IF LASTCC = Ø THEN -
  REPRO ODS(SYS1.IODF00.CLUSTER) INFILE(OUT)
```

Locating members in concatenated PDSs

INTRODUCTION
Some time ago, I wrote a REXX EXEC called XLOCATE to scan a PDS concatenation for the existence of members. It also allowed for scanning the LINKLIST, LPA directory, and/or STEPLIB concatenation for the member specified. I recently needed to rework the EXEC when my shop converted to SWA=ABOVE for TSO in our JES2 parameters. The JFCB pointer needed by the EXEC became a token which requires the use of the SWAREQ macro to retrieve the...
actual virtual storage address. After reworking it, I thought it might be useful for others.

If no parameters are specified, then a brief syntax description will be displayed. The XLOCATE EXEC takes one and optionally two parameters. The first parameter is the member name to locate within the concatenation of PDSs. This parameter allows for generic specification, such as XX* to locate members whose names begin with XX, *XX to locate members whose names end in XX, or even *XX* to locate members whose names contain the string XX anywhere within the member name. This provides very flexible member name processing. If no second parameter is specified, then by default XLOCATE will search the STEPLIB concatenation followed by the LPA directory, and the system LINKLIST. If you wish to limit the search to a specific area, you can use STEP or STEPLIB to scan only the STEPLIB concatenation, LPA or LPALIST to scan only the LPA directory, or LINK or LINKLIST to scan the system LINKLIST concatenation. For the STEPLIB concatenation, the EXEC actually uses the JOBLIB pointer from the TCB, so even ISPF ISPLLIB as well as any other dynamic STEPLIB facility will be part of the search, as well as any actual STEPLIB DDname concatenation. If the second parameter is not one of those specified above, it is considered to be the DDname of a pre-allocated concatenation of PDSs to be searched for the specified member names.

XLOCATE uses two subroutines for some special processing. The first is another REXX EXEC called PDSDIR, which reads the PDS directory blocks to build a member name list for each of either the LINKLIST datasets, or the datasets pointed to by the user-specified DDname (parameter two).

This member name list is what is scanned to determine member name matches against the user specified specific or generic member name (parameter one). The second subroutine is a small Assembler program originally published in *MVS Update Issue 54* by Paul Lemmons back in 1991. The program, SWA2ADDR, uses the SWAREQ macro to convert token values in MVS control blocks to actual virtual storage addresses of SWA control blocks. I have made some minor modifications to the original code because of a subsequent change by IBM to the SWAREQ interface. The change necessitates the coding of the LOCEPAX=YES parameter on the IEFZB505 macro invocation when using the UNAUTH=YES parameter on the SWAREQ macro.
to generate an extended EPA parameter list. The extended EPA must also be cleared in its entirety before calling the SWAREQ service. I also took the liberty of adding a call to the IBM YREGS macro to perform register equates, which were missing from the original code, as well as some other minor changes. All changes are marked in the SWA2ADDR code with three asterisks (**`). The SWA2ADDR program must either reside in the LINKLIST, a STEPLIB or an ISPLLIB dataset.

**XLOCATE EXEC**

```rexx
/******************** REXX ********************/
arg PARM /* Retrieve input parms */
CVT = GETADDR(10) /* Addr CVT */
AOLD = GETADDR(224) /* Addr AOLD */
ASXB = GETADDR(AOLD 6C) /* Addr ASXB */
LTCB = GETADDR(ASXB 8) /* Addr LTCB */
TIOTP = GETADDR(LTCB C) /* Addr TIOT */
LPDIR = GETADDR(CVT 168) /* Addr LPADIR */
LPDIR = substr(LPDIR,2) /* Drop leading flag */
SCOPE = /* Clear search range */
CLEAR /* Clear Screen command? */
if words(PARM) > 1 then /* PARM > 1 word? */
do /* yes. */
  NAME = word(PARM,1) /* Extract search mem. */
  SCOPE = word(PARM,2) /* Extract search range */
end /* */
else NAME = PARM /* Else use mem name */
NAMELEN = length(NAME) /* Get name length */
if NAMELEN > 0 then /* Name exist? */
do /* Yes. */
  PFX = substr(NAME,1,1) /* Extract prefix */
  SFX = substr(NAME,NNAMELEN,1) /* Extract suffix */
end /* */
else do /* Prompt help message */
say 'XLOCATE can be used to locate member(s) in LPA. '
  ' or any allocated DD. The valid parameters are:'
say ''
say 'o no Parameter-this help message'
say ''
say 'o First Parameter-'
say '  member/module name to be searched for in the form of:'
say ''  . Prefix =========> XXXXX'
say ''  . Suffix =========> *XXXXX'
say ''  . Occurrence =========> *XXXXX'
say ''  . Exact name =========> XXXXXX'
say ''
say 'o Second Parameter-'
say '  scope of the search in the form of:'
say ''
```

select
when PFX = '*' & SFX = '*' then
do
   NAMELEN = NAMELEN - 2
   NAME = substr(NAME,2,NAMELEN)
end
when PFX = '*' then
do
   NAMELEN = NAMELEN - 1
   NAME = substr(NAME,2)
end
when SFX = '*' then
do
   NAMELEN = NAMELEN - 1
   NAME = substr(NAME,1,NAMELEN)
end
otherwise
   nop
end
PROC = Ø                                   /* processed special scope*/
if SCOPE = '' | SCOPE = 'STEP' | SCOPE = 'STEPLIB' then
   call STEPLIST
if SCOPE = '' | SCOPE = 'LPA' | SCOPE = 'LPALIST' then
   call LPALIST
if SCOPE = '' | SCOPE = 'LINK' | SCOPE = 'LINKLIST' then
   call Linklist
if SCOPE
   ¬= '' & PROC ¬= 1 then
   call USERLIST
exit
/**********************************************************************/
/*   STEPLIB libraries search routine                                 */
/**********************************************************************/
STEPLIST:
PROC = 1
say 'Now listing' word(PARM,1) 'modules in STEPLIB(s)' time()
JLB = C2X(GETDATA(LTDB 28 4))
if JLB = 'ØØØØØØØØ' then
   return Ø
DCB = C2X(GETDATA(JLB 28 2))
TIOT = D2X(X2D(TIOTP) + X2D(DCB))
TIOTP = D2X(X2D(TIOTP) + X2D(DCB) - X2D(18))
TEMP = SCOPE
SCOPE = GETDATA(TIOT 4 8)
FLG = 1
call USERLIST
SCOPE = TEMP
TIOTP = GETADDR(LTCB C)
return Ø
/***********************************************************************/
/*   LPA directory search routine                                     */
/***********************************************************************/
LPALIST:
PROC = 1
say ""
say 'Now listing' word(PARM,1) 'modules in LPA directory' time()
do forever
  ENAME = GETDATA(LPDIR 8 8)
  if substr(ENAME,1,1) = 'FF'x then
    leave
  EADDR = GETADDR(LPDIR 10)
  EADDR = '0' || substr(EADDR,2)
  MADDR = GETADDR(LPDIR 14)
  if MADDR ≠ '00000000' then
    MNAME = GETDATA(MADDR 8 8)
  else MNAME = '        '  
call COMPARE
  if FLG = Ø then
    do
      LPDIR = d2x(x2d(LPDIR) + x2d(28))
      iterate
    end
  if MNAME ≠ '        ' then
    MNAME = 'as alias of' MNAME
    say right(ENAME,10) 'found at' EADDR MNAME
    LPDIR = d2x(x2d(LPDIR) + x2d(28))
  end
end
return Ø
/***********************************************************************/
/*   LINKLIST libraries search routine                                */
/***********************************************************************/
LINKLIST:
PROC = 1
say ""
say 'Now listing' word(PARM,1) 'modules in LINKLIST' time()
LLT = GETADDR(CVT 4DC)
LLTNUM = GETADDR(LLT 4)
LLTNUM = X2D(LLTNUM)
CNTR = Ø
OFFSET = Ø
DSNLIST =
do I = 1 to LLTNUM
  DSNAME  = GETDATA(LLT 9 OFFSET 44)
  DSNAME = strip(DSNAME)
  CNTR = CNTR + 1
  OFFSET = D2X(X2D(2D)*CNTR)
  DSNLIST = DSNLIST "" || DSNAME || ""
end
call MEMSCAN
return Ø

/**********************************************************************/
/*   USER DD name search routine                                     */
/**********************************************************************/
USERLIST:
PROC = 1
if FLG ≠ 1 then
do
  say ''
say 'Now listing' word(PARM,1) 'members in' SCOPE time()
end
TIOT = D2X(X2D(TIOTP) + X2D(18))
TIOTL = C2X(GETDATA(TIOT 1))
WORKDDN =
DSNLIST =
do WHILE TIOTL L ≠ 'ØØ'
  DDNAME = GETDATA(TIOT 4 8)
  TIOTL = C2X(GETDATA(TIOT 1))
  TIOTF = GETDATA(TIOT 1 1)
  if WORKDDN = '' then
    if DDNAME = SCOPE then do
      WORKDDN = DDNAME
    end
  else do
    TIOT = D2X(X2D(TIOT) + X2D(TIOTL))
    iterate
  end
else if DDNAME ≠ '' then
  do
call MEMSCAN
  return Ø
  end
if TIOTF = 'Ø1'x then
  do
    JFCBTOK = GETDATA(TIOT C 3)
    AREA1 = JFCBTOK || 'ØØ'x
    JFCBADR = 'ØØØØØØØØ'x
    address LINKPGM 'SWA2ADDR AREA1 JFCBADR'
    JFCB = C2X(JFCBADR)
    DSN = GETDATA(JFCB 44)
    if DDNAME = SCOPE | DDNAME = '' then
      DSNLIST = DSNLIST "" || strip(DSN) || ""
    end
    TIOT = D2X(X2D(TIOT) + X2D(TIOTL))
  end
return Ø

/**********************************************************************/
/*   Member search routine (prefix/suffix/occurrence search)          */
/**********************************************************************/
MEMSCAN:
do J = 1 to words(DSNLIST)
  DSNDISP = Ø

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
MEMLIST =
LINECNT = Ø
DSNAME = word(DSNLIST,J)
MBRNAMES = PDSDIR(DSNAME)
do K = 1 to words(MBRNAMES)
   ENAME = word(MBRNAMES,K)
call COMPARE
   if FLG = Ø then
      iterate
   if DSNDISP = Ø then
      do
         DSNDISP = 1
         say DSNAME
      end
   MEMLIST = MEMLIST left(ENAME,8)
   LINECNT = LINECNT + 1
   if LINECNT = 8 then
      do
         say ' ' || MEMLIST
         LINECNT = Ø
         MEMLIST = ''
      end
   end
   if LINECNT > Ø then
      say ' ' || MEMLIST
end
return Ø
/**********************************************************************/
/*         Member prefix/suffix/occurrence compare                      */
/**********************************************************************/
COMPARE:
FLG = Ø
select
   when PFX = '*' & SFX = '*' then
      if index(ENAME,NAME) > Ø then
         FLG = 1
   when PFX = '*' then
      do
         ADJ_NAMELEN = length(strip(ENAME)) - NAMELEN + 1
         if ADJ_NAMELEN > Ø then
            if NAME = substr(ENAME,ADJ_NAMELEN) then
               FLG = 1
         end
      end
   when SFX = '*' then
      if index(ENAME,NAME) = 1 then
         FLG = 1
   when NAMELEN > Ø then
      if NAME = ENAME then
         FLG = 1
   otherwise
      FLG = 1
end
return Ø
GETADDR:
arg ADDR
GETADDR_A = word(ADDR,1)
if words(ADDR) > 1 then
do GETADDR_I = 2 to words(ADDR)
   GETADDR_B = word(ADDR,GETADDR_I)
   GETADDR_A = D2X(X2D(GETADDR_A) + X2D(GETADDR_B))
end
ANSWER = C2X(Storage(GETADDR_A,4))
return ANSWER

GETDATA:
arg ADDR
GETDATA_A = word(ADDR,1)
GETDATA_C = word(ADDR,words(ADDR))
if words(ADDR) > 2 then
do GETDATA_I = 2 to words(ADDR)-1
   GETDATA_B = word(ADDR,GETDATA_I)
   GETDATA_A = D2X(X2D(GETDATA_A) + X2D(GETDATA_B))
end
ANSWER = Storage(GETDATA_A,GETDATA_C)
return ANSWER

PDSDIR EXEC
/* REXX **************************************************************/
/* Allocate, read PDS diretory and build string of member names */
/*********************************************************************/
arg DSN /* GET DATASET NAME */
address TSO
"ALLOC DD(PDS) DA("DSN") SHR REUSE", /* ALLOCATE PDS DIRECTORY */
" RECFM(F) DSORG(PS) LRECL(256) BLKSIZE(256)"
"EXECIO * DISKR PDS (STEM DIR. FINIS" /* READ DIRECTORY BLOCKS */
"FREE DD(PDS)" /* FREE FILE */
PDSDNAME = '' /* INITIALIZE NAME STRING */
do BLK = 1 to DIR.Ø /* SCAN DIRECTORY BLOCKS */
   USEDBYTES = c2d(substr(DIR.BLK,1,2)) /* GET DIRECTORY BLOCK LEN. */
   INDEX = 3 /* SKIP PAST USED BYTES */
do while INDEX < USEDBYTES
   if substr(DIR.BLK,INDEX,8) = 'FFFFFFFFFFFFF'X THEN
      signal DIREOF /* IF LOGICAL EOF FOUND */
      PDSDNAME = PDSDNAME strip(substr(DIR.BLK,INDEX,8)) /*CONCAT NAME*/
      INDEX = INDEX + 11 /* SKIP PAST NAME AND TTR */
      PDSD2INDC = substr(DIR.BLK,INDEX,1) /* GET PDSD2INDC BYTE */
      LEN = bitand(PDSD2INDC,'1F'X) /* ISOLATE USER DATA LENGTH */
      USRDATA = c2d(LEN) * 2 /* HALFWORDS TO BYTES */
      INDEX = INDEX + USRDATA + 1 /* SKIP PAST USER DATA */
   end
DIREOF:                                  /* LOGICAL EOF PROCESSING   */
PDS2NAME = strip(PDS2NAME,'L')           /* STRIP LEADING BLANKS     */
return PDS2NAME                          /* RETURN BLANK DELIM NAMES */

SWA2ADDR
* NAME : SWA2ADDR
* FUNCTION : THIS SUBROUTINE CONVERTS SWA TOKENS TO REAL Addresses.  
* FEATURES : THIS PROGRAM WILL WORK ON ALL MVS/XA SYSTEMS, RELEASE   
* 2.2 OR LATER. IT DOES NOT MATTER WHETHER OR NOT THE SWA *  
* CONTROL BLOCKS RESIDE ABOVE OR BELOW THE 16 MEG LINE.  
* IN EITHER CASE IT WILL STILL RETURN A VALID ADDRESS.  
* CALL FMT : CALL SWA2ADDR(TOKEN,ADDRESS) 
* PARAMETERS: THE TWO PARAMETERS PASSED TO THIS SUBROUTINE ARE DEFINED*  
* AS FOLLOWS.                                                 
* TOKEN      DS   XL3      THE TOKEN FOR A SWA CONTROL BLOCK SUPPLIED *  
* BY THE CALLER.                                             
* ADDRESS    DS   A        A 31 BIT ADDRESS TO BE RETURNED TO THE *  
* CALLER.                                                  
* REG-15 WILL CONTAIN THE RETURN CODE FROM THE SWAREQ MACRO. 
