181 MVS update

October 2001

In this issue

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

North American office
Xephon/QNA
PO Box 350100,
Westminster, CO 80035-0100
USA
Telephone: (303) 410 9344
Fax: (303) 438 0290

Contributions
Articles published in *MVS Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, you can download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

MVS Update on-line
Code from *MVS Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/mvs; you will need to supply a word from the printed issue.

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.

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 ($43.50) each including postage.

Printed in England.

Managing SMP/E SMPPTS out-of-space conditions

INTRODUCTION

The SMPPTS dataset is a partitioned dataset, and is used during SMP/E RECEIVE processing as storage for SYSMODs. Both Modification Control Statements (MCS) and any in-line data from the SYSMODs are stored in the SMPPTS dataset. The SMPPTS must be large enough to contain all the SYSMODs in the SMPPTFIN dataset that will be received. With a trend toward increasingly large PTFs, the size limits of the SMPPTS are being reached. More users are experiencing problems with their SMPPTS dataset running out of space. The SYSMODs currently stored in their SMPPTS along with the space required for incoming SYSMODs is exceeding one physical volume of space in some instances. However, the SMPPTS dataset is a partitioned dataset, and partitioned datasets are restricted to a single volume. This article will explain several ways to relieve this space issue. Specifically, it will address three areas:

- How to free space in your current SMPPTS dataset.
- How to prevent unnecessary SYSMODs from being received.
- Using SMPPTS spill datasets to contain all of your SYSMODs.

HOW TO FREE UP SPACE IN AN SMPPTS DATASET

The following is recommended to free up space in an SMPPTS dataset.

Allocate the SMPPTS as a PDSE

If SMS is active on your system, you can allocate the SMPPTS dataset as a PDSE. This will allow you to use the SMPPTS without needing to compress the dataset periodically to remove wasted space, which may occur over time as members are deleted from the dataset.

Use system-determined blocksize

Allocate the SMPPTS dataset using system-determined blocksize. Do this by specifying BLKSIZE=0. This will allow the system to create a dataset with an optimal blocksize for the device on which the dataset will reside.

REJECT PURGE processing

The SMP/E REJECT command allows you to clean up the global zone, SMPPTS, and associated entries and datasets. REJECT has four modes – MASS, SELECT, NOFMID, and PURGE modes.

In PURGE mode, SMP/E will reject all SYSMODs that have been accepted into the specified distribution zones.

PURGE mode can be used when SYSMODs were not automatically deleted once they were accepted. This will be the case if NOPURGE was coded in the OPTIONS entry used to process the distribution zone. SMP/E's default is to purge during ACCEPT processing. However, you may have added NOPURGE to the OPTIONS entry in order to prevent this. So, unless you have modified the OPTIONS entry to add NOPURGE, SMP/E automatically deletes global zone SYSMOD entries, HOLDDATA entries, SMPPTS MCS entries, and SMPTLIB datasets when SYSMODs are accepted.

If NOPURGE is being used in your OPTIONS entry, you may choose to run a REJECT PURGE command to clean up your SMPPTS dataset and global zone. Run this command by specifying the following:

```
SET BDY(GLOBAL).
REJECT PURGE (dlbzone1, dlbzone2, dlbzone3)
             HOLDDATA COMPRESS(SMPPTS).
```

where dlbzone1, dlbzone2, and dlbzone3 indicate which distribution zones (or ZONESETs) to check for applicability.

Since no SYSMOD types (APARs, FUNCTIONS, USERMODS) were specified on the above command, SMP/E will delete only PTFs (the default) from the global zone and SMPPTS. SYSMOD entries in the global zone and MCS entries in the SMPPTS dataset for applicable SYSMODs will be deleted.

The HOLDDATA operand on the above command indicates HOLDDATA entries associated with the SYSMOD entries being rejected should also be deleted from the global zone. If you have not allocated your SMPPTS dataset as a PDSE, you should use the COMPRESS operand to compress your SMPPTS dataset as shown above.

SMPPTS compaction

In OS/390 Version 2 Release 5 SMP/E introduced a new function called PTF Compaction in the SMPPTS dataset. SMP/E will compact in-line element data in SYSMODs in the SMPPTS dataset in order to reduce the storage requirements of the dataset.

If you are running OS/390 Version 2 Release 5 SMP/E or above, SMPPTS compaction is the default. In-line element data is automatically compacted during receive processing.

You can see the data has been compacted by browsing the SMPPTS dataset. Within the members of the SMPPTS dataset, compacted inline element data will start with the characters '$$GIMC'. The element data will be expanded as needed during APPLY and ACCEPT processing.

Compaction is controlled by the COMPACT subentry in the global zone OPTIONS entry. If you wish to change the COMPACT subentry, you may use UCLIN, or the SMP/E dialogs. Note that COMPACT(YES) is the default, so that compaction is automatically performed. To use UCLIN to change the value from NO to YES, use the following:

```
SET BDY(GLOBAL).
UCLIN.
  REP OPTIONS(GLOBOPT) COMPACT(YES).
ENDUCL.
```

PREVENTING UNNECESSARY SYSMODS FROM BEING RECEIVED

The following is recommended to prevent unnecessary SYSMODs from being received.

Enhanced RECEIVE processing

During SMP/E accept processing, PTFs will be purged from the SMPPTS dataset unless NOPURGE has been specified in the active OPTIONS entry. If, once a PTF has been purged from the SMPPTS dataset, it is found again during RECEIVE processing (from a CBPDO or ESO for example), it will be received again since it no longer exists in the SMPPTS dataset. However, there is really no need to RECEIVE the PTF again. So, SMP/E introduced Enhanced RECEIVE processing in OS/390 Version 2 Release 5 SMP/E.

Enhanced RECEIVE processing allows users to prevent the RECEIVE command from processing SYSMODs that are already applied or accepted. Users can specify this with the OPTIONS entry, on the RECEIVE command, or both. This enhancement reduces the need for the user to manage the SMPPTS with REJECT commands.

Enhanced RECEIVE – OPTIONS entry support

The most automated method of preventing SYSMODs from being received after they have already been applied or accepted is to define a Receive Zone Group in an OPTIONS entry and then make that OPTIONS entry active during RECEIVE command processing.

A Receive Zone Group is a list of zones and/or zonesets, which will be checked during RECEIVE processing. If a SYSMOD has been applied into a target zone which is part of the Receive Zone Group, or accepted into a distribution zone which is part of the Receive Zone Group, it will not be received into the SMPPTS dataset.

Zones and zonesets, that are part of a Receive Zone Group can contain both target and distribution zones.

For example, if you do not want PTFs received into the SMPPTS after they have been accepted into any of your distribution zones, you could define your distribution zones as a Receive Zone Group.