SWA2ADDR CSECT
SWA2ADDR AMODE 31
SWA2ADDR RMODE ANY
YREGS ***  
   BAKR R14,0          ESA STYLE SAVE 
   BASR R12,0          ADDRESS THIS CSECT 
USING *,R12  
   LM R3,R4,0(R1)      R3->TOKEN, R4->ADDRESS 
   LA R1,SWA_EPA       ADDRESS THE ENTRY PARM 
USING ZB505, R10  
   XC SWAEPAX,SWAEPAX *** CLEAN THE AREA 
   MVC SWVA(3),0(R3)    PLACE SWA TOKEN IN PLIST 
SWAREQ FCODE=RL,    READ/LOCATE REQUEST -  
   EPA=SWEPAPTR,       ENTRY PAREMETER LIST -  
   MF=(E,SWAPARMS),    LIST FORM ENTRY -  
   UNAUTH=YES          PROGRAM EXECUTES UNAUTHORIZED 
   L R1,SWBLKPTR       GET ADDRESS OF SCHED CB  
   ST R1,0(R4)         STORE SWA CONTROL BLOCK ADDRESS 
PR EXIT  
LTORG  
SWEPAPTR DC A(SWA_EPA)     ADDRESS OF ENTRY PARM LIST  
SWA_EPA DS XL(ZB505LN) *** PARM LIST MAPED BY IEFZB505 
SWAPARMS SWAREQ MF=L,    LIST FORM OF THE REQUEST -  
   UNAUTH=YES  
   IEFZB505 LOCEPAX=YES *** MAP THE PARM LIST 
ZB505LN EQU -*ZB505 ***  
   IEFJESCT        JES CONTROL TABLE  
   CVT DSECT=YES   GOOD OLD CVT  
END

© Xephon 1999
LISTING ICF Catalog Entries

Introduction

The CATLST program lists entries from the ICF master/user catalog(s) – it uses the Catalog Search Interface (CSI) to obtain information for each or a specified master/user catalog(s). Output may be limited by specifying datasetname, catalogname or volume.

Because the CATLST program uses the new API for catalog requests, it runs much faster than IDCAMS. For example, to list all ICF catalog entries in our environment the program runs for one minute to list 30 catalogs with 340,000 entries. It may be very useful in the following situations:

- In case of a DASD error you need information about which datasets are on the failing DASD volume – if the VTOC option is also corrupted, it is very time consuming to obtain all necessary information from the catalogs using IDCAMS. With this program you can search for all catalogued datasets on a specific volume and print all entries that point to that failing volume.

- Also it may be useful to list all ML2 datasets or to check if there are any datasets on the SYSRES volume that are not catalogued with VOL(******). Furthermore it can be used to check for multi-volume datasets.

The program produces one line for each catalog entry and one additional line for each volume of a multi-volume dataset. It may be useful in various aspects of storage/catalog management – for example, to find out duplicate catalog entries or compare catalog entries with VTOC entries. An example of the output is shown below:

<table>
<thead>
<tr>
<th>ALIAS</th>
<th>DB2</th>
<th>MCAT.SYSCAT.MASTER</th>
<th>SYS1.TESTPLEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIAS</td>
<td>SAMPLE</td>
<td>MCAT.SYSCAT.MASTER</td>
<td>SYS1.TESTPLEX</td>
</tr>
<tr>
<td>NONVSAM</td>
<td>SYS1.LPALIB</td>
<td>*****</td>
<td>MCAT.SYSCAT.MASTER</td>
</tr>
<tr>
<td>NONVSAM</td>
<td>SYS1.MACLIB</td>
<td>*****</td>
<td>MCAT.SYSCAT.MASTER</td>
</tr>
<tr>
<td>CLUSTER</td>
<td>SYS1.MAN1</td>
<td>MCAT.SYSCAT.MASTER</td>
<td>SYS1.TESTPLEX</td>
</tr>
<tr>
<td>DATA</td>
<td>SYS1.MAN1.DATA</td>
<td>VOL002</td>
<td>MCAT.SYSCAT.MASTER</td>
</tr>
<tr>
<td>NONVSAM</td>
<td>SAMPLE.DATASET1</td>
<td>VOL002</td>
<td>UCAT.SYSCAT.USER</td>
</tr>
<tr>
<td>NONVSAM</td>
<td>SAMPLE.DATASET2</td>
<td>VOL001+</td>
<td>UCAT.SYSCAT.USER</td>
</tr>
<tr>
<td>NONVSAM</td>
<td>SAMPLE.DATASET2</td>
<td>VOL004</td>
<td>UCAT.SYSCAT.USER</td>
</tr>
</tbody>
</table>
Note: in case of a multi-volume dataset there is one line for each volume

The following JCL is required to run the program:

```
//CATLST   JOB (),....
//** LIST CATALOG ENTRIES
//**
//S1 EXEC PGM=CATLST
//** EXEC PGM=CATLST,PARM='DSN=SYS*.**'
//** EXEC PGM=CATLST,PARM='DSN=SYS1.**'
//** EXEC PGM=CATLST,PARM='DSN=SYS1.*.LOAD'
//** EXEC PGM=CATLST,PARM='CAT=MCAT.SYSCAT.VSYSCAT'
//** EXEC PGM=CATLST,PARM='CAT=UCAT.SYSCAT.VSYSCAT'
//** EXEC PGM=CATLST,PARM='CAT=UCAT.SYSCAT.TAPE'
//** EXEC PGM=CATLST,PARM='VOL=SYSRES'
//** EXEC PGM=CATLST,PARM='VOL=ARCVOL'
//** EXEC PGM=CATLST,PARM='VOL=******'
//** EXEC PGM=CATLST,PARM='VOL=SYSCAT/UCAT.SYSCAT.VSYSCAT'
//STEPLIB  DD  DSN=your.loadlib,DISP=SHR
//SYSUDUMP DD  SYSOUT=
//SYSOUT   DD  DSN=CATLST.OUTPUT, DISP=(NEW,CATLG),UNIT=DISC,SPACE=(CYL,(10,10))
```

The following REXX can be used to run CATLST under TSO:

```
/**************************** REXX ****************************/
/* CATLST - LIST MASTER/USER CATALOG(S) */
/*****************************/
trace off
parse upper arg parm
if parm = .', then do
  say ,CATLST: one of the following parms','
  ,must be supplied under TSO'
  say ,CATLST: DSN=dsname or CAT=catname or ','
  ,VOL=volser or VOL=volser/catname'
```

OPERATIONAL ENVIRONMENT

The program was developed in an OS/390 Version 2 Release 4 environment and tested under OS/390 Version 2 Release 4 and Version 2 Release 5. Note that, because the program uses the CSI (Catalog Search Interface), a new function in DFSMS 1.4, it requires at a minimum level OS/390 Version 2 Release 4. CSI is an MVS read-only general-use programming interface that is used to obtain information about entries contained in ICF catalogs. A description of the CSI can be found in DFSMS/MVS V1R4 Managing Catalogs Appendix D. Catalog Search Interface User Guide.
REQUIREMENTS: OS/390 V2.4 (DFSMS V1.4) OR A LATER VER./REL. 
IS NECESSARY IN ORDER TO RUN THIS PROGRAM

PARAMETER:
'DSN=DATASETNAME'
'CAT=CATALOGNAME'
'VOL=VOLSER'
'VOL=VOLSER/CATALOGNAME'

EXAMPLE(S):
'DSN=SYS1.**' LIST ALL ENTRIES WITH HLQ SYS1
'DSN=SYS*.**' LIST ALL ENTRIES BEGINNING WITH SYS
'DSN=SYS1.*.LOAD' LIST ALL ENTRIES WITH HLQ SYS1 AN LLQ LOAD
'CAT=UCAT.USRCAT01' LIST ALL CATALOG ENTRIES
'VOL=SYSRES' LIST ALL ENTRIES FOR VOLSER
'VOL=SYSRES/MCAT.SYS01' LIST ALL CATALOG ENTRIES WITH VOLSER

ENVIRONMENT:
AUTHORIZATION - NON REQUIRED
ATTRIBUTES - NONREENTERBL
STATE KEY - PROBLEM STATE
RUNNING MODE - AMODE(31), RMODE(24)

RETURN CODE(S): REG.15
0 = OK
4 = NO ENTRIES LISTED (NO OUTPUT)
8 = INVALID PARAMETER
12 = PROCESSING ERROR

REGISTER AT ENTRY:
GPR 1 = PARAMETER ADDR.
GPR 13 = ADDR.SAVE-AREA
GPR 14 = RETURNADDRESS
GPR 15 = ENTRY POINT ADDR.

REGISTER USAGE:
GPR 0 =
GPR 1 = PARAMETER ADDR.
GPR 2 = WORK
GPR 3 = WORK
GPR 4 = WORK
GPR 5 = WORK / CSI RETURN AREA DSECT
GPR 6 = WORK - END OF CSI RETURN AREA
GPR 7 = WORK - ENTRY DSECT IN CSI RETURN AREA
GPR 8 = WORK
GPR 9 = WORK - UCAT TABLE
GPR 10 = RETURN ADDR.FOR SUBROUTINES - NOT USED
GPR 11 = SEC. BASE REGISTER
GPR 12 = FIRST BASE REGISTER
GPR 13 = ADDR.SAVE-AREA
GPR 14 = ACTUAL RETURN ADDR.
GPR 15 = BRANCH REGISTER

EJECT

*        MODIFIED:
*        DD.MM.JJJJ          XXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXX
*
EJECT

CATLST   AMODE 31
CATLST   RMODE 24
CATLST   CSECT

STM   R14,R12,12(R13)     SAVE REG.CONTENTS IN HIGHER SA
LR    R12,R15             LOAD BASE
LA    R11,4095(R12)
LA    R11,1(R11)

USING CATLST,R12,R11      ESTABLISH ADDRESSABILITY
ST    R13,SA+4            BACKWARDCHAINING
LR    R14,R13             ADDR.HIGHER SAVE AREA
LA    R13,SA              POINTER TO NEW SAVE AREA
ST    R13,8(R14)          FORWARDCHAINING
B     BEGIN
SPACE 3
DC    CL8'CATLST'
DC    C'
DC    C'&SYSDATE'
SA    DC 18F'0'
EJECT
BEGIN   EQU   *

*—— DATASETNAME SUPPLIED ?

*        PROCESS PARAMETER
*

L     R1,0(R1)         GET PARAMETER ADDR.
LH    R2,0(R1)        PARM LEN
LTR   R2,R2            ANY PARM?
BZ    NOPARM         ...NO > NO PARM
STH   R2,PARMLEN     SAVE PARM LEN
LA    R3,2(R1)       PARM DATA ADDR.
ST    R3,PARMADDR    SAVE PARM DATA ADDR.
DSNPARM  CLC  Ø(4,R3),=C'DSN='  DSN PARM?
BNE  CATPARM
SH  R2,=H'4'
BNE  DSN PARM?
CH  R2,=H'44'
BH  ERRORPRM
BCTR  R2,0
EX  R2,*+4
MVC  DSNFILT(Ø),4(R3)
B  ENDPARM

*—— CATALOG SUPPLIED ?
CATPARM  CLC  Ø(4,R3),=C'CAT='  CATALOG PARM?
BNE  VOLPARM
SH  R2,=H'4'
BNE  LEN OF CATNAME >= 1?
CH  R2,=H'44'
BH  LEN OF CATNAME > 44?
BH  ERRORPRM
BCTR  R2,0
EX  R2,*+4
MVC  CATFILT(Ø),4(R3)
MVC  UCATTAB,CATFILT
MVI  UCATTAB+UCATTABL,X'FF'
NI  ONLY ONE UCAT ENTRY
OI  CATALOG SUPPLIED ?
B  ENDPARM

*—— VOLSER SUPPLIED ?
VOLPARM  CLC  Ø(4,R3),=C'VOL='  VOLSER PARM?
BNE  ERRORPRM
SH  R2,=H'4'
BNE  LEN OF VOLSER >= 1?
CH  R2,=H'6'
BH  LEN OF VOLSER < 6?
BH  ERRORPRM
MVC  VOLFILT,4(R3)
MVC  UCATTAB+UCATTABL,X'FF'
NI  ONLY ONE UCAT ENTRY
OI  VOLSER + CATALOG PARM?
B  ENDPARM

*—— + CATALOG SUPPLIED ?
CH  R2,=H'6'
BNE  ENDPARM
CLI  1Ø(R3),C'/'
BNE  ERRORPRM
SH  R2,=H'7'
BNE  LEN OF CATNAME >= 1?
CH  R2,=H'44'
BH  LEN OF CATNAME > 44?
BH  ERRORPRM
BCTR  R2,0
EX  R2,*+4
MVC  CATFILT(Ø),11(R3)
MVC  UCATTAB,CATFILT
MVI  UCATTAB+44,X'FF'
NI  ONLY ONE UCAT ENTRY
OI  VOLSER + CATALOG PARM?
B  ENDPARM

NOPARM  DS  ØH
ENDPARM  DS  ØH

EJECT

* Observe System Info and Master Catalog Name/Volser

L     R2,CVTPTR
USING CVTMAP,R2
CLC   CVTCVT,=CL4' CVT'
BNE   ERRORCVT
      ... NO CVT > ERROR
L     R3,CVTECVT
USING ECVT,R3
CLC   ECVTECVT,=CL4'ECVT'
BNE   ERRORECV
      ... NO ECVT > ERROR
L     R4,ECVTIPA
USING IPA,R4
CLC   IPAID,=CL4'IPA '
BNE   ERRORIPA
      ... NO IPA > ERROR
L     R5,CVTSMCA
USING SMCABASE,R5
CLC   SMCASMCA,=CL4'SMCA'
BNE   ERRORSMC
      ... NO SMCA > ERROR

MVC   SYSID,SMCASID
      SMF/SYSTEM ID
MVC   LPARNAME,IPALPNAM
      LPAR NAME
MVC   PLEXNAME,IPASXNAM
      SYSPLEX NAME
MVC   MCATNAME,IPASCDSN
      MASTER CATALOG NAME
MVC   MCATVOL,IPASCVOL
      MASTER CATALOG VOLSER
EJECT

* Obtain All UCATs

CATSW1 NOP   SKIPCAT
      *** SWITCH *** > SKIP UCATS?

*—— Get CSI Return Area
GETMAIN RC,LV=32767
LTR   R15,R15
BNZ   ERRORGM
ST    R1,PARMRWK
      SAVE ADDR. OF RETURN AREA
MVC   Ø(4,R1),=F'32767'
      STORE LENGTH

*—— Set Up Catalog Search Interface
MVI   CSIFIELD,C' '          CLEAR SELECTION FIELDS
MVC   CSIFIELD+1(CSIFIELL-1),CSIFIELD
MVC   CSIFILTK(2),=C'***'    GET ALL ENTRIES
MVC   CSICATNM,MCATNAME      SET CATALOG = MCAT
MVI   CSIDTYPs,C'U'          SET TYPE = UCAT
MVI   CSICLDI,C'Y'
MVI   CSISICAT,C'Y'
MVI   CSIRESRV,X'ØØ'
      RESERVED
MVC   CSINUMEN,=H'Ø'
      NO ADDITIONAL FIELDS

*—— Invoke Catalog Search Interface
LA    1,PARMLIST
CALL  IGGCSIØØ
      >>> CATALOG SEARCH INTERFACE
LTR   R15,R15
      TEST RETURN CODE
BZ    ++36
      ... PGM/ENV/SYSTEM ERROR

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
DC X'0000' ABEND S0C1
DC CL30'*** ABEND - IGGCSIØØ ERROR ***'

*—— SAVE MCAT
LA  R9,UCATTAB ADDR. OF UCAT TABLE
MVC Ø(44,R9),MCATNAME MOVE MCAT NAME TO UCAT TAB
LA  R9,44(R9) = NEXT UCAT TAB ENTRY

*—— PROCESS RETURNED DATA
L  R5,PARMRWK ADDR. OF RETURNED DATA
USING CSIRWORK,R5
LR  R6,R5 ADDR. OF RETURNED DATA
A  R6,CSIUSDLN + LEN = END ADDR.
LA  R7,CSIRWORL(R5) ADDR. OF FIRST ENTRY
USING CSIRWENT,R7

NEXTUCAT DS ØH
MVC Ø(44,R9),CSIENAME MOVE ENTRY NAME TO UCAT TAB
LA  R9,44(R9) = NEXT UCAT TAB ENTRY
LA  R7,CSIRWENL(R7) = NEXT RETURNED UCAT ENTRY
CR  R7,R6 END OF WORK AREA?
BL  NEXTUCAT
MVI  Ø(R9),X'FF' SET END OF UCAT TAB

*—— FREE CSI RETURN AREA
L  R2,PARMRWK
FREEMAIN RC,LV=32767,A=(2)

*—— OPEN OUTPUT DATASET

*—— GET CSI RETURN AREA
GETMAIN RC,LV=65535
LTR  R15,R15
BNZ  ERRORGM
ST  R1,PARMRWK SAVE ADDR. OF RETURN AREA
MVC  Ø(4,R1),=F'65535' STORE LENGTH
*—— SET UP CATALOG SEARCH INTERFACE
MVI CSIFIELD,C' ' CLEAR SELECTION FIELDS
MVC CSIFIELD+1(CSIFIELL-1),CSIFIELD
MVC CSIFILTK,DSNFILT SET FILTER
MVC CSICATNM,UCATNAME SET CATALOG = MCST/UCAT
MVI CSICLDI,C'Y'
MVI CSISCAT,C'Y'
MVI CSIRESV,X'ØØ'
MVC CSINUMEN,=H'1'
MVC CSIFLDNM,=CL8'VOLSER' RETURN VOLUME INFO
*—— INVOKE CATALOG SEARCH INTERFACE
LOOPCSI DS ØH
LA 1,PARMLIST
CALL IGGCSIØØ >>> CATALOG SEARCH INTERFACE
LTR R15,R15 TEST RETURN CODE
BZ *+36 ... PGM/ENV/SYSTEM ERROR
DC X'ØØØØ' ABEND SØC1
DC CL3Ø'*** ABEND - IGGCSIØØ ERROR ***'
*—— PROCESS RETURNED DATA FROM CSI
L R5,PARMRWK ADDRESS OF RETURNED DATA USING CSIRWORK,R5
TM CSICFLG,B'Ø1ØØØØØØ' NO ENTRY FOUND FOR THIS CAT?
BO NOENTRY
LR R6,R5 ADDRESS OF RETURNED DATA + LEN = END ADDR.
A R6,CSIUSDLN ADDR. OF FIRST ENTRY USING CSIRWENT,R7
LOOPENTR DS ØH
MVC OCNAME,CSICATNM MOVE CATALOG NAME
MVC OSYSID,SYSID MOVE SYSTEM-/SMF-ID
MVC OPLEXNM,PLEXNAME MOVE SYSPLEX NAME
MVC OENAME,CSIENAME MOVE ENTRY NAME
CLI CSIENAME,X'ØØ' CATALOG SELF DESCR. ENTRY?
BNE *+10
MVC OENAME,CSICATNM MOVE CATALOG NAME
CLI CSIETYPE,C'A' NONVSAM?
BNE *+14
MVC OETYPE,=CL8'NONVSAM'
B VOLFLD
CLI CSIETYPE,C'B' GDG?
BNE *+14
MVC OETYPE,=CL8'GDG'
B NOVOLSER
CLI CSIETYPE,C'C' CLUSTER?
BNE *+14
MVC OETYPE,=CL8'CLUSTER'
B NOVOLSER
CLI CSIETYPE,C'D' DATA?
BNE *+14
MVC OETYPE,=CL8'DATA'
B VOLFLD
CLI CSIETYPE,C'G' AIX?
*—— PROCESS CATALOG ENTRIES WITH VOLUME(S)

VOLFLD
LA   R8,CSRWRNL(R7)    ADDR. OF FIRST FIELD
LH   R2,Ø(R8)         LOAD LENGTH OF VOLSER FIELD(S)
CH   R2,=H'6'        ONLY 1 VOLSER ?
BNH  ++8
MVI  OEMVOL,C'+'     SET MULTI VOLUME INDICATOR
LA   R8,2(R8)        SKIP LENGTH
LOOPVOL DS  ØH
VOLSW1 B  ++14        *** SWITCH *** > SKIP VOLSER?
CLC  VOLFILT,Ø(R8)  ... ONLY SUPPLIED VOLSER
BNE  SKIPVOL
MVC  OEVOL,Ø(R8)    MOVE VOLSER
PUT  SYSOUT,OREC
AP   CNTOUT,=P'1'   INCREASE OUTPUT COUNTER
SKIPVOL MVI  OEMVOL,C' ' , CLEAR VOLSER
MVI  OEMVOL,C' ' , CLEAR MULTI VOLUME INDICATOR
LA   R8,6(R8)       NEXT VOLSER
SH   R2,=H'6'
CH   R2,=H'6'       MORE VOLUMES?
BNL  LOOPVOL
B    NEXTENTR

*—— PROCESS CATALOG ENTRIES WITHOUT VOLUMES

NOVOLSER DS  ØH
VOLSW2 NOP SKIPENT        *** SWITCH *** > SKIP ENTRY?
PUT  SYSOUT,OREC
AP   CNTOUT,=P'1'   INCREASE OUTPUT COUNTER
SKIPENT MVI  OREC,C' ' , CLEAR OUTPUT RECORD
MVC OREC+1(ORECLEN-1),OREC
*—— POINT TO NEXT ENTRY
NEXTENTR LA R7,CSIRWENL(R7) ADDR. OF FIRST FIELD
AH R7,0(R7) + LEN OF FIELD(S)
LA R7,2(R7) + LEN FIELD = NEXT ENTRY
CR R7,R6 END OF WORK AREA?
BL LOOPENTR
*—— MORE ENTRIES TO PROCESS ?
CLI CSIRESUM,C'Y'
BE LOOPCSI
*—— FREE CSI RETURN AREA
NOENTRY L R2,PARMRWK
FREEMAIN RC,LV=65535,A=(2)

*——————————————————————————————————-
LA R9,UCATTABL(R9) NEXT UCAT TO PROCESS
B LOOPCAT
EJECT
* ———————————————————————————————————
* END-OF-PROGRAM
*——————————————————————————————————
EOF DS ØH
CLOSE SYSSOUT
CP CNTOUT,=PL1'Ø' ... NO OUTPUT?
BE WARNING
EOP DS ØH
L R13,4(R13) ADDR.HIGHER SAVE AREA
LM R14,R12,12(R13) RESTORE REG.CONTENTS
XR R15,R15 RETURNCODE = Ø
BR R14 RETURN TO CALLER
SPACE 3
*—— WARNING / ERROR(S)
*——————————————————————————————————
WARNING DS ØH
L R13,4(R13) ADDR.HIGHER SAVE AREA
LM R14,R12,12(R13) RESTORE REG.CONTENTS
LA R15,4 RETURNCODE = 4
BR R14 RETURN TO CALLER
* ERRORPRM WTO ',CATLST: PARAMETER ERROR', ROUTCDE=(11)
L R15,=F'8' RETURN CODE = 8
B ERROR
ERRORCVT WTO ',CATLST: CVT NOT FOUND', ROUTCDE=(11)
L R15,=F'12' RETURN CODE = 12
B ERROR
ERRORECV WTO ',CATLST: ECVT NOT FOUND', ROUTCDE=(11)
L R15,=F'12' RETURN CODE = 12
B ERROR
ERRORIPA WTO ',CATLST: IPA NOT FOUND', +
ROUTCDE=(11)
L R15,=F'12' RETURN CODE = 12
B ERROR
ERRORSMC WTO ,CATLST: SMCA NOT FOUND', +
ROUTCDE=(11)
L R15,=F'12' RETURN CODE = 12
B ERROR
ERRORGM WTO ,CATLST: GETMAIN ERROR', +
ROUTCDE=(11)
L R15,=F'12' RETURN CODE = 12
B ERROR
ERROROUT WTO ,CATLST: SYSOUT OPEN ERROR', +
ROUTCDE=(11)
L R15,=F'12' RETURN CODE = 12
B ERROR
ERROR DS ØH
L R13,4(R13) LOAD ADDRESS HIGHER SAVE AREA
L R14,12(R13) RESTORE RETURN ADDR.
LM RØ,R12,2Ø(R13) RESTORE REGISTER CONTENTS
BR R14
SPACE 3
*
ABEND DS ØH
DC X'0000' ABEND SØC1
DC C'*** ABEND ***'
EJECT
*
* SUBROUTINE(S)
*
SPACE 3
EJECT
*
* READ ONLY STORAGE
*
EJECT
*
* PRIVATE WORK AREAS
*
SPACE 2
PARMLEN DS H PARAMETER LENGTH
PARMADDR DS F PARAMETER ADDRESS
SYSID DS CL4 SMF/SYSTEM ID
LPARNAME DS CL8 LPAR NAME
PLEXNAME DS CL8 SYSPLEX NAME
MCATNAME DS CL44 MASTER CATALOG NAME
MCATVOL DS CL6 MASTER CATALOG VOLSER
UCATNAME DS CL44 USER CATALOG NAME
DSNFILT DC CL44'**' DATASET FILTER FOR CSI
CATFILT DC CL44' ' CATALOG FILTER FOR CSI

VOLFILT DC CL6'???????' VOLUME FILTER
CNTOUT DC PL5'Ø' OUTPUT LINE COUNTER

* UCAT TABLE *

UCATTAB DS 100CL44 RESERVE SPACE FOR 100 UCATS
UCATTABL EQU 44

* PARAMETER LIST FOR IGGCSI00 INVOCATION *

PARMLIST DS ØD
PARMMRR DC A(CSIMRR) MODULE/REASON/RETURN
PARMSCF DC A(CSIFIELD) SELECTION CRITERIA FIELDS
PARMRWK DC A(Ø) RETURNED WORK AREA

* SELECTION CRITERIA FIELDS FOR IGGCSI00 INVOCATION *

*** SEE DFSMS MANAGING CATALOGS APPENDIX D ***

******** IGGCSINA MAPPING MACRO
CSIFIELD DS ØF
CSIFILTK DC CL44'**' GENERIC FILTER KEY
CSICATNM DC CL44'??????????' CATALOG NAME OR BLANKS
CSIRESNM DC CL44' ' RESUME NAME OR BLANKS
CSIDTYPD DS ØCL16 ENTRY TYPES
CSIDTYPDC DC 16CL1' '
CSIIOPTS DS ØCL4 CSI OPTIONS
CSICLDI DC CL1'Y' RETURN DATA OR INDX, Y OR BLANK
CSIRESUM DC CL1' ' RESUME FLAG Y OR BLANK
CSISICAT DC CL1'Y' SEARCH CATALOG Y OR BLANK
CSIRESRV DC XL1'ØØ' RESERVED
CSINUMEN DC H'Ø' NUMBER OF ENTRIES FOLLOWING
CSIENTS DS ØCL8 VARIABLE # OF ENTRIES
CSIFLDNM DC CL8' ' FIELD NAME
CSIFIELLEQU *-CSIFIELD *

* RETURNED MODULE/REASON/RETURN FROM CSI *

CSIMRR DS ØF
CSIMODID DC XL2'ØØØØ' MODULE ID
CSIRSRNC DC XL1'ØØ' REASON CODE
CSIRTNC DC XL1'ØØ' RETURN CODE

* FILE DECLARATIONS *

SYOUT DCB DDNAME=SYOUT,DSORG=PS,MACRF=PM, +
RECFM=FB,LRECL=ORECLEN

* OUTPUT AREA(S) *

OREC EQU *
OETYPE  DS  CL8  ENTRY TYPE
    DS  CL1
OENAME  DS  CL44  ENTRY NAME
    DS  CL1
OEVOL   DS  CL6  VOLSER OR BLANK
OEMVOL  DS  CL1  MULTI VOLUME OR BLANK
    DS  CL1
OCNAME  DS  CL44  CATALOG NAME
    DS  CL1
OSYSID  DS  CL4  SYSTEMID/SMFID
    DS  CL1
OPLEXNM DS  CL8  SYSPLEX NAME
ORECLEN EQU  *-OREC
    EJECT
* ________________________________________________
*        SYMBOLIC REGISTER EQUATES
* ________________________________________________
RØ     EQU  Ø
R1     EQU  1
R2     EQU  2
R3     EQU  3
R4     EQU  4
R5     EQU  5
R6     EQU  6
R7     EQU  7
R8     EQU  8
R9     EQU  9
R1Ø    EQU  1Ø
R11    EQU  11
R12    EQU  12
R13    EQU  13
R14    EQU  14
R15    EQU  15
    EJECT
* ________________________________________________
*        LITERAL(S)
* ________________________________________________
    LTORG
    EJECT
* *
*        DUMMY SECTION(S)
* ________________________________________________
    SPACE 2
* *
*        CSI RETURN WORK AREA
* ________________________________________________
CSIRWORK DSECT
* INFORMATION RETURNED FOR WORK AREA
    CSIUSRLN DS  F  TOTAL LENGTH OF WORKAREA
    CSIREQQLN DS  F  MIN REQUIRED WORK AREA LENGTH
    CSIUSDLN DS  F  TOTAL USED WORK AREA LENGTH
CSINUMFD DS H
* INFORMATION RETURNED FOR EACH CATALOG
CSICFLG DS CL1
   CATALOG FLAG
CSICTYPE DS CL1
   CATALOG TYPE
CSICNAME DS CL44
   CATALOG NAME
CSICRETN DS 0CL1
   RETURN INFO FOR CATALOG
CSICRETM DS CL2
   CATALOG RETURN MODULE ID
CSICRETR DS CL1
   CATALOG REASON CODE
CSICRETC DS CL1
   CATALOG RETURN CODE
CSIRWORL EQU *-CSIRWORK
* INFORMATION RETURNED FOR EACH ENTRY
CSIRWENT DSECT
CSIEFLAG DS XL1
   ENTRY FLAG INFO
CSIETYPE DS XL1
   ENTRY TYPE - A.B,C,D,G,H,...
CSIENAME DS CL44
   ENTRY NAME
CSIERETN DS XL4
   ENTRY ERROR INFO
CSIERETM DS CL2
   ENTRY RETURN MODULE ID
CSIERETR DS XL1
   ENTRY REASON CODE
CSIERETC DS XL1
   ENTRY RETURN CODE
CSIRWENL EQU *-CSIRWENT
* 
CSIEDATA DS 0CL3
CSIFLEN DS CL2
   FIRST LENGTH FIELD, AND SO ON
CSIFDATA DS CL1
   FIRST FIELD DATA, AND SO ON

SPACE 2

* DUMMY'S FOR SYSTEM CONTROL BLOCKS
* 
DCBD
   DCB
EJECT
CVT DSECT=YES
   CVT
EJECT
IHAECVT DSECT=YES,LIST=YES
   ECVT
EJECT
IHAIPA
   IPA
EJECT
IEESMCA
   SMCA
EJECT
*
END

CATLST JCL
//CATLST JOB (),
/**_____________________________________________________
/** LIST CATALOG ENTRIES
/**_____________________________________________________
//S1 EXEC PGM=CATLST
/// EXEC PGM=CATLST,PARM='DSN=SYS*.**'
/// EXEC PGM=CATLST,PARM='DSN=SYS1.**'
/// EXEC PGM=CATLST,PARM='CAT=MCAT.CAT.SYSCAT'
/// EXEC PGM=CATLST,PARM='CAT=UCAT.CAT.TEST'
CATLST REXX

/* REXX ——————————————————————————————————————————————————————————————*/
/* CATLST - LIST MASTER/USER CATALOG(S)                               */
/*————————————————————————————————————————————————————————————————————*/
trace off
parse upper arg parm
if parm = '.' then do
    say 'CATLST: one of the following parms',
        'must be supplied under TSO'
    say 'CATLST: DSN=dsname or CAT=catname or',
        'VOL=volser or VOL=volser/catname'
    exit 8
end
address TSO     "ALLOC FILE(SYSOUT) UNIT(VIO) NEW",
                 " TRACKS SPACE(150,150) DELETE REUSE "
if rc > 0 then do
    say 'CATLST: SYSOUT allocation error'
    exit rc
end
say 'CATLST: +———————————————————————————————————————————+'
say 'CATLST: ! ... processing your request - please wait !'
say 'CATLST: +———————————————————————————————————————————+'
address TSO     "CALL .your.loadlib(CATLST)'" "","||parm||"""
if rc = 4 then do
    say 'CATLST: no entries found or meet selection criteria'
    exit rc
end
if rc > 4 then do
    say 'CATLST: RC='!!rc
    exit rc
end
address ISPEXEC "LMINIT DATAID(TEMP) DDNAME(SYSOUT)"
address ISPEXEC "BROWSE DATAID("||temp")"
address ISPEXEC "LMFREE DATAID("||temp")"
address TSO     "FREE FI(SYSOUT)"
return

Norbert Schuech
Systems Programmer
RPZ Vienna (Austria) © Xephon 1999
JES2 checkpoint sizing

THE PROBLEM
Recently I had to increase the number of jobs that JES2 could support at our site. Not surprisingly my first concern was to check if the current checkpoint would take the increase. For the sake of speed and convenience I simply checked the size of the checkpoint on another LPAR where I knew the number of jobs supported was considerably higher. Because the checkpoint on the LPAR to be changed turned out to be nearly three times the size of the other, it seemed a safe option to carry out the change. Unfortunately, when I started JES, I received the message £HASP537 telling me that my checkpoint was too small. My error turned out to be a foolish oversight in that I had been looking at a catalogued checkpoint dataset on my reference LPAR, and not the uncatalogued one that was actually being used. The catalogued one merely being a left-over from the OS/390 install.