A Receive Zone Group can be defined by using the RECZGRP subentry of an OPTIONS entry. The following UCLIN can be used to set up the RECZGRP subentry in the RECOPT OPTIONS entry:

```
SET BDY(GLOBAL).
UCLIN.
  ADD OPTIONS(RECOPT) RECZGRP(dlbzone1,dlbzone2,dlbzone3).
ENDUCL.
```

Then, to make the RECOPT OPTIONS entry active during a RECEIVE command, use the following SET and RECEIVE command or add RECOPT as an OPTIONS subentry in the GLOBALZONE entry in the global zone.

```
SET BDY(GLOBAL) OPTIONS(RECOPT).
RECEIVE SYSMODS.
```

During RECEIVE processing, any SYSMODs that have already been accepted in zones dlbzone1, dlbzone2, or dlbzone3 are not selected to be processed by this RECEIVE command. These SYSMODs will not be received into the SMPPTS dataset.

In addition, 'ALLZONES' can be specified in the RECZGRP subentry to indicate all target and dlib zones defined by a ZONEINDEX subentry in the GLOBALZONE entry are eligible to be checked during RECEIVE command processing.

Enhanced RECEIVE – RECEIVE command support

If you do not want to make use of the OPTIONS entry method for enhanced RECEIVE, another way to achieve the same thing is to use the ZONEGROUP operand on your RECEIVE command, as follows:

```
SET BDY(GLOBAL).
RECEIVE ZONEGROUP(dlbzone1, dlbzone2, dlbzone3) SYSMODS.
```

This RECEIVE command will receive SYSMODs that have not been applied or accepted into any of the defined zones. If a SYSMOD has been applied or accepted into one of the zones, that SYSMOD will not be received into the SMPPTS. The ZONEGROUP can identify zones, zonesets, or both.

You can also specify 'ALLZONES', which uses all target and dlib zones listed in the ZONEINDEX subentry of the GLOBALZONE entry as the Receive Zone Group. The ZONEGROUP operand on the RECEIVE command will override any values specified on the RECZGRP and RECEXZGRP subentries in the OPTIONS entry.

USING SMPPTS SPILL DATASETS TO CONTAIN ALL SYSMODS

If after trying all the suggestions already mentioned, you still have not reclaimed enough space in your SMPPTS dataset, then you should

consider defining one or more SMPPTS spill datasets. SMPPTS spill datasets are used to contain SYSMODs when the primary SMPPTS dataset is full. Specifically, when the primary SMPPTS dataset is full, SYSMODs are written to the first spill dataset. When the first spill dataset is full, SYSMODs are then written to the second spill dataset, and so on. This continues until all SYSMODs are written to one of the datasets, or until the primary SMPPTS and all of the spill datasets are full. SMPPTS spill datasets are a new function in SMP/E provided by PTF UR52517 for OS/390 Version 2 Release 5 and 6, and by PTF UR52518 for OS/390 Version 2 Release 7, 8, 9, and 10.

SMPPTS spill datasets are permanent partitioned datasets that you allocate and manage just like the SMPPTS dataset. You then define these permanent spill datasets to SMP/E using either DD statements or DDDEF entries. The first spill dataset must be specified with a DD statement or DDDEF entry named SMPPTS1, the second must be specified with a DD statement or DDDEF entry named SMPPTS2, and so on, up to a maximum of SMPPTS99.

When you specify SMPPTS spill datasets, do not skip any spill dataset DDnames; if a spill dataset is omitted, this indicates the end of the list, and SMP/E ignores any spill datasets which may follow the omitted dataset. For example, if you specify only SMPPTS1 and SMPPTS3, then SMP/E uses only SMPPTS1 and ignores SMPPTS3.

If SMPPTS spill datasets are defined using DDDEF entries, then the DDDEF entries for the spill datasets must be added to all zones (global, target, and distribution zones). If the DDDEF entries are added only to the global zone, the spill datasets will not be detected and used during APPLY and ACCEPT processing, and you will probably get a message indicating that a SYSMOD could not be located in the SMPPTS dataset.

Whether your SMPPTS spill datasets are defined by DDDEF entries or DD statements, SMP/E will refer to each of the datasets, in sequential order, when writing to, reading from, and deleting SYSMOD members from the SMPPTS datasets.

Also, after SMP/E writes a SYSMOD member to the SMPPTS or a spill dataset, SMP/E does not keep track of which dataset that SYSMOD member resides in. This means you are free to manage the SMPPTS and spill datasets however you like. You can merge datasets,

split datasets, and move members from one dataset to another, without affecting SMP/E.

As part of SMPPTS spill dataset support, SMP/E will now compress an out-of-space SMPPTS dataset and retry any operation which results in an out-of-space condition. The RETRYDDN subentry of an OPTIONS entry indicates whether SMP/E should compress datasets and retry operations. You should specify RETRYDDN(ALL) in your active OPTIONS entry. This tells SMP/E to compress all out-of-space datasets and retry operations. The primary SMPPTS dataset and all SMPPTS spill datasets are eligible for RETRY processing.

If RETRYDDN(ALL) is not specified, then SMP/E will not compress the SMPPTS or spill datasets. Rather, an out-of-space condition will cause SMP/E to simply move on to the next dataset in sequential order, without first compressing the dataset and retrying the operation.