A SOLUTION
The result of this was to drive me back to the manuals to ensure I would not make the same mistake again. In the JES2 Initialization and Tuning guide there is a detailed method for calculating the checkpoint in the same manner as JES does before issuing the £HASP537. In order to make this calculation easier, I have translated it into REXX and arranged for the REXX to attempt to scan SYS1.PARMLIB for the values to carry out the calculation. Should you wish to exploit this REXX yourself, all that is required is that you install it into your SYSPROC as member SPOOLCAL and issue the command TSO SPOOLCAL your.parmlib (jesparm) to obtain a screen as shown in Figure 1. Note that if any errors occur, it should be because of problems in the scan process of your PARMLIB and not because of the calculation, and it should be easy to resolve.

OPERATIONAL ENVIRONMENT
Operating system and other software constraints and pre-requisites include: OS/390, JES2, and TSO/E.
The JES2 checkpoint will require 431 4K BLOCKS which equates to 36 339 tracks or equates to 44 338 tracks.

Figure 1: Sample output

SPOOLCAL REXX

/* REXX */
arg dsname
/* */
/* This REXX reads the JES2 parm member to pick up the necessary */
/* information to allow a calculation of the number of 4K blocks */
/* needed to estimate the JES checkpoint size */
/* */
/* */
x=OUTTRAP("save.") /* eliminate messages */
'FREE FI(SPONGE)'
"ALLOC FI(SPONGE) DA("dsname") SHR"
'EXECIO * DISKR SPONGE (FINIS' DO QUEUED()
PULL line
IF INDEX(line,'TGSPACE=(MAX=')¬Ø THEN DO /* max found */
PARSE var line .'=(MAX=' max ')'.
PARSE VAR max max ','.
END
IF INDEX(line,'JOENUM=')¬Ø THEN DO /* max found */
PARSE var line 'JOENUM=' joenum
PARSE VAR joenum joenum ','.
END
IF INDEX(line,'JOBNUM=')¬Ø THEN DO /* max found */
PARSE var line 'JOBNUM=' jobnum
PARSE VAR jobnum jobnum ','.
END
IF INDEX(line,'SPOOLNUM=')¬Ø THEN DO /* max found */
PARSE var line 'SPOOLNUM=' spoolnum
PARSE VAR spoolnum spoolnum ','.
IF INDEX(line,'LOGSIZE=')¬=Ø THEN DO /* max found */
    PARSE var line 'LOGSIZE=' logsize
    PARSE VAR logsize logsize ',' .
END

"FREE FJ(Sponge)"

/* default corrections */
/* if logsize not specified assume 1 */
/* max must be multiple of 16288 */
/* */
IF logsize='' THEN logsize=1
rem=max//16288
IF rem¬=Ø THEN max=16288*((max%16288)+1)
/* now calculate the size of the ckpt */
/* CONSTANTS */
prefix=24 /* NUMBER OF BYTES FOR EACH CONTROL BLOCK */
rnd=0.5 /* rounding factor */
pg=4096 /* size of a page in bytes */
/* */
/* ALL VALUES CALCULATED ARE IN BYTES. THESE NEED TO BE CONVERTED */
/* TO 4K BLOCKS, AND ALL FRACTIONS MUST BE ROUNDED UP. */
/* */
/tgm=(max/4)+prefix;tgm=FORMAT((tgm/pg)+rnd,,0)
/scq=(32*32*16)+prefix;scq=FORMAT((scq/pg)+rnd,,0)
/jix=(32767*2)+prefix;jix=FORMAT((jix/pg)+rnd,,0)
/jobq=(jobnum+1)*(96+spoolnum/8)+prefix;jobq=FORMAT((jobq/pg)+rnd,,0)
/pst=(jobenum*4)+prefix;pst=FORMAT((pst/pg)+rnd,,0)
/jot=(joenum*104)+520+prefix;jot=FORMAT((jot/pg)+rnd,,0)
/tgr=(32*3*255)+prefix;tgr=FORMAT((tgr/pg)+rnd,,0)
/rs0=9999+prefix;rs0=FORMAT((rs0/pg)+rnd,,0)
/lck=(56*8)+prefix;lck=FORMAT((lck/4096)+rnd,,0)
/das=(spoolnum*212)+prefix;das=FORMAT((das/4096)+rnd,,0)
/* */
/* THEREFORE CHECKPOINT RECORDS IS */
/* */
total=tgm+scq+jix+jobq+pst+jot+tgr+rs0+lck+das
/* */
/* NOW CALCULATE THE MASTER RECORD */
/* */
hct=580;QSE=200*32;extension=4000
/kit=10*36;ckptio=4*total:dase=2*spoolnum
/master_total=hct+QSE+extension+kit+ckptio+dase
/master_total=FORMAT((master_total/pg)+rnd,,0)
/* */
/* NOW NEED THE SIZE OF THE CHANGE LOG */
/* */
logsize=1
/* */
INTRODUCTION
Those of you who have used VSE will certainly remember that there is a very nice feature in VSE systems – while you are browsing the system log, you can obtain an on-line explanation of system messages by pressing the PF9 key. While there are also similar solutions for MVS or OS/390, these solutions require specific software such as the use of a particular terminal emulation program. We have developed another solution that enables a user to obtain an on-line explanation of any system (or application-specific) messages and code in an efficient
manner. There is no need to invoke the Bookmanager program to achieve it. There are two steps in our solution.

- To transfer the system messages and codes into a manageable format.
- To extract the system message from the screen contents, perform look-up and display the message explanation.

For the first step, we have to first extract the messages from the messages and code manuals to a text file. This can be performed by the ‘copy’ function of the IBM Bookmanger Library reader (DOS or OS/390 version) or the ‘print to file’ function of the Windows version. If the text file is prepared under DOS or Windows, then it is uploaded to the host using file transfer programs such as TSO IND$FILE or TCP/IP FTP, using the ASCII option. In order to save DASD space, it is better to allocate a variable block dataset (ie RECFM=VB) for the destination dataset since there are a lot of empty lines in the text file. Some editing may be required to change some the non-printable hexadecimal characters to space after the file transfer is performed.

The next step is to split the large text file into multiple entries, one for each message.

To make things simple, for each message and code manual, one large PDS is used and one member of the PDS corresponds to one message. This makes the message look-up very simple and efficient. It is, of course, up to you to determine how many members each PDS contains to make the retrieval faster. Note that you have to reserve sufficient directory blocks for the PDS. Typically, about every 20 members require one directory block when ISPF statistics are turned off (six members when turned on).

We then use the ISPF editor to copy the text file containing the messages and codes to the PDS and perform the splitting. It is achieved using an edit macro, CREBK, which first identifies the message ID (assuming that it is at the first line of the text file), then searches for the beginning of the next message, and creates a new member for every message within the text file. This makes use of the fact that all the message IDs appear in the same column of the text file. Sometimes we may have to trim the length of the message ID because
some, like, DFHSI1517, have more than eight characters.

For a large manual, like the five volumes of OS/390 system messages, it will take a large amount of time, so it is suggested you perform the splitting of messages in non-prime time. After the splitting process is completed, we proceed to the second step, to extract the message and code from screen and perform the look-up.

It is quite difficult to find out from the screen how to extract the message because that invokes a look-up of the ISPF screen buffer. Luckily, we found that a similar function is provided from the freeware DSLIST REXX program, which is available from CBT tape, file 183. (For more information on the DSLIST program and the CBT file 183, please refer to the Web site http://members.home.net/gsf/tools/ or the CBT homepage http://www.cbtape.org) With reference to that, we have written another REXX program, GETSC, which extracts the word under the cursor position, then calls different REXX programs to perform the message and code look-up depending on the contents. REXX program SM390 is for the five volumes of the OS/390 messages, and SC390 is for the OS/390 system codes. What these REXX programs do is just browse the message and code PDSs for the corresponding member, in which the directory look up is performed automatically.

The final thing to do is to assign a PF key to the GETSC program so that it can be accessed in a point-and-shoot way (by placing the cursor over the system message to be looked up and pressing that function key). Please note that the program must be invoked as a TSO function, i.e., when you assign a PF key to the command, you must specify TSO %GETSC in the PF key definition. Alternatively, you can also define GETSC to be an ISPF command so that you can just specify GETSC in the PF key definition. To do so, add the following to ISPF command table using the ISPF command table utility:

<table>
<thead>
<tr>
<th>Verb</th>
<th>T</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>GETSC</td>
<td>Ø</td>
<td>SELECT CMD(%GETSC) PARM(&amp;ZPARM)</td>
</tr>
</tbody>
</table>

The REXX programs are tested and work under MVS/ESA Version 4 Release 3 with ISPF Version 3 Release 5, and OS/390 Version 1 Release 3. The response time is also quite fast, even when several
thousand members are placed in each PDS. This approach enables any system message or user-defined messages to be readily looked up by just pressing a key, saving much of the time in finding the hard-copy and turning over pages. We found the time spent on uploading and splitting the members is worthwhile.

GETSC

/* REXX */
ADDRESS ISPEXEC; "CONTROL ERRORS RETURN"
/* —————————————————————————————————————————————————— */
/* RETRIEVE LINE ADDRESS AND CURSOR POSITION */
/* CODE ADAPTED FROM DLIST COMMAND FROM CBT 183 */
/* —————————————————————————————————————————————————— */

TCB = PTR(54Ø) /* TCB (EXEC COMMAND) PSATOLD */
TCB = PTR(TCB+132) /* TCB (ISPTASK) TCBOTC */
FSA = PTR(TCB+112) /* FIRST SAVE AREA TCBFSA */
R1 = PTR(FSA+24) /* ISPTASK'S R1 */
TLD = PTR(R1) /* TLD ADDRESS */
TLS = PTR(TLD+Ø96) /* SCREEN BUFFER TLDTLSP */
CSR = PTR(TLD+164) /* RELATIVE CURSOR POS. TLDCSR */
SCRW = PTR(TLD+192) /* SCREEN WIDTH TLDCLSWD */
OFFL = SCRW * TRUNC(CSR/SCRW) /* OFFSET TO CURRENT LINE */
CSR = PTR(TLD+192) /* SCREEN WIDTH TLDCLSWD */
MESSCODE=''

VALID='ABCDEFGHIJKLMNOPQRSTUVWXYZØ123456789'
UPPER LINE:
P=VERIFY(LINE,VALID,,CSRP) /* FIND DELIMITER AFT DSN */
IF P>Ø THEN LINE=LEFT(LINE,P-1) /* TRUNCATE AFTER DSN */
P=VERIFY(REVERSE(LINE),VALID) /* FIND DELIMITER BEF DSN */
IF P>Ø THEN LINE=RIGHT(LINE,P-1) /* TRUNCATE BEFORE DSN */
MESSCODE = LINE

/* —————————————————————————————————————————————————— */
/* GET MESSAGE CODE FROM USER IF NULL INPUT FROM SCREEN */
/* —————————————————————————————————————————————————— */

DO WHILE MESSCODE=''
ADDRESS ISPEXEC 'ADDPOP'
ZWINTTL = 'OS/39Ø SYSTEM MESSAGE '
PROMPT = 'PLEASE INPUT A SYSTEM MESSAGE, <F3> TO EXIT'
ADDRESS ISPEXEC 'DISPLAY PANEL(ASKMENU)';
IF RC <> Ø THEN EXIT
MESSCODE = ANS
ADDRESS ISPEXEC 'REMPOP ALL';
END
/* —————————————————————————————————————————————————— */
/* FIND THE SYSTEM MESSAGE OR CODE */
/* DEPENDING ON THE PREFIX OF THE MESSAGE */
/* THE FOLLOWING CODE SHOULD BE CUSTOMIZED DEPENDING ON THE ACTUAL */
/* SET UP OF SYSTEM MESSAGE OR CODE DATASETS */

IF SUBSTR(MESSCODE,1,3) = 'DFH' THEN
   /* CICS MESSAGES */
   CALL $CICSMMSG MESSCODE
END
ELSE
   DO
   IF SUBSTR(MESSCODE,1,1) = '$' THEN
      /* JES2 MESSAGES */
      CALL $JESM39Ø MESSCODE
   END
   ELSE
      IF LENGTH(MESSCODE) > 4 THEN
         /* OS/39Ø MESSAGES */
         CALL $SM39Ø MESSCODE
      ELSE
         /* CONSIDER THE REST AS SYSTEM CODES */
         CALL $SC39Ø RIGHT(MESSCODE,3)
      END
   END
END
RETURN

PTR: ARG VALUE; RETURN X2D(C2X(STORAGE(D2X(VALUE),4)))

ASKMENU PANEL

)ATTR
+ TYPE(TEXT) INTENS(LOW) COLOR(WHITE)
- TYPE(TEXT) INTENS(LOW) COLOR(TURQ)
* TYPE(TEXT) INTENS(LOW) COLOR(BLUE)
! TYPE(INPUT) INTENS(LOW) CAPS(ON)
   COLOR(PINK) HILITE(USCORE)
@ TYPE(INPUT) INTENS(LOW) CAPS(OFF)
   COLOR(TURQ) HILITE(USCORE)
)BODY WINDOW(60,5) CMD(ZCMD)
%CMD ==> @Z
%
%&PROMPT
%
%INPUT ==>!Z
)INIT
   .ZVARS = '(ZCMD ANS)'
   &ZCMD = ''
   .CURSOR=ANS
)REINIT
   .CURSOR=ANS
)PROC
   VPUT (ANS) SHARED
)END

CREBK

/** REXX **/
/** FOR STORAGE CONSIDERATIONS PLS USE VB FILE DURING UPLOAD */
/* TO USE THIS REXX, EDIT THE MESSAGE DATASET SO THAT THE FIRST */
/* CONTAINS THE FIRST MESSAGE ID */
/* SO THAT THE REXX CAN DYNAMICALLY DETERMINE THE POSITION */
/* AND THE PREFIX OF THE MESSAGE */
ADDRESS ISREDIT
'MACRO ()'
'RESET'
    /* REDUCE THE NUMBER OF DIRECTORY BLOCKS USED BY TURNING OFF */
    /* ISPF STATISTICS */
'RESET'
/* THE FIRST LINE OF THE MEMBER SHOULD CONTAIN THE MESSAGE TITLE */
/* SO THAT THE PROGRAM CAN DETERMINE THE POSITION OF THE TITLE */
AUTOMATICALLY */
'LOCATE .ZFIRST'
    '(L1) = LINE .ZCSR'
    L2=STRIP(L1,'L')
    ARG=SUBSTR(L2,1,3)
    SAY 'THE MESSAGE PREFIX IS ' ARG
    APOS = POS(ARG,L1)
    SAY 'THE POSITION OF THE PREFIX IS AT ' APOS
'F FIRST '||ARG||' '||APOS
'(I) = FIND_COUNTS'
DO WHILE I > Ø
    '(X Y) = CURSOR'
    '(L1) = LINE .ZCSR'
    'LABEL .ZCSR = .PROC'
    PROCNAM = WORD(SUBSTR(L1,APOS),1)
    NEWNAM = ''
    /* THERE ARE CASES WHERE ONE MESSAGE HAS MULTIPLE ENTRIES */
    /* IN THE MESSAGE AND CODE MANUAL IE IEAØØØI. */
    /* THE FOLLOWING LOOP FIND ALL OF THEM OUT AND PLACE */
    /* THEM IN THE SAME MEMBER */
    DO UNTIL NEWNAM <> PROCNAM
        'F '||ARG||' '||APOS
        IF RC <> Ø THEN
            DO
                NEXTNF = 1
                LEAVE
            END
        END
        NEWNAM = SUBSTR(NEWNAM,1,8)
    END
    IF RC = Ø THEN DO
        '(X Y) = CURSOR'
        XX = X - 7           /* MOVE UP SEVERAL LINES */
'LABEL ' || XX || ' = .PRND'
IF LENGTH(PROCNAM) > 8 THEN
  DO
    PROCNAM = SUBSTR(PROCNAM,1,8)
  /* CHANGE TO IF LENGTH(PROCNAM) > 8 THEN
     PROCNAM = SUBSTR(PROCNAM,4,8) FOR DFHXXYYYY */
  /* SINCE SOME CICS MESSAGES ARE LONGER THAN 8 CHARACTERS */
  END
/* FOR SYSTEM CODES, ADD '#' TO THE BEGINNING OF PROCNAM
   SINCE MAY SYSTEM CODES BEGIN WITH A NUMBER AND CANNOT
   BE USED FOR MEMBER NAMES */

'CREATE ' || PROCNAM || ' .PROC .PRND'
IF RC = Ø THEN DO
  'DELETE .PROC .PRND'
  IF RC = Ø THEN DO
    'F FIRST '||ARG||' '||APOS
    'I) = FIND_COUNTS'
  END
  ELSE LEAVE
END
ELSE LEAVE
END
ELSE LEAVE
END
END
IF NEXTNF = 1 THEN DO
  SAY 'THE REST OF THE MESSAGES IS SAVED IN MEMBER ' PROCNAM
  IF LENGTH(PROCNAM) > 8 THEN
    DO
      PROCNAM = SUBSTR(PROCNAM,1,8)
    /* CHANGE TO PROCNAM = SUBSTR(PROCNAM,4,8) FOR DFHXXYYYY */
    END
  'CREATE ' || PROCNAM || ' .ZCSR .ZLAST'
  IF RC = Ø THEN
    'DELETE .ZCSR .ZLAST'
  END
END
RETURN

SM390
/* REXX */
ADDRESS ISPEXEC 'CONTROL ERRORS RETURN'
ARG CODE
CODE=STRIP(SUBSTR(CODE,1,8))
CODE2=SUBSTR(CODE,1,3)
MSG = 'Y'
SELECT
  /* ONE PDS FOR EACH SYSTEM MESSAGE MANUAL */
  WHEN CODE2 >= 'ABA' & CODE2 <= 'ASA' & CODE2 <> 'ACP' THEN
    DATASET = 'XTSB.SYSTEM.MESSAGES.ABA-ASA'
  WHEN (CODE2 >= 'ASB' & CODE2 <= 'EZM') | CODE2 = 'ACP' THEN
    DATASET = 'XTSB.SYSTEM.MESSAGES.ASB-EZM'
WHEN CODE2 >= 'GDE' & CODE2 <= 'IEB' THEN
  DATASET = 'XTSB.SYSTEM.MESSAGES.GDE-IEB'
WHEN CODE2 >= 'IEC' & CODE2 <= 'IFD' THEN
  DATASET = 'XTSB.SYSTEM.MESSAGES.IEC-IFD'
WHEN CODE2 >= 'IGD' & CODE2 <= 'IZP' THEN
  DATASET = 'XTSB.SYSTEM.MESSAGES.IGD-IZP'
ELSE
  MSG = 'N'
END

IF MSG = 'Y' THEN
  ADDRESS ISPEXEC "BROWSE DATASET('"||DATASET||"(" || CODE || "")')"
IF RC <> Ø | MSG = 'N' THEN DO
  ZEDSMGS = 'MSG ' || CODE || ' NOT FOUND.'