SOURCE

```
/* PSA => CVT =>                                                    */
/*               ASVT => ASCB                                       */
/*               ASVT => ASCB => ASXB => TCB => JSCB => JCT         */
/*               ....                                               */
/*                                                                  */
/*                                                                  */

   /* default values                                 */

skey   =  N

   /* main routine                                   */
disp_rc = 0
do while disp_rc <= 4         /* until PF3 */
  "ispexec tbcreate tabasinf keys(name)
    names(asid uid pgm jsd jst ssd sst) nowrite»
  "ispexec control display lock"
  function = «Address Spaces Scan... "
  "ispexec display panel(asinfow)"
  call get_info
  call sort_table
  "ispexec tbdispl   tabasinf panel(asinfo)"
  disp_rc = rc
  "ispexec tbclose   tabasinf"
  "ispexec tberase   tabasinf"
end
exit
```

```
get_info:
  /*=========================================================================*/
  /* PSA  - offsets are decimal                                              */
  /*      - subpool: 239       - common                                      */
  /*=========================================================================*/
  o_flccvt   =  00016                     /* psa   => cvt       */
  /*=========================================================================*/
  /* CVT  - offsets are decimal                                              */
  /*      - subpool: nucleus   - common                                      */
  /*=========================================================================*/
  o_cvtprodn = -00040                     /* sp level           */
  o_cvtprodi = -00032                     /* sp fmid            */
  o_cvtasvt  =  00556                     /* cvt   => asvt      */
  /*=========================================================================*/
  /* ASVT - offsets are decimal                                              */
  /*      - subpool: 245       - common                                      */
  /*=========================================================================*/
  o_asvtasvt =  00512                     /* asvt acronym       */
  o_asvtmaxu =  00516                     /* max number of as   */
  o_asvtenty =  00528                     /* asid entries       */
  /*=========================================================================*/
  /* ASCB - offsets are decimal                                              */
  /*      - subpool: 245       - common                                      */
  /*=========================================================================*/
  o_ascbascb =  00000                     /* ascb acronym       */
  o_ascbasid =  00036                     /* asid               */
  o_ascbjbns =  00176                     /* pointer to jobname */
  o_ascbasxb =  00108                     /* pointer to asxb    */
  o_ascbrctp =  00124                     /* pointer to rct tcb */
  /*=========================================================================*/
  /* ASXB - offsets are decimal                                              */
  /*      - subpool: 255       - private                                     */
  /*=========================================================================*/
  o_asxbasxb =  00000                     /* asxt acronym             */
  o_asxbftcb =  00004                     /* pointer to first tcb     */
  o_asxbltcb =  00008                     /* pointer to last  tcb     */
  o_asxbuser =  00192                     /* userid                   */
  /*=========================================================================*/
  /* TCB  - offsets are decimal                                              */
  /*      - subpool: 253       - private                                     */
  /*=========================================================================*/
  o_tcbjscbb =  00181                     /* pointer to jscb    */
  o_tcbtcbid =  00256                     /* tcbid acronym      */
  /*=========================================================================*/
  /* JSCB - offsets are decimal                                              */
  /*      - subpool: 253       - private                                     */
  /*=========================================================================*/
  o_jscbjcta =  00261                     /* pointer to jct     */
  o_jscbpgmn =  00360                     /* job step pgm name  */
```

```
/*====================================================================*/
/* JCT  - offsets are decimal                                         */
/*      - subpool: 236       - private                                */
/*====================================================================*/
o_jctjname =  00008                     /* jobname                    */
o_jctjtptn =  00016                     /* terminal name              */
o_jctjmrss =  00143                     /* step start time            */
o_jctjmrjt =  00146                     /* job start time             */
o_jctjmrjd =  00149                     /* job start date             */
o_jctssd   =  00157                     /* step start date            */
/********************************************************************/
  p_cvt       =  d2x(o_flccvt)
  a_cvt       =  d2x( c2d(storage(p_cvt,4)) )
     /* get sp level                                    */
  a_cvtprodn  = d2x( x2d(a_cvt) + o_cvtprodn )
  v_cvtprodn  = storage(a_cvtprodn,8)
  if debug = "Y" then say   v_cvtprodn
     /* get sp fmid                                     */
  a_cvtprodi  = d2x( x2d(a_cvt) + o_cvtprodi )
  v_cvtprodi  = storage(a_cvtprodi,8)
  p_asvt      = d2x( x2d(a_cvt) + o_cvtasvt )
  a_asvt      = d2x( c2d(storage(p_asvt,4)) )
  a_asvtasvt  = d2x( x2d(a_asvt) + o_asvtasvt )
  v_asvtasvt  = storage(a_asvtasvt,4)
     /* get max number of address spaces                */
  a_asvtmaxu  = d2x( x2d(a_asvt) + o_asvtmaxu )
  v_asvtmaxu  = storage(a_asvtmaxu,4)
  max_as      = c2d(v_asvtmaxu)
     /* get asid entries                                */
  a_asvtenty  = d2x( x2d(a_asvt) + o_asvtenty )
  p_entry     = a_asvtenty
  do i = max_as to 1 by -1
       /* get ascb for this asid                        */
    p_ascb      = p_entry
    v_p_ascb = right(d2x( c2d(storage(p_ascb,4))),8,'0')
    first_byte =  substr(v_p_ascb,1,1)
                                      /* high order bit off?  */
                                      /* yes = valid entry    */
    if c2d(bitand(first_byte,'8"x)) = 0 then
     do
       a_ascb      = d2x( c2d(storage(p_ascb,4)) )
       ascb_address = right(a_ascb,8,'0')
       /* get ascb acronym          */
       a_ascbascb  = d2x( x2d(a_ascb) + o_ascbascb )
       v_ascbascb  = storage(a_ascbascb,4)
       /* get ascb asid             */
       a_ascbasid  = d2x( x2d(a_ascb) + o_ascbasid )
       v_ascbasid  = storage(a_ascbasid,2)
       asid_dec = c2d(v_ascbasid)
       /* get jobname / stc name    */
       a_ascbjbns  = d2x( x2d(a_ascb) + o_ascbjbns )
```

```
v_ascbjbns  = storage(a_ascbjbns,4)
p_jobname   = d2x( c2d( v_ascbjbns ) )
v_jobname   = storage(p_jobname,8)

/* Point to asxb           */
p_asxb      = d2x( x2d(a_ascb) + o_ascbasxb )
a_asxb      = d2x( c2d(storage(p_asxb,4)) )
asxb_address = right(a_asxb,8,'0')
/* get asxb acronym in private area */
/*                    ============ */
a_asxbasxb  = d2x( x2d(a_asxb) + o_asxbasxb )
addr_dec    = X2D(a_asxbasxb)
len_dec     =   4
v_asxbasxb     = xtsomem(asid_dec,addr_dec,len_dec)
/* get userid       in private area */
/*                    ============ */
a_asxbuser  = d2x( x2d(a_asxb) + o_asxbuser )
addr_dec    = X2D(a_asxbuser)
len_dec     =   8
v_asxbuser  = xtsomem(asid_dec,addr_dec,len_dec)
/* get rct tcb pointer  in private area */
/*                         ============ */
a_asxbltcb  = d2x( x2d(a_asxb) + o_asxbltcb )
addr_dec    = X2D(a_asxbltcb)
len_dec     =   4
v_asxbltcb     = xtsomem(asid_dec,addr_dec,len_dec)
a_tcb       = d2x( c2d(v_asxbltcb) )
tcb_address = right(a_tcb,8,'0')
/* get tcb  acronym in private area */
/*                    ============ */
a_tcbtcbid  = d2x( x2d(a_tcb) + o_tcbtcbid )
addr_dec    = X2D(a_tcbtcbid)
len_dec     =   4
v_tcbtcbid     = xtsomem(asid_dec,addr_dec,len_dec)
/* get jscb pointer  in private area */
/*                      ============ */
a_tcbjscbb  = d2x( x2d(a_tcb) + o_tcbjscbb )
addr_dec    = X2D(a_tcbjscbb)
len_dec     =   3
v_tcbjscbb  = xtsomem(asid_dec,addr_dec,len_dec)
a_jscb      = d2x( c2d(v_tcbjscbb) )
jscb_address = right(a_jscb,8,'0')
/* get step program name in private area */
/*                           ============ */
a_jscbpgmn  = d2x( x2d(a_jscb) + o_jscbpgmn )
addr_dec    = X2D(a_jscbpgmn)
len_dec     =   8
v_jscbpgmn  = xtsomem(asid_dec,addr_dec,len_dec)
/* get jct  pointer  in private area */
/*                      ============ */
a_jscbjcta  = d2x( x2d(a_jscb) + o_jscbjcta )
```

```
addr_dec    = X2D(a_jscbjcta)
len_dec     =   3
v_jscbjcta  = xtsomem(asid_dec,addr_dec,len_dec)
a_jct       = d2x( c2d(v_jscbjcta) + 16 )
jct_address = right(a_jct,8,'0')
/* get jobname           in private area */
/*                          ============ */
a_jctjname  = d2x( x2d(a_jct ) + o_jctjname )
addr_dec    = X2D(a_jctjname)
len_dec     =   32
v_jctjname  = xtsomem(asid_dec,addr_dec,len_dec)
/* get terminal name     in private area */
/*                          ============ */
a_jctjtptn   = d2x( x2d(a_jct ) + o_jctjtptn  )
addr_dec    = X2D(a_jctjtptn )
len_dec     =   8
v_jctjtptn   = xtsomem(asid_dec,addr_dec,len_dec)
/* get job start date    in private area */
/*                          ============ */
a_jctjmrjd  = d2x( x2d(a_jct ) + o_jctjmrjd )
addr_dec    = X2D(a_jctjmrjd)
len_dec     =   3
v_jctjmrjd  = xtsomem(asid_dec,addr_dec,len_dec)
vv             = c2x(v_jctjmrjd)
job_start_date = substr(vv,1,2) || «.» || substr(vv,3,3)
/* get job start time    in private area */
/*                          ============ */
a_jctjmrjt  = d2x( x2d(a_jct ) + o_jctjmrjt )
addr_dec    = X2D(a_jctjmrjt)
len_dec     =   3
v_jctjmrjt  = xtsomem(asid_dec,addr_dec,len_dec)
vv             = c2d(v_jctjmrjt) % 100
ss             = right(vv // 60 , 2, "0")
vv             = vv % 60
mm             = right(vv // 60 , 2, "0")
hh             = right(vv % 60 , 2 , "0")
job_start_time = hh || «:» || mm || «:» || ss
/* get step start date    in private area */
/*                           ============ */
a_jctssd    = d2x( x2d(a_jct ) + o_jctssd   )
addr_dec    = X2D(a_jctssd)
len_dec     =   3
v_jctssd    = xtsomem(asid_dec,addr_dec,len_dec)
vv             = c2x(v_jctssd)
step_start_date = substr(vv,1,2) || «.» || substr(vv,3,3)
/* get step start time    in private area */
/*                           ============ */
a_jctjmrss  = d2x( x2d(a_jct ) + o_jctjmrss )
addr_dec    = X2D(a_jctjmrss)
len_dec     =   3
v_jctjmrss  = xtsomem(asid_dec,addr_dec,len_dec)
```

```
          vv              = c2x(v_jctjmrss)
          vv              = c2d(v_jctjmrss) % 100
          ss              = right(vv // 60 , 2, "0")
          vv              = vv % 60
          mm              = right(vv // 60 , 2, "0")
          hh              = right(vv % 60 , 2 , "0")
          step_start_time = hh || «:» || mm || «:» || ss
          /*                                          */
           name       = v_jobname
           asid       = asid_dec
           uid        = v_asxbuser
           pgm        = v_jscbpgmn
           jsd        = job_start_date
           jst        = job_start_time
           ssd        = step_start_date
           sst        = step_start_time
           «ispexec tbadd  tabasinf»
         end
        p_entry     = d2x( x2d(p_entry) + 4 )
      end
return
/*********************************************************************/
parse_parm:
  parse arg parm
  select
   when abbrev("DEBUG",parm,1) then DEBUG = "Y"
   otherwise say parm "invalid option"
  end
return
/*********************************************************************/
/* function to call xtsomes1 module                                  */
/*********************************************************************/
xtsomem:
 arg a1,a2,a3
 xtsomem_asid = a1
 xtsomem_addr = a2
 xtsomem_len  = a3
 xtsomes1
return xtsomem_field
/*********************************************************************/
/* function to sort ispf table                                       */
/*********************************************************************/
sort_table:
  select
   when skey = "N" then
     do
      skeyf = "name"
      sd = "A"
      fields = "fields("skeyf",c,"sd")"
     end
   when skey = "A" then
```

```
     do
      skeyf = "asid"
      sd = "A"
      st = "n"
      fields = "fields("skeyf","st","sd")"
     end
   when skey = "JD" then
     do
      skeyf1 = "jsd"
      sd1 = "A"
      skeyf2 = "jst"
      sd2 = "A"
      fields = "fields("skeyf1",c,"sd1","skeyf2",c,"sd2")"
     end
   when skey = "SD" then
     do
      skeyf1 = "ssd"
      sd1 = "A"
      skeyf2 = "sst"
      sd2 = "A"
      fields = «fields("skeyf1",c,"sd1","skeyf2",c,"sd2")"
     end
   otherwise
  end
  «ispexec tbsort    tabasinf " fields
return
```