  ZEDLMSG = 'UNABLE TO OBTAIN EXPLANATION FOR MESSAGE '|| CODE || '.'
  ADDRESS ISPEXEC 'SETMSG MSG(ISRZ001)'
END
EXIT

SC390
/* REXX */
ADDRESS ISPEXEC 'CONTROL ERRORS RETURN'
ARG CODE
IF LENGTH(CODE) = 3 THEN DO
  /* SYSTEM CODES ØCX AND FNN APPEARS ON SAME MEMBER */
  IF SUBSTR(CODE,1,1)='F' THEN CODE='FNN'
  IF SUBSTR(CODE,1,2)='ØC' THEN CODE='ØCX'
  ADDRESS ISPEXEC "BROWSE DATASET('XTSB.SYSTEM.CODES.OS39Ø(#" || CODE || ")")"
END
IF RC <> Ø | LENGTH(CODE) <> 3 THEN DO
  ZEDSMGS = 'CODE ' || CODE || ' NOT FOUND.'
  ZEDLMSG = 'UNABLE TO OBTAIN EXPLANATION FOR CODE '|| CODE || '.'
  ADDRESS ISPEXEC 'SETMSG MSG(ISRZ001)'
END
EXIT

Chan Tin Pui
The Government of the Hong Kong SAR (China) © Xephon 1999
An IPL subsystem

THE PROBLEM
This Dirt Cheap Initial-Program-Load Subsystem (DCIPLS) is dedicated to all of the mainframe operators worldwide who are assigned the onerous task of laboriously typing and entering the system commands that are required to activate on-line systems, started tasks, and other program products. The historical approach to IPLing a mainframe was prone to errors ranging from documentation to typing.

A SOLUTION
DCIPLS eliminates such errors. It operates on a principle of one-start-command one-response – an approach so simple that even a manager would have no difficulty IPLing a mainframe into its intended full capability.

DCIPLS may be used to terminate all of a mainframe’s activities, except JES2, and it may be used to activate all mainframe activities. DCIPLS does not actually activate all activities here because Operations wanted to retain a modicum of control for itself. However, it would be relatively simple for a systems programmer to enhance it to do so. It may be used to deactivate on-line systems in preparation of reloading an NCP and then reconnecting them to VTAM after the NCP and VTAM are active once again. DCIPLS verifies that NCP channels defined within it, for the system on which it is active, are on-line before proceeding with its processing; if they are not, it attempts to place them on-line and, if unsuccessful, will prompt an operator to do so.

INSTRUCTIONS
In order to activate DCIPLS, enter – S DCIPLES where DCIPLES is the name of a PROC that has been stowed in a procedure library. When DCIPLS is ready to process commands, it notifies the operator who initiated it to respond by entering a valid command. The character ‘?’
was chosen as the command character for our shop. It is set by the instruction following the one with the label of DCSTOID in DCIPLS.

Several of DCIPLS’ commands are universally applicable to all systems; others are system-specific. ?PAP, ?WARN, ?CONNECT, ?HALT, ?VERIFYUP, and ?VERIFYDW are universal commands for all systems. ?sysidUP and ?sysidDOWN are specific to the system whose identifier is ‘sysid.’

?PAP causes cessation of all activities on the system for which it was entered. ?WARN message broadcasts a warning message to all logged-on users of ROSCOE and TSO. ?CONNECT re-establishes links between CICS and IMS and VTAM. ?sysidUP will reactivate all system activities that were suspended for sysid – sysid must be the identifier of the system on which the command was entered, otherwise the command will be rejected. Similarly, ?sysidDOWN will terminate activities and break connections between applications, including ones that can tolerate it, and VTAM so that the NCP can be reloaded. Since DCIPLS processing is similar in all regards on all domains in our shop, only examples of the commands used for our production domain will be provided.

DCIPLS comprises three components – DCIPLS, DCIPLSFR, and DCIPLSRB.

- DCIPLS is the mainline body of code. It loads DCIPLSFR, initializes cells that are to contain commands and, when one is available, peels it off the chain of commands and processes it. A command is subrogated into the name of member that resides in a PDS used to house source code (DDname COMMAND). A member may contain commands used to cancel, stop, and modify a task’s activity, or it may contain valid responses to an outstanding WTOR. Some tasks that are common to all operating systems, such as LLA, VLF, etc, are terminated via coded commands. Checks are made to ensure that DB2 finishes its processing before LLA and VLF are terminated. All ‘batch’ work must have completed before DCIPLS terminates LLA, VLF, VTAM, and ThruPut manager. ‘Batch’ work as is used in this context means any task not in performance group zero nor in seventeen (started task).
Since the flow of logic within DCIPLS is dependent upon a system’s identifier, DCIPLS must of necessity be modified before it can be used elsewhere. Replace our system identifiers (VS01, VS02, ..., VS05) with yours. I would suggest that you test DCIPLS by replacing the system identifier VS05 with one of yours, modify the contents of the members described in a following section to be compatible with your environment, but use the same member names, and change, if necessary, the performance group used for started tasks. This would allow you to conduct a test of DCIPLS in your shop with a minimum of change. The name of your VTAM application may also require changing to be that of yours. NET is the name of our VTAM application. A1 and A4 are the only two valid operands of LIST= when VTAM is started. DCIPLS expects the member CURLIST in the source PDS to be one of these. The DDname of the source PDS is CURLIST. Channel addresses for your 37x5 must replace the values in PPGVS01, etc. The miniscule amount of effort required to make DCIPLS work in your environment is irrelevant since the reward, in the form of reduced down time for your system, far exceeds the value of that effort.

- DCIPLSFR acquires control whenever an operator enters a command. If the command entered does not have a control character of ?, then no further processing of it takes place. If the command entered is intended for DCIPLS, then it is moved into a holding cell and an SRB is constructed and scheduled to convey to DCIPLS that it has a command to process.

- DCIPLSRB notifies DCIPLS, via POST, that a command is available for it to process, then notifies DCIPLSFR, via Cross-Memory POST, that it can free the resources which it obtained for DCIPLSRB’s processing.

All of DCIPLS’ components must reside in an authorized library that is in the LINKLST concatenations. All pieces must be link-edited with an option of AC=1.

DCIPLS could be readily modified to allow control of its processing sequence to be done via an option on the start command such as is done with VTAM. This would obviate the need for DCIPLSRB and DCIPLSFR. I will not engage in philosophizing why DCIPLS’ processing is arranged the way it is.
My advice to anyone who wants to use DCIPLES is to read the code, ignoring all that has been written in this article, because the code is the final authority on what actually happens whenever it is invoked.

JCL

//DCIPLES PROC
//DCIPLES EXEC PGM=DCIPLES,TIME=1440
//CURLIST DD DSN=SYS1.CURLIST,DISP=SHR
//COMMAND DD DSN=SYS1.COMMANDS,DISP=SHR

DCIPLES
TITLE 'DIRT CHEAP INITIAL-PROGRAM-LOAD SUBSYSTEM'
SPACE 1
DCIPLES CSECT ,
DCIPLES AMODE 31
DCIPLES RMODE 24
SPACE 1

***********************************************************************
*        DIRT CHEAP IPL SYSTEM                                        *
*                                                                     *
*    COMMAND PROCESSING SUBSYSTEM                                     *
*                                                                     *
*    PROVIDES SUPPORT FOR OPERATOR COMMANDS AS FOLLOWS:               *
*                                                                     *
*              INITIALIZE THIS SUBSYSTEM                              *
*              - BUILD SSVT                                          *
*              - BUILD AND CHAIN CELLS TO HOLD COMMANDS               *
*              - LOAD FUNCTION ROUTINE INTO FIXED COMMON STORAGE      *
*              - CHAIN SSVT TO SSCVT                                *
*                                                                     *
*              MAINLINE                                               *
*              - WAIT FOR OS SERVICE REQUEST ROUTINE TO POST WAIT     *
*              - INDICATING A COMMAND HAS ARRIVED TO PROCESS         *
*              - ECHO OPERATOR COMMAND TO ISSUING CONSOLE             *
*              - PROCESS COMMAND                                     *
***********************************************************************
EJECT

USING PSA,R0 ESTABLISH PSA ADDRESSABILITY
SPACE 1
SAVE (14,12),* SAVE REGISTERS.
SPACE 1
LR R12,R15 SET BASE REGISTER (USES THREE BASES)
USING DCIPLES,R12,R7,R11 ESTABLISH ADDRESSABILITY TO SUBSYS
LA R7,2048(R12) SET UP 2ND BASE REGISTER
LA R7,2048(R7)
LA R11,2048(R7) SET UP 3RD BASE REGISTER
LA R11,2048(R11)
SPACE
LA R0,72  SIZE OF REGISTER SAVE AREA
SR R2,R2  NUMBER OF SUBPOOL
BAS R10,CPSTORA GET REGISTER SAVE AREA
SPACE 1
MVI 0(R1),0 INITIAL ZERO
MVC 1(71,R1),0(R1) PROPAGATE ZEROES
ST R1,8(R13) CHAIN ( FORWARD )
ST R13,4(,R1) SAVE ( BACKWARD )
LR R13,R1 AREAS ( CURRENT )
EJECT
***********************************************************************
* ASCERTAIN IF DCIPLS IS FAIT ACCOMPLI                           *
***********************************************************************
SPACE 1
L R1,PSAAOLD ISHMAEL
USING ASCB,R1 ESTABLISH ASCB ADDRESSABILITY
SPACE 1
ICM R1,15,ASCBJBSNS POINTER TO START/MOUNT/LOGON TASK
BE CMATASID IF NOT AVAILABLE, USE PROGRAMED NAME
MVC PATNAME,0(R1) SET NAME OF TASK USED FOR DCIPLS
SPACE 1
DROP R1 FORGET ASCB
SPACE 1
CMATASID BAS R10,CMRENQ SERIALIZE ON USE OF CSCB CHAIN
SPACE 1
L R4,CVTTPTR POINT TO CVT
USING CVT,R4 ESTABLISH CVT ADDRESSABILITY
L R5,CVTMSER DATA AREA OF MSTR SCHED RES DATA AREA
USING CHAIN,R5 SET ADDRESSABILITY TO CHAIN CSCB
SR R6,R6 ZERO ACTIVE COUNTER
SPACE 1
DCLOC ICM R5,15,CHPTR CSCB CHAIN POINTER
BZ ENDCSCB TEST FOR END OF CHAIN
SPACE 1
CLC CHKEY,PATNAME TEST FOR DCIPLS THAT ARE ACTIVE
BNE DCLOC GET ANOTHER CSCB CHAIN POINTER
LA R6,1(R6) ADD 1 TO ACTIVE DCIPLS COUNTER
B DCLOC GET ANOTHER CSCB CHAIN POINTER
SPACE 1
ENDCSCB C R6,PATONE CHECK DCIPLS ACTIVE COUNTER
BE BESUP ONLY ONE IS ACTIVE
SPACE 1
BAS R10,CMRDEQ REMOVE SERIALIZATION OF CSCB CHAIN
SPACE 1
WTO 'DCIPLØ7E SUBSYSTEM ALREADY ACTIVE',DESC=2,ROUTCDE=8
SPACE 1
B CPRETURN EXIT - ANOTHER DCIPLS IS ACTIVE
SPACE 1
DROP R4,R5
EJECT
***********************************************************************
* ISSUE MODESET TO ENTER SUPERVISOR STATE IN KEY ZERO.  *
*****************************************************************************
SPACE 1
BESUP BAS R1Ø,CMRDEQ REMOVE SERIALIZATION OF CSCB CHAIN
SPACE 1
MODESET MODE=SUP,KEY=ZERO BECOME GEORGE
SPACE 1
* ESTABLISH RECOVERY ENVIRONMENT
SPACE 1
LA R3,PATEXIT POINT TO STAE EXIT ROUTINE ADDRESS
ESTAE (R3),PARAM=PATLIST ESTABLISH STAE ENVIRONMENT
LTR R15,R15 TEST IF UNDER STAE AEGIS
BE PATGO BRANCH IF SO
SPACE 1
WTO 'DCIPLØØE UNABLE TO ESTABLISH ESTAE ENVIRONMENT'
SPACE 1
B CPRETURN DEPART
EJECT
PATGO DS ØH PROVIDE TARGET FOR BRANCH OP CODE
* ESTAE ERRTN,TERM=YES ESTABLISH ERROR RECOVERY.