ASINFO PANEL

```
)Attr Default(%[_)
 ' type(text) intens(low)  color(green) hilite(reverse)
 ? type(output) intens(low) caps(off) color(turq)
 # type(output) intens(high) caps(off) color(yellow)
 £ type(output) intens(low) caps(off) color(green)
 } type(output) intens(high) color(green)
 { type(text) intens(high)  color(turq)
 | type(text) intens(low) color(green)
 ] type(text) intens(low) color(red)
)Body Expand(oo) Width(&ZSCREENW)
%-o-o-'Address Spaces Info%-o-o-
%Command ===>_ZCMD                                o o%Scroll ===>_amt [
%Sort Key%===>_z | N[/¨A[/¨JD[/¨SD
%
  [                                Job    Job      Step   Step
  [Name     Asid Userid   Program  Start  Start    Start  Start
  [         Dec                    Date   Time     Date   Time
%  ____ __ ____ ____ ___ ____ ___ ____
)Model
  #name    ?asid?uid     ?pgm     ?jsd   ?jst     ?ssd   ?sst
)Init
  &tsel = ""
```

```
  .HELP = asinfoh
  .ZVARS = "(skey)"
)Proc
)End
```

ASINFOW WAIT PANEL

```
)attr
 £ TYPE(TEXT)   INTENS(LOW)  COLOR(TURQ)
 [ TYPE(TEXT)   INTENS(HIGH) COLOR(GREEN)  HILITE(BLINK)
 } TYPE(TEXT)   INTENS(LOW)  COLOR(BLUE)
 ( TYPE(TEXT)   INTENS(HIGH) COLOR(RED)
 ) TYPE(OUTPUT) INTENS(HIGH) COLOR(PINK) hilite(blink)
 { type(text) intens(high) color(yellow)
 | type(text) intens(low)  color(turq)
)Body Expand(§§) Width(&ZSCREENW)
[WW        WW  OOOOOOOOOOOO  RRRRRRRRRRR   KK        KK    {Time     -|&ZTIME
[WW        WW  OOOOOOOOOOOO  RRRRRRRRRRRR  KK       KK     {Date     -|&ZDATE
[WW        WW  OO        OO  RR        RR  KK      KK      {Julian   -|&ZJDATE
[WW        WW  OO        OO  RR        RR  KK     KK       {System   -|&ZSYSID
[WW        WW  OO        OO  RR        RR  KK    KK
[WW        WW  OO        OO  RRRRRRRRRRR  KKKKKKK       ______
[WW   WW   WW  OO        OO  RRRRRRRRRRR   KKKKKKK       ______
[WW  WWWW  WW  OO        OO  RR    RR      KK    KK
[WW WW  WW WW  OO        OO  RR     RR     KK     KK
[WWWW    WWWW  OO        OO  RR      RR    KK      KK
[WWW      WWW  OOOOOOOOOOOO  RR       RR   KK       KK
[WW        WW  OOOOOOOOOOOO  RR        RR  KK        KK
                                  [IIIIIIIIII   NN        NN   GGGGGGGGGG
                                 [IIIIIIIIII   NNN       NN  GGGGGGGGGGG
                                    [II       NNNN      NN  GG        GG
(Please be patient, performing:      [II       NN NN     NN  GG
                                    [II       NN  NN    NN  GG
                                   [II       NN   NN   NN  GG
)FUNCTION                         [II       NN    NN  NN  GG     GGGGG
                                 [II       NN     NN NN  GG     GGGGG
                                [II       NN      NNNN  GG        GG
                               [II       NN       NNN  GG        GG
                            [IIIIIIIIII   NN        NN  GGGGGGGGGGG
{SMP/E SMPOUT processing    [IIIIIIIIII   NN        N   GGGGGGGGGG
)init
)proc
)end
```

ASINFOH HELP PANEL

```
)Attr Default(%[_)
 ' type(text) intens(low)  color(green) hilite(reverse)
 ? type(output) intens(low) caps(off) color(turq)
 # type(output) intens(high) caps(off) color(yellow)
```

```
 £ type(output) intens(low) caps(off) color(green)
 } type(output) intens(high) color(green)
 { type(text) intens(high)  color(turq)
 | type(text) intens(low) color(green)
 ] type(text) intens(low) color(red)
)Body Expand(oo) Width(&ZSCREENW)
%-o-o-'Address Spaces Info{-o-o-
%Command ===>_ZCMD                                  o o%Scroll ===>_amt [

 ¨This panel shows "Address Space Info" for all active address spaces.

  The list can be sorted by:

    ]N [: Address space name (ascending).
    ]A [: Asid decimal value (ascending).
    ]JD[: Job Start Date (ascending).
    ]SD[: Step Start Date (ascending).

)Init
)Proc
)End
```

ASINFO SAMPLE DISPLAY

```
 ——————————— Address Spaces Info ——— Row 1 to 16 of 52
 Command ===>                                                Scroll ===>
PAGE
 Sort Key ===> N   N / A / JD / SD

                                 Job    Job      Step   Step
   Name     Asid Userid   Program  Start  Start    Start  Start
            Dec                   Date   Time     Date   Time
   ———— —— ———— ———— ——— ———— ——— ————
   *MASTER* 1    *BYPASS* IEEMB86Ø Ø1.1Ø1 13:Ø9:15 Ø1.1Ø1 13:Ø9:15
   ALLOCAS  18                     Ø1.1Ø1 13:Ø9:15 Ø1.1Ø1 13:Ø9:15
   ANTASØØØ 13   +ANTASØØ ANTXAINI Ø1.1Ø1 13:1Ø:52 Ø1.1Ø1 13:1Ø:52
   ANTMAIN  12   +ANTMAIN ANTMAIN  Ø1.1Ø1 13:1Ø:41 Ø1.1Ø1 13:1Ø:41
   APPC     42   *BYPASS* ATBINITM Ø1.1Ø1 13:12:15 Ø1.1Ø1 13:12:15
   APSWPR72 39   TCP      APSPPIEP Ø1.1Ø1 16:14:53 Ø1.1Ø1 16:14:53
   ASCH     43   *BYPASS* ASBSCHIN Ø1.1Ø1 13:12:15 Ø1.1Ø1 13:12:15
   ASCHINT  47   *BYPASS* IEFIIC   Ø1.1Ø1 13:12:21 Ø1.1Ø1 13:12:21
   BPXAS    32   TCP      BPXPRFC  Ø1.1Ø1 13:13:24 Ø1.1Ø1 13:13:24
   BPXOINIT 4Ø1  OMVSKERN BPXPINPR Ø1.1Ø1 13:11:3Ø Ø1.1Ø1 13:11:3Ø
   CATALOG  3Ø   +CATALOG IGGØCLXØ Ø1.1Ø1 13:1Ø:42 Ø1.1Ø1 13:1Ø:42
   CMF      48   *BYPASS* BBM9DAØØ Ø1.11Ø 16:58:Ø1 Ø1.11Ø 16:58:Ø1
   CMFCAS   41   *BYPASS* BBM9ZAØØ Ø1.11Ø 16:57:32 Ø1.11Ø 16:57:32
   CONSOLE  1Ø   *BYPASS*          Ø1.1Ø1 13:Ø9:15 Ø1.1Ø1 13:Ø9:15
```

Memory mapping

As all sysprogs are no doubt aware, the memory layout of OS/390 consists of a variety of components (such as CSA, SQA, PVT, etc). However, that does not mean that it is always easy to understand where it all is and how big it is without some research to find the figures, plus it is not always easy to remember the order of the various 'bits'. As a consequence, and also as part of an attempt to explain the memory situation to someone new to OS/390, I created a simple dialog to map the memory. It runs under ISPF and is invoked by issuing TSO RMAPSTR (see code on the following pages). Once this command has been issued it should display a screen similar to this:

```
Memory Information for LPAR PRD1        Row 1 to 19 of 19
Command ===>                                          Scroll ===> PAGE

  Name              Start    End          Size
 PSA               00000000  00000FFF        4K
 RCT               00001000  00004FFF       16K
 PVT               00005000  009FFFFF     9.98MB
 CSA               00A00000  00CEDFFF     2.93MB
 MLPA              00000000  00000000        0K
 FLPA              00CEE000  00CF7FFF       40K
 PLPA              00CF8000  00E6FFFF     1.47MB
 SQA               00E70000  00FCAFFF     1.36MB
 RW-NUCLEUS        00FCB000  00FD97BF       57K
 RO-NUCLEUS        00FDA000  00FFFFFF      152K
 16MEG LINE        ————  ————
 ERO-NUCLEUS       01000000  015B992F     5.72MB
 ERW-NUCLEUS       015BA000  025C7FFF    16.05MB
 E-SQA             025C8000  09E4BFFF   120.52MB
 E-PLPA            09E4C000  0C238FFF    35.93MB
 E-FLPA            0C239000  0C23BFFF       12K
 E-MLPA            0C23C000  0C38BFFF     1.31MB
 E-CSA             0C38C000  14AFFFFF   135.45MB
 E-PVT             14B00000  7FFFFFFF  1717.00MB
```

As an extra bit of functionality it is possible to dump out onto the screen each area of memory simply by using Z as a line command (as in Z for ZAP, but I have not included any storage alter capability in this dialog for safety reasons). As an example of what the output from Z looks like, the following is of a selection of the PSA.

```
STORAGE DISPLAY             Row 1 to 19 of 64
Command ===>                                                Scroll ===> PAGE

         Ø        4        8        C
======== |─── |─── |─── |───    ================ Region
00000000 040C0000 814DE908 00000000 00000000    ..  A(Z          PSA
00000010 00FD5C90 00000000 070C0000 91A5E1C4     .*.    ..  JV.D PSA
00000020 078D0000 8B7E95FA 078D2000 8B7CD1A6    ..  .=N.... .@JW PSA
00000030 00000000 00000000 078D1000 8B7E7A06           ... .=:. PSA
00000040 00000000 00000000 00000000 00FD5C90              .*. PSA
00000050 00000000 00000000 040C0000 81306ED0           ..  A.>} PSA
00000060 040C0000 80FE8380 00080000 FD21F560    ..  ..C.    ..5- PSA
00000070 00080000 FD220408 040C0000 814D7280       ... ..  A(.. PSA
00000080 00000000 00001202 0002006D 00020011         .. . _ . . PSA
00000090 14B90000 00000000 00000000 00000000    ..               PSA
000000A0 0000EF00 014DCB88 00000000 00000000      . .(.H         PSA
000000B0 00000000 00000000 00010E4B 02DF3C98            ......Q PSA
000000C0 28000100 00000000 00000000 00000000    . .              PSA
000000D0 00000000 00000000 00000000 00000000                     PSA
000000E0 00000000 00000000 00000000 00000000                     PSA
000000F0 00000000 00000000 00000000 00000000                     PSA
00000100 00000000 00000000 00000000 00000000                     PSA
00000110 00000000 00000000 00000000 00000000                     PSA
00000120 00000000 00000000 00000000 00000000                     PSA
```

Note that the storage display is based on using REXX's STORAGE function, so it may be restricted at your site and it is also restricted to your own address space if viewing PVT or E-PVT. It will also not always read some unallocated CSA and SQA areas.

INSTALLATION

The dialog consists of one Assembler module that requires no special linkage that needs to be in your logon procedure STEPLIB, two REXX routines and four panels.

RMAPSTR

```
/* REXX                                                             */
/*                                                                  */
/* Obtain the lpar name for the panel display                       */
/*                                                                  */
CVTECVT=D2X(C2D(STORAGE(10,4))+140)       /* point to cvtsysad */
lparname=STRIP(STORAGE(D2X(C2D(STORAGE(CVTECVT,4))+344),8))
/*                                                                  */
```

```
/* Analyse the storage layout on this OS/390                    */
/*                                                              */
NUMERIC DIGITS 15
rowpos=1
looper:
'ISPEXEC TBCREATE REGION NAMES(name start end size) NOWRITE REPLACE'
/* */
/* Call Assembler support routine to obtain relevant information */
/* */
CALL RMAPSTOR
/* */
/* Now loop around to create the table */
/* */
address ispexec
/* */
DO x=1 to map_name.0
/* */
if map_name.x='ERO-NUCLEUS' then do /* check for 16 meg line */
   name='16Meg Line'                /* and output a special  */
   start='————'                     /* bit of info.          */
   end='————'
   size=''
  'TBADD REGION'
  END
name=map_name.x
start=C2X(start_address.x)
end=C2X(end_address.x)
size=C2D(end_address.x)-c2d(start_address.x)+1
size=size%1024
IF size>1024 THEN DO
   size=FORMAT(size/1024,,2)
   size=size||'Mb'
   END
ELSE size=size||'K'
'TBADD REGION'
END
redisplay:
'TBTOP REGION'
'TBSKIP REGION NUMBER('rowpos')'
'TBDISPL REGION PANEL(RMAPSTP)'
rowpos=ztdtop
if reply='END' then EXIT
IF ztdsels¬=0 THEN DO
   CALL table_routine
   selector=''
   SIGNAL redisplay
   END
if reply='ENTER' then signal looper
/* */
/* now process the selection commands */
```

```
/* */
table_routine:
DO prime=1 to ztdsels
   UPPER selector
   selector.prime=selector
   start.prime=start
   IF ztdsels =1 THEN LEAVE
   IF ztdsels >1 THEN 'TBDISPL REGION'
END
DO x=1 TO prime
   IF selector.x='Z' THEN ADDRESS TSO '%STORDISR' start.x
END
RETURN
```

RMAPSTP

The display panel RMAPSTP is shown below:

```
)Attr Default(%+_)
   ! type(output) intens(high) caps(on ) just(left )
   $ type(output) intens(high) caps(on ) just(right)
   @ type(output) intens(low ) caps(off) just(asis )
)Body  Expand(//)
/ /%Memory Information for LPAR!lparname / /
%Command ===>_zcmd                                / /%Scroll ===>_amt +
+
   Name               Start    End          Size
)Model
_z!z                !z        !z        $z        +
)Init
  .HELP = RMAPSTH                 /* INSERT NAME OF TUTORIAL PANEL  */
  .ZVARS = '(selector name start end size)'
  &amt = PAGE
)PROC
&REPLY = .RESP
)End
```