SPACE 1
L RØ,WORKSP LENGTH OF WORK AREA
LA R2,252 SET NUMBER OF SUBPOOL
BAS R1Ø,CPSTORA OBTAIN WORK AREA FOR WTO MESSAGES
SPACE 1
LR R9,R1 SET WORK AREA BASE.
USING WORK,R9 ESTABLISH WORK AREA ADDRESSABILITY.
SPACE 1
LR RØ,R9 POINT TO WORK AREA
LA R1,WORKLENH SET SIZE OF WORK AREA
SR R15,R14 SET FILL CHARACTER TO HEXADECIMAL Ø
MVCL RØ,R14 INITIALIZE WORK AREA TO HEX ZEROES
SPACE 1
*****************************************************************************
* GET THE SYSTEM ID AND STOW IT FOR LATER USE  *
* IN VERIFICATION PRIOR TO ACTIVATION OR DEACTIVATION.  *
* IF MTO IS NOT ON PROPER SYSTEM, DCIPLES DOES NOTHING.  *
*****************************************************************************
SPACE 1
L R1,CVTPTR ADDRESS OF CVT
USING CVT,R1 ESTABLISH CVT ADDRESSABILITY
L R1,CVTSMCA ADDRESS OF SMF CONTROL AREA
USING SMCABASE,R1 ESTABLISH SMF ADDRESSABILITY
MVC SYSID,SMCASID SAVE SYSTEM ID (VSØ1=DEVELOPMENT
* VSØ2=YMM
* VSØ3=TECHNOLOGY
* VSØ4=ACCENT
* VSØ5=PRODUCTION).
EJECT
*****************************************************************************
* ASCERTAIN AVAILABILITY OF ALL 37X5-TYPE DEVICES  *
* THAT ARE REQUIRED FOR THIS SYSTEM'S IDENTIFIER  *
*****************************************************************************
SPACE
LA R14,PPGSYSNT NUMBER OF SYSTEM-IDENTIFIER ENTRIES
LA R1,PPGSYSTM POINT TO FIRST ENTRY
SPACE
PPGFNID CLC SYSID,Ø(R1) TEST IF IDENTIFIER OF THIS SYSTEM
BE PPGSETID BRANCH IF SO
SPACE
LA R1,PPGSYSIZ(R1) POINT TO NEXT ENTRY
BCT R14,PPGFNID ATTEMPT TO LOCATE SYSTEM ENTRIES
SPACE 1
WTO 'DCIPLØ4I UNABLE TO DETERMINE SYSTEM IDENTIFICATION; ACT TIVATION PROCESS TERMINATED'
SPACE 1
B DCABORT AT END ABORT DCIPLES
SPACE 1
PPGSETID ST R1,CLAMHOLD SET ADDRESS OF THIS SYSTEM'S 37X5S EJECT
***********************************************************************
* LOCATE THIS SUBSYSTEM'S SSVT AS FOLLOWS: *
* JESCT POINTS TO SCCVT(SSCT) AND IT POINTS TO SSVT. *
* EACH UNIQUE SUBSYSTEM HAS AN SCCVT THAT IS CONSTRUCTED FROM *
* THE JES2 AND SUBSYSTEM NAME TABLES. *
* SSVT IDENTIFIES FUNCTIONS WITHIN DCIPLES SUBSYSTEM (DCIPLSUB). *
* THE STATE OF SCCVT/SSVT UPON ENTRY CAN BE ONE OF THE FOLLOWING:* *
* 1. NEITHER-BUILD BOTH *
* 2. SSCVT BUT NO SSVT-BUILD SSVT *
* 3. BOTH PRESENT-LOAD FUNCTION ROUTINE(DCIPLSFR) ADRESED IN SSVT*
***********************************************************************
SPACE 1
PATRETRY L R1,CVTPTR ADDR OF COMMUNICATIONS VECTOR TABLE
 USING CVTMAP,R1 ESTABLISH CVT ADDRESSABILITY
 SPACE 1
 L R1,CVTJESCT OBTAIN ADDRESS OF THE JESCT
 USING JESCT,R1 SET JES CONTROL TABLE ADDRESABILITY
 SPACE 1
 ICM R6,15,JESSSCT FETCH ADDRESS OF THE SSCT
 USING SSCT,R6 SET BASE OF FIRST SUBSYS COMM TABLE
 SPACE 1
DCSEARCH BZ CPBLDCIP THEN BUILD SCCVT FOR DCIPLES
 LR R2,R6 PREERVE ADDRESS OF LAST SCCVT
 CLC SSCTS Nam,PATNAME IF CORRECT SUBSYSTEM NAME
 BE DCGOTSSN THEN PROCESS IT
 ICM R6,15,SSCTSSCTA ELSE LOAD ADDRESS OF NEXT SCCVT
 B DCSEARCH AND CONTINUE SEARCH
 SPACE 1
DCGOTSSN ICM R8,15,SSCTSSVT GET SSVT POINTER
 USING SSVT,R8 ESTABLISH ADDRESSABILITY TO SSVT
 BZ DCINTSSVT IF NO SSVT, THEN CREATE ONE
 BAS R1Ø,CPLOADSS LOAD SUBSYSTEM FUNCTION ROUTINE
 B DCSTOID SSVT AND BYPASS INITIALIZATION
 SPACE 1
DROP R1

© 1999. Reproduction prohibited. Please inform Xephon of any infringement. 57
EJECT
***********************************************************************
* LOAD THE SUBSYSTEM FUNCTION ROUTINE INTO FIXED COMMON VIRTUAL STORAGE.*
* INITIALIZE THE SSVT WITH THE ADDRESS THAT IS RETURNED IN GENERAL PURPOSE REGISTER ZERO. *
***********************************************************************

SPACE 1
CPLOADSS LOAD EP=DCIPLSR,GLOBAL=(YES,F),EOM=YES GET SUBSYS FUNCT RTN
SPACE 1
ST R0,SSVTFTN PUT FUNCTION ROUTINE ADDR INTO SSVT
BR R1Ø RETURN TO CALLER
EJECT
***********************************************************************
* GETMAIN AND INITIALIZE THE SSCT TO ZEROS; *
* STOW SSCVT IDENTIFIER IN FIRST FULL WORD OF SSCT; *
* STOW 'DCIPLES' IN SUBSYSTEM NAME FIELD OF SSCT; *
* THEN PROCEED... *
***********************************************************************

SPACE 1
CPBLDCIP DS ØH
SPACE 1
LR R6,R2 PRESERVE R2
L R0,SSCTSP SIZE AND SUBPOOL OF SSVT AREA
LA R2,245 NUMBER OF STORAGE SUBPOOL
BAS R1Ø,CPSTORA ACQUIRE AREA FOR SSVT CONTROL BLOCK
LR R2,R6 REINSTATE R2
SPACE 1
LR R6,R1 SET SSCT BASE REGISTER
LR R14,R6 REPEAT SSCT BASE (MVCL TARGET)
LA R15,SSCTSIZE SET SIZE OF SSCT AND FILL CHARACTER
LR R0,R6 REPEAT SSCTBASE (MVCL SOURCE)
SR R1,R1 SIZE OF TARGET(ZERO=>FILL CHAR ONLY)
MVCL R14,R0 CLEAR SSCT AREA
SPACE 1
MVC SSCTID,PATSSCVT SET CONTROL BLOCK IDENTIFIER
MVC SSCTSnam,PATNAME SET NAME OF SUBSYSTEM IN SSCT
SPACE 1
CS R15,R6,SSCTSCTA-SSCTID(R2) LINK DCIPLS SSCT WITH OTHERS
BZ DCGOTSSN BRANCH IF SUCCESSFUL, ELSE
SPACE 1
LA R2,245 LOAD SSCT SUBPOOL AND
L RØ,SSCTSP LENGTH AND
LR R1,R6 ADDRESS AND
BAS R10,CPSTORF THEN FREE IT
B PATURETRY 'PLAY IT AGAIN, SAM.'
EJECT
***********************************************************************
* GETMAIN AND INITIALIZE THE SSVT TO ZEROS. *
* INDICATE THAT THE COMMAND BROADCAST FUNCTION HAS BEEN ACTIVATED. THE SSVT WILL BE LOADED WITH THE ADDRESS OF THE SUBSYSTEM FUNCTION ROUTINE AT A LATER TIME. *
DCINTSVT DS 0H

L R0,SSVTSP SIZE AND SUBPOOL OF SSVT AREA
LA R2,245 SET NUMBER OF STORAGES SUBPOOL
BAS R10,CPSTORA ACQUIRE VIRTUAL STORAGE FOR SSVT

LR R8,R1 SET SSVT BASE REGISTER
LR R14,R8 REPEAT SSVT BASE (MVCL TARGET)
LA R15,SSVTLEN SET SIZE OF SSVT AND FILL CHARACTER
LR R0,R8 REPEAT SSVTBASE (MVCL SOURCE)
SLR R1,R1 SIZE OF TARGET(ZERO=>FILL CHAR ONLY)
MVCL R14,R0 CLEAR SSVT AREA

MVI SSVTCMDS,1 PREDEFINED CMD BROADCAST FUNCT FIELD
MVI SSVTFNUM+1,1 SET THE NUMBR OF SUPPORTED FUNCTIONS
EJECT

BAS R10,CPLOADSS FETCH SUBSYSTEM FUNCTION ROUTINE

L R0,TABLSP COMMAND TABLE SUBPOOL AND LENGTH
LA R2,245 SET NUMBER OF SUBPOOL'S STORAGE
BAS R10,CPSTORA ALLOCATE STORAGE FOR COMMAND TABLE

ST R1,SSVTANKR STOW ITS AD INTO SSVT USER EXTENTION
LR R5,R1 RETAIN TABLE ADDRESS
XC Ø(TABPRE,R5),Ø(R5) ZERO TABLE PREFIX

SLR R0,R0 INITIALIZATION VALUE
LA R1,ENTNUM NUMBER ENTRIES IN TABLE
LA R4,TABPRE(),R5 LOAD FIRST ENTRY ADDRESS

DCINTLUP ST R0,4(R5) INITIALIZE PREFIX OR ENTRY
ST R4,Ø(),R5 POINTERS
LR R5,R4 TRANSFER ENTRY ADDRESS
LA R4,ENTLEN(),R4 LOAD NEXT ENTRY ADDRESS AND
BCT R1,DCINTLUP CONTINUE TABLE INITIALIZATION

ST R0,4(R5) INITIALIZE LAST
ST R0,Ø(),R5 TABLE ENTRY.
EJECT

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

* LOAD THE SUBSYSTEM FUNCTION ROUTINE INTO FIXED COMMON VIRTUAL *
* STORAGE. FORMAT THE COMMAND TABLE FOR CONSOLE OPERATORS.       *
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SPACE 1

BAS R10,CPLOADSS FETCH SUBSYSTEM FUNCTION ROUTINE

SPACE 1

L R0,TABLSP COMMAND TABLE SUBPOOL AND LENGTH
LA R2,245 SET NUMBER OF SUBPOOL'S STORAGE
BAS R10,CPSTORA ALLOCATE STORAGE FOR COMMAND TABLE

SPACE 1

ST R1,SSVTANKR STOW ITS AD INTO SSVT USER EXTENTION
LR R5,R1 RETAIN TABLE ADDRESS
XC Ø(TABPRE,R5),Ø(R5) ZERO TABLE PREFIX

SPACE 1

SLR R0,R0 INITIALIZATION VALUE
LA R1,ENTNUM NUMBER ENTRIES IN TABLE
LA R4,TABPRE(),R5 LOAD FIRST ENTRY ADDRESS

SPACE 1

DCINTLUP ST R0,4(R5) INITIALIZE PREFIX OR ENTRY
ST R4,Ø(),R5 POINTERS
LR R5,R4 TRANSFER ENTRY ADDRESS
LA R4,ENTLEN(),R4 LOAD NEXT ENTRY ADDRESS AND
BCT R1,DCINTLUP CONTINUE TABLE INITIALIZATION

SPACE 1

ST R0,4(R5) INITIALIZE LAST
ST R0,Ø(),R5 TABLE ENTRY.
EJECT

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

* ANNOUNCE THAT DCIPLES IS PREPARED TO ACCEPT VALID COMMANDS.    *
* ACTIVATE THE SUBSYSTEM. CHAIN FROM THE CVT TO THE JESCT TO       *
* THE SSCVT CHAIN AND LOCATE THIS SUBSYSTEMS SSCVT.                *
* STORE THE ADDRESS OF THE SSVT INTO THIS SUBSYSTEM'S SSCVT.       *

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

WTO 'DCIPL01A INITIALIZATION COMPLETE: READY TO PROCESS COMMA ANDS'

L R1, CVTPTR
   FETCH POINTER TO COMM VECTOR TABLE
   USING CVTMAP,R1
   ESTABLISH CVT ADDRESSABILITY

L R1, CVTJESCT
   FETCH POINTER TO JES2 CONTROL TABLE
   USING JESCT,R1
   ESTABLISH JESCT ADDRESSABILITY

ICM R6,15, JESSSCT
   ADDR OF FIRST SUBSYSTEM COMM TABLE
DCLOCVT BZ DCABORT
   BRANCH IF END OF SSCVT CHAIN.
CLC SSCTSNAME,PATNAME
   TEST IF THIS SUBSYSTEM'S SSCVT
BE DCFNDCVT
   BRANCH IF SO
ICM R6,15, SSCTSCTA
   ELSE FETCH ADDRESS OF NEXT SSCVT
B DCLOCVT
   AND TRY AGAIN.

DCFNDCVT ST R8, SSCTSSVT
   STOW SSVT ADDRESS IN SSCVT.

DCSTOID MVC SSVTASCB, PSAAOLD
   PUT CURRENT ASCB ADDR INTO SSVT EXT.
MVI SSVTCMDQ, 'C'
   PUT COMMAND ID INTO SSVT EXT.
B DCNOWAIT
   AND BYPASS WAIT.

TITLE 'SUBSYSTEM ADDRESS SPACE MAINLINE CODE.'

DCWAIT XC SSVTECB, SSVTECB
   RESET ECB AND
WAIT 1, ECB= SSVTECB
   TARRY AWHILE...

DCNOWAIT L R2, SSVTANKR
   LOAD COMMAND TABLE POINTER AND
ICM R6,15,4(R2)
   FETCH ALLOCATED QUEUE
BZ DCWAIT
   BRANCH IF NOTHING TO PROCESS

DCLOCKTB SR R3, R3
   CLEAR COMPARE REGISTER
LA R5, 256
   SET REPLACEMENT VALUE
CS R3, R5, 8(R2)
   OBTAIN EXCLUSIVE CONTROL
BNE DCLOCKTB
   OF THE COMMAND TABLE
MVC COMMDWK, 0(R6)
   MOVE COMMAND FOR PROCESSING

************************************************
* AWAIT ACTIVATION BY THE SERVICE REQUEST ROUTINE. *
* WHEN ACTIVATED:                                 *
* OBTAIN EXCLUSIVE CONTROL OF THE COMMAND TABLE,  *
* DECHAIN THE OPERATOR COMMAND ENTERED, AND       *
* RELEASE CONTROL OF THE COMMAND TABLE.           *
************************************************

DCNOWAIT L R2, SSVTANKR
   LOAD COMMAND TABLE POINTER AND
ICM R6,15,4(R2)
   FETCH ALLOCATED QUEUE
BZ DCWAIT
   BRANCH IF NOTHING TO PROCESS

DCLOCKTB SR R3, R3
   CLEAR COMPARE REGISTER
LA R5, 256
   SET REPLACEMENT VALUE
CS R3, R5, 8(R2)
   OBTAIN EXCLUSIVE CONTROL
BNE DCLOCKTB
   OF THE COMMAND TABLE
MVC COMMDWK, 0(R6)
   MOVE COMMAND FOR PROCESSING

************************************************
* COMMDWK NOW CONTAINS AN ENTRY FROM THE COMMAND TABLE FOR THE *
* CURRENT COMMAND THAT HAS BEEN FORMATTED AS FOLLOWS:          *
* +Ø CHAINING FIELD
* +4 CONSOLE ID
* +8 COMMAND (MINUS
* SUBSYSTEM IDENTIFIER)

***********************************************************************
SPACE 1
MVC CMDSYSID(4),SYSID INSERT SYSTEM ID
MVC 4(R2),Ø(R6) MAKE NEXT ENTRY FIRST ENTRY.
ICM R1,15,Ø(R2) POINT TO FIRST FREE ENTRY - IF ANY.
BZ DCLRQPTR BRANCH IF NONE.
ST R1,Ø(.R6) CHAIN FREE ENTRIES TO FREED ENTRY.
B DCHAIN CONTINUE...

SPACE 1
DCLRQPTR XC 0(4,R6),Ø(R6) SHOW END OF FREE QUEUE CHAIN
DCHAIN ST R6,0(.R2) MAKE FREED ENTRY FIRST ON FREE CHAIN
XC 8(4,R2),8(R2) RESET COMMAND TABLE LOCK WORK
SPACE 2
DROP R1,R6,R8 FORGET JESCT, SSCT AND SSVT
TITLE ' PROCESS THE OPERATOR COMMAND.'
***********************************************************************
SPACE 1
WTO 'DCIPLØ2I COMMAND WAS RECEIVED AND PROCESSED'
EJECT

***********************************************************************
SPACE 1
L RØ,CMDCONID SET CONSOLE ID
LA R1,ENTLEN+4 LENGTH OF MESSAGE + MCS
STH R1,WTOMSG SET LENGTH OF MESSAGE IN PARM LIST
SPACE 1
MVC WTOMSG+2(2),=X'EØØØ' SET MCSFLAGS.
* 8-CONSOLE ID IS IN REGISTER Ø
* 4-WTO IS A COMMAND RESPONSE
* 2-PRIMARY SUBSYSTEM CAN'T MODIFY MESSAGE

SPACE 1
MVC COMMNDWK(8),PATNAME SET REPLY PREFIX
SPACE 1
WTO MF=(E,WTOMSG) ECHO ECHO ECHO OPERATOR COMMAND
EJECT
**TEST FOR ENTRY OF A VALID COMMAND**

SPACE

CLC CMDTEXT(5),=C'Y2KUP' TEST IF YMM WANTS NET UP
BE CHKY2KID BRANCH IF SO.
SPACE 1

CLC CMDTEXT(5),=C'TECUP' TEST IF TECH WANTS NET UP
BE CHKTECID BRANCH IF SO.
SPACE 1

CLC CMDTEXT(7),=C'TECDOWN' TEST IF TECH WANTS QUICK NET
BE TECDOWN SHUTDOWN IF SO DO IT...
SPACE 1

CLC CMDTEXT(5),=C'DEVUP' TEST IF DEV WANTS NET UP
BE CHKDEVID BRANCH IF SO.
SPACE 1

CLC CMDTEXT(7),=C'DEVDOWN' TEST IF DEV WANTS QUICK NET
BE DEVDOWN SHUTDOWN IF SO DO IT...
SPACE 1

CLC CMDTEXT(5),=C'ACCUP' TEST IF ACCENT WANTS NET UP
BE CHKACCID BRANCH IF SO.
SPACE 1

CLC CMDTEXT(7),=C'ACCDOWN' TEST IF ACCENT WANTS QUICK NET
BE ACCDOWN SHUTDOWN IF SO DO IT...
SPACE 1

CLC CMDTEXT(5),=C'PROUP' TEST IF PRODUCTION WANTS NET UP
BE CHKPROID BRANCH IF SO.
SPACE 1

CLC CMDTEXT(7),=C'PRODOWN' TEST IF PRODUCTION WANTS QUICK
BE PRODOWN SHUTDOWN IF SO DO IT...
SPACE 1

CLC CMDTEXT(3),=C'PAP' TEST IF SYSTEM IS TO BE IPL'ED
BE PATDRAIN BRANCH IF SO.
SPACE 1

CLC CMDTEXT(4),=C'WARN' TEST IF BROADCAST COMMAND ENTERED
BE PATWARN BRANCH IF IT IS
SPACE 1

CLC CMDTEXT(8),=C'VERIFYUP' TEST IF VERIFY PRODUCTS UP
BE CPVERUP BRANCH IF IT IS
CLC CMDTEXT(8),=C'VERIFYDW' TEST IF VERIFY PRODUCTS DOWN
BE CPVERDWN BRANCH IF IT IS
SPACE 1

CLC CMDTEXT(7),=C'CONNECT' CONNECT PRODUCTS
BE CONNECT BRANCH IF IT IS
CLC CMDTEXT(4),=C'CHKEY' TEST IF CHKEY DESIRED
BE CPCHKEY BRANCH IF IT IS
SPACE 1

CLC CMDTEXT(4),=C'HALT' TEST IF BROADCAST COMMAND ENTERED
BE DCAHABORT CLEAN UP AND TERMINATE
* INFORM OPERATOR OF HIS FINGER-CHECK *

SPACE 1

CPWTOERR MVC CMDTEXT(35),=CL35'COMMAND IS INVALID FOR THIS DOMAIN'
WTO MF=(E,WTOMSG) ISSUE AN ERROR MESSAGE
WTO 'DCIPLØ3A IS READY FOR YOUR VALID COMMAND'
B DCNOWAIT AND PROCESS NEXT COMMAND

SPACE 2

* ENSURE THAT COMMAND IS A VALID ONE FOR THIS SYSTEM *

SPACE 1

CHKY2KID CLC CMDSYSID(4),=C'VSØ2' IS THIS THE CORRECT DOMAIN
BE PPGFNDSY ASCERTAIN AVAILABILITY OF 3745S
B CPWTOERR ENTER COMMON CODE

SPACE 1

CHKTECID CLC CMDSYSID(4),=C'VSØ3' IS THIS THE CORRECT ID
BE PPGFNDSY SEE IF 3745S UP
B CPWTOERR OTHERWISE DO NOTHING

SPACE 1

CHKDEVID CLC CMDSYSID(4),=C'VSØ1' IS THIS THE CORRECT ID
BE PPGFNDSY SEE IF 3745S UP
B CPWTOERR OTHERWISE DO NOTHING

SPACE 1

CHKACCID CLC CMDSYSID(4),=C'VSØ4' IS THIS THE CORRECT ID
BE PPGFNDSY SEE IF 3745S UP
B CPWTOERR OTHERWISE DO NOTHING

SPACE 1

CHKPROID CLC CMDSYSID(4),=C'VSØ5' IS THIS THE CORRECT ID
BE PPGFNDSY SEE IF 3745S UP
B CPWTOERR OTHERWISE DO NOTHING

EJECT

* ASSUME ALL 3745'S ARE OFFLINE THEN REPUDIATE THAT ASSUMPTION *

SPACE

PPGFNDSY L R1,CLAMHOLD GET ADDRES OF THIS SYSTEM'S ENTRIES
L R1,4(R1) ADDRESS OF 37X5S FOR THIS SYSTEM
SCLAGAIN CLI Ø(R1),C' ' TEST END OF ENTRIES
BE CHK3745 AT END, PROCEED TO CHECK NET
MVI 2(R1),C'N' SET DEVICE OFFLINE
LA R1,PPGUCBLN(R1) POINT TO NEXT ENTRY
B SCLAGAIN AND CONTINUE TO INITIALIZE ENTRIES

SPACE 1

CHK3745 DS ØH
SR R4,R4 CLEAR PATH CONTROL REGISTER

SPACE 1

MVC WTORWTOR(PPGLENWR),PPGWTOR WTOR PATRN INTO ACQUIRED AREA
MVC WTORSVC(PPGSVCPL),SVCPARM SVC COMMAND INTO ACQUIRED AREA
SPACE 1

BAS R5, PPGETUCB ATTEMPT TO PLACE ALL 37X5S ON-LINE
BAS R10, PATREST PAUSE FOR THE CAUSE
LA R4, I ALTER PATH CONTROL TO QUERY OPERATOR
BAS R5, PPGETUCB ASCERTAIN AVAILABILITY OF ALL 37X5S
SPACE

L R1, CLAMHOLD ADDRESS OF THIS SYSTEM’S ENTRIES
L R1.4(R1) ADDRESS OF 37X5S FOR THIS SYSTEM
NJTAGAIN CLI Ø(R1), C' ' TEST IF END OF ENTRIES
BE NJTCNT AT END, PROCEED TO CHECK NET
CLI 2(R1), C'O' TEST IF DEVICE IS ON-LINE
BNE NJTQUIET QUERY OPERATOR FOR TERMINATION OPTION
LA R1, PPGUBLN(R1) POINT TO NEXT ENTRY
B NJTAGAIN AND TRY ONCE AGAIN TO LOCATE UCB

***********************************************************************
* DETERMINE IF ALL 37X5S ARE ON-LINE. VARY ALL OFF-LINE 37X5S       *
* ON-LINE AND WAIT FOR PROCESS TO COMPLETE. AFTERWARDS, RESCAN        *
* 37X5 CHAIN FOR OFF-LINE 37X5S. IF ANY REMAIN, QUERY OPERATOR       *
* ALLOWING HIM TO DECIDE IF ACTIVATION PROCESS SHOULD CONTINUE OR    *
* TERMINATE.                                                         *
***********************************************************************
SPACE 1
PPGETUCB UCBINFO DEVCOUNT,COUNT=PPGCOUNT,DEVCLASS=COMM,PLISTVER=MAX
SPACE 2
LTR R15, R15 TEST IF SUCCESSFUL
BE PPGLSCP BRANCH IF SO
EJECT

***********************************************************************
* PROCESS ERRORS AND TERMINATE                                       *
***********************************************************************
SPACE 1
PPGDEVER ST R15, PPGDOUBL STOW RETURN CODE
UNPK PPGRETC, PPGDOUBL+3(2) CONVERT TO PACKED DECIMAL
MVI PPGRETC+2, C' ' REMOVE DE TRASH
TR PPGRETC(2), PATRANS-24Ø CONVERT TO EBCDIC
SPACE 1
ST RØ, PPGDOUBL STOW RETURN CODE
UNPK PPGREAC, PPGDOUBL+3(2) CONVERT TO PACKED DECIMAL
MVI PPGREAC+2, C' ' REMOVE DE TRASH
TR PPGREAC(2), PATRANS-24Ø CONVERT TO EBCDIC
SPACE 1
MVC COMMNDWK(43), PPGERMSG MOVE IN VERIFICATION FAIL FORMAT
LA R1, 50+4 LENGTH OF EACH COMMAND PLUS CONSTANT
STH R1, WTMOSG SET LENGTH IN INTERNAL COMMAND
WTO MF=(E, WORK), DESC=2, ROUTCDE=8 FOR ERROR CONDITIONS, RETURN TO DUST
B DCAABORT

***********************************************************************
* SET ITERATION COUNT AND POINT TO AREA FOR RETURN OF A COPY          *
***********************************************************************
* OF A TELECOMMUNICATION'S UNIT CONTROL BLOCK *

******************************************************************************
SPACE 1
PPGLSCP ICM R3,15,PPGCOUNT NUMBER OF UCB'S WITHIN DEVICE CLASS
BE PPGDEVER BRANCH IF NONE
LA R2,PPGUCB CONTAINMENT AREA ADDRESS FOR UCB DATA

*A AVOID RETURN CODE OF 4 WITH A REASON CODE OF 1*
MVI PPGWORK,Ø ERASE FIRST BYTE OF WORK AREA
MVC PPGWORK+1(99),PPGWORK ERASE THE REMAINING PORTION OF IT
EJECT
******************************************************************************

* PROCESS ONLY THE 37X5S THAT ARE REQUIRED ON THIS SYSTEM *
******************************************************************************
SPACE
PATFINDV DS ØH
UCBSCAN COPY,WORKAREA=PPGWORK,UCBAREA=PPGUCB,RANGE=ALL,
DEVCLASS=COMM,PLISTVER=MAX,DYNAMIC=YES
SPACE 1
LTR R15,R15 TEST IF SUCCESSFUL
BNE PPGDEVER BRANCH IF NOT
SPACE
L R1Ø,CLAMHOLD ADDRESS OF THIS SYSTEM'S REQUIREMENTS
USING UCBCMSEG,R2 SET UCB ADDRESSABILITY
CLI UCBBYT4,X'25' TEST IF 37X5 ON 5.2
BNE PPGNXUCB BRANCH IF NOT
SPACE 2
PPGISX25 L R1Ø,4(R1Ø) ADDRESS OF 37X5S FOR THIS SYSTEM
PPGAGAIN CLI Ø(R1Ø),C' ' TEST IF END OF ENTRIES
BE PPGNXUCB IF SO, SKIP THIS UCB
CLC UCBCHAN,Ø(R1Ø) TEST IF ENTRY MATCHES THIS UCB
BE PPGDOUCB PROCESS THIS UCB
LA R1Ø,PPGUCBLN(R1Ø) POINT TO NEXT ENTRY
B PPGAGAIN AND TRY ONCE AGAIN TO LOCATE UCB
EJECT
******************************************************************************

* ASCERTAIN AVAILABILITY OF REQUIRED UNIT(S) *
******************************************************************************
SPACE
PPGDOUCB UNPK WTORWTOR+9+4+4+4+7(5),UCBCHAN(3) STOW NAME OF DEVICE
MVI WTORWTOR+9+4+4+4+7+4,C' ' CLEAR DE TRASH FROM MESSAGE
MVC WTORCMG+2(4),WTORWTOR+9+4+4+4+7 DEVICE # TO KOMAND AREA
SPACE

* TM UCBFLA,UCBNRY TEST IF DEVICE IS IN READY
* BO PPGOPCP BRANCH IF SO
SPACE

* LTR R4,R4 TEST IF VARY ONLINE'S HAVE BEEN DONE
* BNE PPGDOWTO BRANCH IF SO
SPACE
PPGOPCP TM UCBFLB,UCBNOPTH+UCBNOCON TEST IF PATH IS OPERATIONAL
BZ PATOP BRANCH IF VALID
LTR R4,R4 TEST IF FIRST PASS
BNE PPGDOWTO BRANCH IF NOT
SPACE

PATOP TM UCBFL1,UCBPERM TEST IF SUBCHANNEL IS USABLE
BZ PATUSE BRANCH IF VALID
LTR R4,R4 TEST IF FIRST PASS
BNE PPGDOWTO BRANCH IF NOT
SPACE 1

PATUSE TM UCBSTAT,UCBONLI TEST IF DEVICE IS ONLINE
BO NTONLINE BRANCH IF SO
SPACE

LTR R4,R4 TEST IF VARY ONLINE'S HAVE BEEN DONE
BNE PPGDOWTO BRANCH IF SO
OI CLAMHOLD,X'80' SET DEVICE OFFLINE
SPACE 1
SR 0,0 CONSOLE ID
LA R1,WTORSVC POINT TO COMMAND
SVC 34 AND THEN ISSUE IT
SPACE 2

PPGNXUCB BCT R3,PATFINDV LOOP POWER≥
1 B Ø(R5)
SPACE

NTONLINE MVI 2(R10),C'O' INDICATE DEVICE IS ON-LINE
B PPGNXUCB CONTINUE...
DROP R2 FORGET UCB
EJECT
***********************************************************************
*    CONSTRUCT A WTOR; ALLOW AN OPERATOR TO CONFIRM CONTINUATION OF   *
*    ACTIVATION PROCESS WHENEVER A 37X5 IS FOUND TO BE OFF-LINE.      *
***********************************************************************
SPACE

PPGDOWTO DS ØH
BAS R14,CLAMINIT FORMAT WTOR
OI CLAMHOLD,X'80' SET DEVICE OFFLINE
WTOR MF=(E,(1)) QUERY OPERATOR
SPACE
WAIT ECB=WTORECB TARRY WHILE OPERATOR COGITATES
SPACE
CLI WTORECB,C'Y' SHOULD ACTIVATION CONTINUE?