RMAPSTH

The help panel RMAPSTH is shown below:

```
)ATTR
 ' TYPE(PT)                          /* panel title line            */
 ? TYPE(PIN)                         /* panel instruction line      */
 # TYPE(NT)                          /* normal text attribute       */
 } TYPE(ET)                          /* emphasized text attribute   */
 ! TYPE(DT)                          /* description text            */
 [ AREA(SCRL)                        /* scrollable area attribute   */
)BODY
'———————— Help panel for MVS address map ————————————-
```

```
+
+Command ==>_ZCMD        +
+
+This is a reference panel to assist in understanding the memory
+organization in your MVS system.
+
+============================================================================
[pnarea                                                                     [
[                                                                           [
[                                                                           [
[                                                                           [
[                                                                           [
[                                                                           [
[                                                                           [
[                                                                           [
[                                                                           [
[                                                                           [
[                                                                           [
+============================================================================
+
%Use ENTER to scroll downwards through the available data.
)AREA pnarea
#
}DESCRIPTION:
+
+This screen provides an address map of the MVS running on this LPAR
+as follows:
+
+PSA .......... Prefix storage area
+RCT .......... Region Control Task
+PVT .......... Private Address Space address range
+CSA .......... Common System Area.
+MLPA ......... Modifiable Link Pack Area
+FLPA ......... Fixed Link Pack Area
+PLPA ......... Pageable Link Pack Area
+SQA .......... System Queue Area
+RW-NUCLEUS ... Read Write Nucleus.
+RO-NUCLEUS ... Read Only Nucleus.
+16meg line ... This line inserted to indicate the end of 24bit
+               addressing.
+ERO-NUCLEUS .. Extended Read Only Nucleus.
+ERW-NUCLEUS .. Extended Read Write Nucleus.
+E-SQA ........ Extended System Queue Area.
+E-PLPA ....... Extended Pageable Link Pack Area
+E-FLPA ....... Extended Fixed Link Pack Area.
+E-MLPA ....... Extended Modifiable Link Pack Area.
+E-CSA ........ Extended Common System Area.
+E-PVT ........ Extended Private Area.
+
+In all cases the size of the memory chunk is also displayed.
#
```

```
}LINE COMMANDS:
+Z .... display the storage as hex data
)PROC
&ZTOP=RMAPSTH
&ZUP=RMAPSTH
&ZCONT=RMAPSTH
)END
```

STORDISR

The REXX code for storage display is shown below:

```
/* REXX                                                  */
/* Retrieve storage information for a particular address */
/*                                                       */
arg addr
holdaddr=0                          /*  fix address errors */
CALL RMAPSTOR                       /* retrieve memory map */
/*                                                       */
/* need a check for greater than 2                       */
/*                                                       */
numeric digits 20
restart:
rowpos=1
restart2:
IF DATATYPE(addr,'X')=0 | LENGTH(addr)>8 THEN DO
   ZEDSMSG='Address incorrect'
   ZEDLMSG='Please retype your address'
   ADDRESS ISPEXEC 'SETMSG MSG(ISRZ001)'
   addr=holdaddr
   SIGNAL restart2
   END
IF C2D(X2C(addr))>(2**31)-1024 THEN DO
   ZEDSMSG='Address incorrect'
   ZEDLMSG='Specified value too large'
   ADDRESS ISPEXEC 'SETMSG MSG(ISRZ001)'
   addr=(2**31)-1024
   addr=D2X(addr)
   END
/*                                                          */
/* now use the REXX storage function to retrieve 1k of info */
/*                                                          */
storage_area=STORAGE(addr,1024)
ADDRESS ISPEXEC
'TBCREATE STORTABL NAMES(address hex text location) NOWRITE REPLACE'
DO x=1 to 1024 by 16
   text=SUBSTR(storage_area,x,16)
   hextext=C2X(text)
   mask=SUBSTR(mask_data,x,16)
   DO WHILE INDEX(mask,'.')¬=0 /* sub _'s for unallocated bytes */
```

```
      hextext=OVERLAY('__',hextext,(INDEX(mask,'.')*2-1))
      text=OVERLAY('.',text,INDEX(mask,'.'))
      mask=OVERLAY('_',mask,INDEX(mask,'.'))
      END
   hex=INSERT(' ',INSERT(' ',INSERT(' ',hextext,8),17),26)
   address=D2X(C2D(X2C(addr))+(x-1),8)
   CALL SETADDR address
   location=result
   'TBADD STORTABL'
END
/* */
redisplay:
'TBTOP STORTABL'
'TBSKIP STORTABL NUMBER('rowpos')'
holdaddr=address   /* hang on to page top address in case of errors */
ztdret='VERTICAL'
'TBDISPL STORTABL PANEL(STORDISP) POSITION(CURPOS) AUTOSEL(YES)'
/*                          */
/* check for end of dialog */
/*                          */
IF reply='END' THEN EXIT
UPPER zcmd
IF WORD(zcmd,1)='DUMP' THEN DO
   addr=WORD(zcmd,2)
   SIGNAL restart
   END
rowpos=ztdtop
/*                                                                  */
/* If the end of the table is reached, determine if going up or down */
/* through the rowpos and set a new address accordingly before       */
/* building the table afresh.                                        */
/*                                                                  */
IF ztdadd='YES' THEN DO
   IF ztdsrid=Ø THEN DO /* we are going upwards */
     'TBSKIP STORTABL NUMBER('rowpos')'
     addr=C2D(X2C(address))-(16*ztdamt)
     IF addr<Ø THEN DO
        addr=Ø
        END
     addr=D2X(addr)
     rowpos=1
     SIGNAL restart2
     END
   IF ztdsrid¬=Ø THEN DO
     'TBSKIP STORTABL NUMBER('ztdscrp-1')'
     addr=C2D(X2C(address))+16 /* address will be last value set */
     addr=D2X(addr)
     rowpos=1
     SIGNAL restart2
     END
   END
```

```
rowpos=ztdtop
SIGNAL restart2
/*                            */
/* Retrieve the storage areas */
/*                            */
setaddr:
arg testaddr
testaddr=X2C(RIGHT('00000000'||testaddr,8))
location='Unknown Address'
DO y=1 TO map_name.0
   IF testaddr>=start_address.y & testaddr<=end_address.y
   THEN LEAVE
END
location=map_name.y
RETURN location
```

STORDISP

The storage display panel is shown below:

```
)Attr Default(%+_)
   ! type(output) intens(high) caps(on ) just(left )
   @ type(output) intens(low) caps(on ) just(left )
   _ type(input) intens(low ) caps(off) just(asis )
)Body  Expand(//)
% / /  STORAGE DISPLAY  / /
%Command ===>_zcmd                                / /%Scroll ===>_amt +
+
+         0        4        8        C
%======== |─── |─── |─── |───    ================ Region
)Model
@address |hex                                    |text
@location
)Init
  .Help = stordish               /* insert name of tutorial panel  */
  &amt = PAGE
)PROC
&REPLY = .RESP
)End
STORDISH
```

The help panel for storage display is shown below:

```
)ATTR
 ' TYPE(PT)                        /* panel title line            */
 ? TYPE(PIN)                       /* panel instruction line      */
 # TYPE(NT)                        /* normal text attribute       */
 } TYPE(ET)                        /* emphasized text attribute   */
 ! TYPE(DT)                        /* description text            */
 [ AREA(SCRL)                      /* scrollable area attribute   */
```

```
)BODY
'──────── Help panel for storage display ──────────────-
+
+Command ==>_ZCMD        +
+
+This screen displays a hex storage layout for your own address space
+and for the system common areas.
+
+=============================================================================
[pnarea                                                                      [
[                                                                            [
[                                                                            [
[                                                                            [
[                                                                            [
[                                                                            [
[                                                                            [
[                                                                            [
[                                                                            [
[                                                                            [
[                                                                            [
[                                                                            [
[                                                                            [
+=============================================================================
+
%Use ENTER to scroll downwards through the available data.
)AREA pnarea
#
}DESCRIPTION:
+
+In low intensity on the left column of the screen is the address
+of the data being shown in high intensity alongside. The high intensity
+information consists of 16 hex bytes followed by a text representation
+of the hex. The final low intensity column on the right shows what part
+of the memory is being displayed (eg PSA, PVT, etc)
+
#
}SUBCOMMANDS:
+
+DUMP:    The DUMP command followed by a suitable address will cause the
+         display to move to the specified address.
+
)PROC
&ZTOP=STORDISH
&ZUP=STORDISH
&ZCONT=STORDISH
)END
```

RMAPSTOR

The Assembler code for the memory mapping is shown below.