BE NTONLINE IF SO, SET ONLINE; CONTINUE WITH IT
SPACE
WTO 'DCIPLØ4A PLACE REQUIRED DEVICES ON-LINE; RESTART DCIPLES'
B DCABORT ENTER WORTHLESS CODE
SPACE

CLAMINIT MVI WTORECB,C' ' CLEAR ANSWER
MVI WTORWTOR+8,1 LENGTH OF REPLY
LA R1,WTORWTO POINT TO RESPONSE AREA
ST R1,WTORWTO STOW IT IN PARAMETER AREA
OI WTORWTO,X'80'
LA R1,WTORECB  POINT TO ECB FOR WTOR
ST R1,WTORWTOR+4  STOW IT INTO PARAMETER AREA
LA R1,WTORWTOR  POINT TO WTOR ITSELF
BR R14  RETURN TO CALLER
EJECT

***********************************************************************
*        CHECK TO SEE IF NET IS UP (IF NOT TERMINATE DCIPLES)         *
***********************************************************************

SPACE 1
NJTCOUNT L R2,PSAATCVT  POINT TO VTAM'S CVT
TM ATCSTAT1(R2),ATCACTIV TEST IF VTAM IS ACTIVE
BNO BEFORNET IF NOT, ACTIVATE THE NETWORK
SPACE 1
TM CLAMHOLD,X'80' TEST FOR ALL 37X5S AVAIL AT STARTUP
BNO CHKNET IF SO, CONTINUE
SPACE 1
NJTQUIET MVC WTORWTOR(PPGLENWD),PPGWTDOR WTOR PATN INTO ACQUIRED AREA
BAS R14,CLAMINIT FORMAT WTOR
SPACE 1
WTOR MF=(E,(1)) QUERY OPERATOR
SPACE
WAIT ECB=WTORECB TARRY WHILE OPERATOR COGITATES
NI CLAMHOLD,255-X'80' RESET SWITCH
SPACE
CLI WTORANS,C'Y' SHOULD ACTIVATION CONTINUE?
BE CHKNET IF SO, CONTINUE TO START PRODUCTS
SPACE
WTO 'DCIP02A ENSURE ALL 37X5S ARE ON-LINE; RESTART NET FIRST
T THEN RESTART DCIPLES'
B DCABORT ENTER WORTHLESS CODE
SPACE 1

CHKNET DS 0H
L R2,PSAATCVT  POINT TO VTAM'S CVT
TM ATCSTAT1(R2),ATCACTIV TEST IF VTAM IS ACTIVE
BO AFTERNET IF SO CONTINUE TO START PRODUCTS
EJECT

***********************************************************************
*        PREPARE DATA SET FOR USE                                    *
***********************************************************************

SPACE 1
BEFORNET LA R8,PATDCB  POINT TO DCB
USING IHADCB,R8  ESTABLISH DCB ADDRESSABILITY
SPACE 1
OPEN (PATDCB)  PREPARE DATASET FOR ACCESS
SPACE 1
LH R0,=H'80' SET DEFAULT FOR GETMAIN
SR R2,R2 ACQUIRE STORAGE FROM SUBPOOL ZERO
TM DCBOFLGS,DCBOFOPN TEST IF DATASET OPENED SUCCESSFULLY
BNO PATMAIN BRANCH IF SO
SPACE 1

**LH R0,DCBBKSI**
*Fetch block-size of data set*

**PATMAIN BAS R1,CPSTORA**
*Obtain an I/O area*

**LR R5,R1**
*Save area address*

**SPACE 1**

**TM DCBOFLGS,DCBOFOPN**
*Test if dataset opened successfully*

**BNO PATOPER**
*Query MTO if not*

**EJECT**

*****************************
* Obtain the current argument for a START-NET command*
*****************************

**SPACE 1**

**BLDL PATDCB,LISTADDR**
*Find current list for (LIST=XX) parm*

**LTR R15,R15**
*Test return code*

**BNE PATOPER**
*Branch if error*

**SPACE 1**

**POINT PATDCB,BLOCKADR**
*Point to start of procedure*

**SPACE 1**

**READ DECB1,SF,PATDCB,(5),'S'**
*Fetch list argument*

**CHECK DECB1**
*Wait for completion of I/O*

**SPACE 1**

**Patretrx CLC Ø(3,R5),A1**
*Test if valid parameter*

**BE PATSETL**
*Branch if so*

**CLC Ø(3,R5),A4**
*Test if valid parameter*

**BE PATSETL**
*Branch if so*

**EJECT**

*****************************
* Query MTO for valid list option*
*****************************

**SPACE 1**

**PATOPER WTOR 'LIST ARGUMENT FOR NET IS INVALID, SPECIFY A1 OR A4', X (5),2,PATECB**
*Tarry-a-bit*

**WAIT ECB=PATECB**
*Remove trash from answer area*

**MVI 2(R5),C' '**
*Reset ECB*

**B PATRETRX**
*Try again*

**SPACE 1**

*****************************
* Build a command to initiate telecommunications processing*
*****************************

**SPACE 1**

**PATSETL ICM R1,3,Ø(R5)**
*Fetch list option for start command*

**MVC Ø(PATSLEN,R5),PATSNET**
*Move start command into work area*

**STCM R1,3,PATSARG(R5)**
*Stow list option in start command*

**EJECT**

*****************************
* Issue a command to initiate telecommunications processing*
*****************************

**SPACE 1**

**SR R0,R0**
*Clear console identification*

**STH R0,2(R5)**
*Clear garbage*
LR R1,R5  POINT TO OPERATOR COMMAND  
SVC 34  RESURRECT THE STATE'S TP NETWORK  
SPACE 1  
SR R3,R3  ZERO LOOP COUNTER  

PATNAP  
STIMER WAIT,BINTVL=CLAMTIMX SLEEP...  
LA R3,1(R3)  INCREMENT LOOP COUNTER  
CH R3,=H'18Ø'  TEST IF EXCEEDS 3 MINUTES  
BH SOMETING  EXIT IF SO  
ICM R2,15,PSAATCVT  PICK UP ADR OF VTAM ACTIVE BIT  
BE PATNAP  IF NOT THERE THEN WAIT FOR IT  
SPACE 1  
TM ATCSTTI(R2),ATCACTIV TEST IF VTAM IS ACTIVE  
BNO PATNAP  IF NOT, WAIT  
SPACE 1  

SOMETING  
LH R0,=H'8Ø'  SET DEFAULT FOR FREEMAIN  
TM DCBOFLGS,DCBOFOPN  TEST IF DATASET OPENED SUCCESSFULLY  
BNO PATFREE  IF NOT, FREE DEFAULT  
LH R0,DCBBLSI  SIZE OF GETMAINED AREA  

PATFREE  
SR R2,R2  LOAD SSCT SUBPOOL AND  
LR R1,R5  ADDRESS AND  
BAL R1Ø,CPSTORF  THEN RELEASE I/O AREA  
SPACE  
TM DCBOFLGS,DCBOFOPN  TEST IF DATASET OPENED SUCCESSFULLY  
BNO NANCYJT  IF NOT, SKIP CLOSE  
CLOSE (PATDCB)  DONE WITH DATASET  
SPACE 2  

NANCYJT  
ICM R2,15,PSAATCVT  PICK UP ADR OF VTAM ACTIVE BIT  
BE DCABORT  IF NOT THERE THEN CLEAN UP/TERMINATE  
TM ATCSTTI(R2),ATCACTIV MAKE SURE VTAM IS ACTIVE  
BNO DCABORT  IF NOT, CLEAN UP AND TERMINATE  
DROP R0,R8  DROP R0 R8  
BAS R1Ø,PATREST  

*******************************************************************************  
* ENSURE THAT ALL NCPS ARE ACTIVE BEFORE CONTINUING *  
*******************************************************************************  

SPACE  
AFTERNET ESAR R1  GET SECONDARY ASID  
ST R1,PPHCASID  SAVE CURRENT SECONDARY ASID  
SPACE  
*******************************************************************************  
* LOCATE NET *  
*******************************************************************************  

SPACE 1  
PPHACTIV BAS R1Ø,PATREST  PAUSE FOR THE CAUSE  
SPACE 1  
USING PSA,R0  ESTABLISH PSA ADDRESSABILITY  
L R3,CVTPTR  ADDRESS OF CVT  
USING CVT,R3  ESTABLISH CVT ADDRESSABILITY  
SPACE 1  

L     R5, CVTASVT          FETCH ADDRESS OF ASVT
DROP  R3                  FORGET CVT
SPACE 1
USING  ASVT, R5
L     R4, ASVTMAXU        MAXIMUM NUMBER OF ADDRESS SPACES
SPACE 1
PPGLOC  TM  ASVTENTY, ASVTAVAL  TEST IF ENTRY IS AVAILABLE
B0    PPGGRUVE            BRANCH IF SO
SPACE 1
L     R6, ASVTENTY       RETRIEVE ADDRESS OF ASCB
USING ASCB, R6          ESTABLISH ASCB ADDRESSABILITY
SPACE 1
ICM  R1, 15, ASCBJBNI    POINTER TO INITIATED JOBNAME
BZ    PPGJJNI
SPACE 1
CLC  Ø(8, R1), PPHJNAME  TEST IF CORRECT JOB
BNE   PPGGRUVE
B     PPGGOTIT
SPACE 1
PPGJJNI  EQU  *
SPACE 1
ICM  R1, 15, ASCBJBNS    POINTER TO START/MOUNT/LOGON TASK
BZ    PPGGRUVE            FORMAT IT
SPACE 1
CLC  Ø(8, R1), PPHJNAME  TEST IF CORRECT JOB
BE    PPGGOTIT            BRANCH IF SO
SPACE 1
PPGGRUVE  LA    R5, 4(R5)   NEXT ENTRY
BCT   R4, PPGLOC          LOOP POWER
B     DCABORT             ISSUE ERROR MESSAGE; RETURN TO DUST
EJECT
***********************************************************************
*        ACCESS CONTROL BLOCKS IN VTAM'S ADDRESS SPACE ENSURING       *
*        THAT ALL KNOWN NCPS ARE ACTIVE                              *
***********************************************************************
SPACE 1
PPGGOTIT  LH    R2, ASCBASID          OBTAIN ASID OF VTAM'S ADDRESS SPACE
SPACE
LAM   R4, R4, PPHONE         INITIALIZE ACCESS REGISTER
LAM   R5, R5, PPHONE         INITIALIZE ACCESS REGISTER
SPACE 1
LA    R1, 1                SET AUTHORIZATION
AXSET AX=(R1)               INDEX TO ONE
SSAR  R2                  USE DATA IN ADDRESS SPACE OF VTAM
SAC   512                 UNIVERSAL ACCESS MODE
SPACE 1
L    R1, =A(PPH31AMD+X'80000000')  SET BRANCH ADDRESS
BSM   R0, R1               ENTER 31-BIT AMODE
EJECT
PPH31AMD  L    R1, PSAATCVT     ADDRESS OF VTAM'S VECTOR TABLE
L R4,ATCCONFT(R1) ADDRESS OF VTAM CONFIGURATION TABLE
L R5,CONVTHAA(R4) ADDRESS OF VTAM RDT HEADER AREA

SPACE 1

PPHFORM TM RDTPLEN(R5),RDTPRIRN TEST IF NCP SEGMENT
BNO PPHGO
SPACE
CLI RPRCURS1(R5),FSMCDATA TEST IF NCP IS ACTIVE
BNE PPHNNOGO BRANCH IF NOT TO ISSUE MSG AND WAIT
SPACE 1

PPHGO ICM R5,15,RDTFORW(R5) ADDRESS OF NEXT CDRM RDTE
BNE PPHFORM

SPACE 1

BAL R2,PPHSACØ RETURN TO NORMAL SPACE
LA R1,POSTNET SET BRANCH ADDRESS
BSM RØ,R1 ENTER 24-BIT AMODE
SPACE 1

PPHNNOGO BAL R2,PPHSACØ RETURN TO NORMAL SPACE
LA R1,PPHWTO5I SET BRANCH ADDRESS
BSM RØ,R1 ENTER 24-BIT AMODE
SPACE 1

PPHWTO5I WTO 'DCIPLØ5A DCIPLES IS AWAITING ACTIVATION OF ALL NCPS'
B PPHACTIV TARRY AWHILE...
SPACE 2

PPHSACØ  L R1,PPHCASID OBTAIN ACTUAL SECONDARY ASID
SSAR R1 SET SECONDARY TO CURRENT
LAM R4,R4,PPHZERO CLEAR ACCESS REGISTER
LAM R5,R5,PPHZERO CLEAR ACCESS REGISTER
SAC Ø ACCESS DATA ONLY WITHIN THIS ASID
BR R2 RETURN TO CALLER
EJECT

***********************************************************************
* CHECK FOR SYSTEM TO BE BROUGHT UP AFTER NETWORK START            *
***********************************************************************

SPACE 1

POSTNET  CLC CMDTEXT(5),=C'TECUP' TEST IF TECH WANTS NET UP
BE TECUP BRANCH IF SO.
SPACE 1

CLC CMDTEXT(5),=C'DEVUP' TEST IF DEV WANTS NET UP
BE DEVUP BRANCH IF SO.
SPACE 1

CLC CMDTEXT(5),=C'Y2KUP' TEST IF Y2K wants NET UP
BE YMMUP BRANCH IF SO.
SPACE 1

CLC CMDTEXT(5),=C'ACCUP' TEST IF ACCENT WANTS NET UP
BE ACCUP BRANCH IF SO.
SPACE 1

CLC CMDTEXT(5),=C'PROUP' TEST IF PRODUCTION WANTS NET UP
BE PROUP BRANCH IF SO.
SPACE 1

Systems Programmer (USA) © Xephon 1999

IBM has extended its S/390 usage pricing to level D, targeting customers with growing new workloads on single or sysplex systems, and those looking to consolidate on to larger processors.

Effective at the start of July, usage pricing level D, reckons IBM, will result in about 25% lower charges than level C with workloads greater than 78 MSUs.

The scheme applies to OS/390 Version 2 and recent versions and releases of IMS, DB2, CICS, and MQSeries, all of which now have usage measurement technology built in. Reports are required once a year, so budgeting should be more predictable and administrative tasks reduced.

The move will be generally welcomed, but it’s true to say that it is not a great leap forward. The problem is that while usage pricing for all software makes a good theoretical case, no one, whether vendor or user, wants the responsibility of managing it all. Clearly IBM wants to improve its pricing policy slowly and placate users who have been paying over the odds but, at the same time, wants to maintain its revenue stream.

On the application front, it’s difficult to gauge precisely how many mainframe sites will benefit from this. No doubt IMS users will generally see some returns, but without knowing how much capacity DB2, CICS, and MQSeries are taking at user sites, hard facts won’t emerge for some time. Initial indications suggest CICS sites may be the least likely to benefit.

***

NEON Systems this week released its Halo SSO single sign-on solution for mainframe applications in mixed OS/390-MVS and SNA Server/NT sites. All this is achieved, it’s claimed, without custom coding or installation of software at the desktop. Security administrators, says the firm, can manage OS/390-MVS/Windows security issues with automated password synchronization from a centralized, secure control point.

The product will virtually eliminate password recall inquiries to help desks in mixed sites, with users only having to remember one password, without having to log into the mainframe. Halo SSO single sign-on includes both the MVS and NT components, both of which continuously monitor mainframe or NT-initiated password changes and propagate password changes to the appropriate domains. It includes logging and monitoring capabilities, provides single sign-on benefits when used with application logon scripts, and supports RACF, ACF2, and Top Secret security systems.

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.
http://www.neonsys.com

***