RMAPSTOR

```
*********************************************************************
* RMAPSTOR: A REXX FUNCTION TO DISPLAY A MEMORY MAP OF THE MVS UPON
*           WHICH THIS REXX IS RUN.
* USAGE: CALL RMAPSTOR
* NOTE:  RMAPSTOR WILL RETURN THE FOLLOWING INFORMATION:
*        MAP_NAME.X ....... AMOUNT OF REAL STORAGE AVAILABLE
*        START_ADDRESS.X .. STARTING BINARY ADDRESS FOR SPECIFIED MAP.
*        END_ADDRESS.X .... THE END ADDRESS FOR THE SPECIFIED MAP.
*
*        THE SUPPLIED DETAILS WILL BE SUPPLIED IN ASCENDING ADDRESS
*        ORDER AS FOLLOWS:
*        PSA .... PREFIXED SAVE AREA
*        RCT .... REGION CONTROL TABLE
*        PVT .... PRIVATE REGION
*        CSA .... COMMON SYSTEM AREA
*        MLPA ... MODIFYABLE LINK PACK AREA
*        FLPA ... FIXED LINK PACK AREA
*        PLPA ... PAGEABLE LINK PACK AREA
*        SQA .... SYSTEM QUEUE AREA
*        RWNUC .. READ-WRITE NUCLEUS
*        RONUC .. READ ONLY NUCLEUS
*        ERONUC . READ ONLY NUCLEUS ABOVE THE LINE
*        ERWNUC . READ-WRITE NUCLEUS ABOVE THE LINE
*        ESQA ... SQA ABOVE THE LINE
*        EPLPA .. PAGEABLE LINK PACK AREA ABOVE THE LINE
*        EFLPA .. FIXED LINK PACK AREA ABOVE THE LINE
*        EMLPA .. MODIFYABLE LINK PACK AREA ABOVE THE LINE
*        ECSA ... CSA ABOVE THE LINE
*        EPVT ... EXTENDED PRIVATE REGION
*********************************************************************
RMAPSTOR TITLE 'REXX FUNCTION TO RETRIEVE SYSPROG INFORMATION'
RMAPSTOR AMODE 31
RMAPSTOR RMODE ANY
RMAPSTOR CSECT
        REXREGS
        PRINT GEN
        BAKR 14,Ø
        LR   12,15
        LA   11,2Ø48(,12)
        LA   11,2Ø48(,11)
        USING RMAPSTOR,12,11
         PRINT GEN
         LR    R1Ø,RØ                  *R1Ø -> A(ENVIRONMENT BLOCK)
         USING ENVBLOCK,R1Ø
         L     R9,ENVBLOCK_IRXEXTE     *R9 -> A(EXTERNAL EP TABLE)
         USING IRXEXTE,R9
* GET A WORK AREA FOR REXX OUTPUT
* MAP WITH R2 ... NEED TO DO THIS BEFORE ANY ROUTING TO POSSIBLE
* REXX VARIABLE OUTPUT (EG ROUTINE ABENDØØ1)
         STORAGE OBTAIN,LENGTH=AREALEN,ADDR=(2)
```

```
         USING WORKAREA,2
* PREPARE THE REXX AREA FOR USE
         XC  COMS(COMSLEN),COMS * SET TO LOW VALUES
         LA  15,COMID
         ST  15,COMS
         LA  15,COMDUMMY
         ST  15,COMS+4
         ST  15,COMS+8
         LA  15,COMSHVB
         ST  15,COMS+12
         LA  15,COMRET
         ST  15,COMS+16
         OI  COMS+16,X'8Ø'
         MVC COMID,=C'IRXEXCOM'
VARLOOP DS ØH
*********************************************************************
*        BUILD IRXEXCOM PARAMETERS
*********************************************************************
         XR 3,3
         USING PSA,3
         L  4,FLCCVT
         USING CVT,4
         L  3,PSAAOLD
         USING ASCB,3
         L   3,ASCBLDA
         USING LDA,3
         L   7,CVTSMEXT
         USING CVTVSTGX,7
         L   R5,CVTGDA
         USING GDA,R5
*** HAVING MAPPED THE AREAS. NOW NEED TO CALCULATE START AND END
*** ADDRESS FOR ALL THE MAPPED AREAS.
*
*** FIRST START WITH THE PSA. THE START FOR THIS MUST BE ZERO.
*** THE END IS SYSTEM_START_ADDRESS-1
         SHOWARAY PSALIT,MAP_NAME
         SHOWARAY ZERO,START_ADDRESS
         L R1,LDASTRTS   * GET THE ADDRESS
         BCTR 1,Ø        * AND DECREMNET IT
         ST  R1,ENDADD   * AND STORE FOR ARRAY
         SHOWARAY ENDADD,END_ADDRESS
*** NOW DO THE RCT
         SHOWARAY RCTLIT,MAP_NAME
         SHOWARAY LDASTRTS,START_ADDRESS
         L R1,LDASTRTA
         BCTR R1,Ø
         ST R1,ENDADD
         SHOWARAY ENDADD,END_ADDRESS
*** NOW DO THE PVT
         SHOWARAY PVTLIT,MAP_NAME
         SHOWARAY LDASTRTA,START_ADDRESS
```

```
         L R1,GDACSA
         BCTR R1,Ø
         ST R1,ENDADD
         SHOWARAY ENDADD,END_ADDRESS
* HAVING REACHED THE POINT OF THE CSA, THINGS GET COMPLICATED.
* MLPA AND FLPA MAY NOT NECESSARILY BE PRESENT. SO WORKING OUT THE
* END ADDRESS FOR THE CSA IS BETTER CARRIED OUT BY ADDING THE SIZE
* OF THE CSA TO THE START ADDRESS.
         SHOWARAY CSALIT,MAP_NAME
         SHOWARAY GDACSA,START_ADDRESS
         L R1,GDACSA
         L R15,GDACSASZ
         AR R1,R15
         BCTR R1,Ø
         ST R1,ENDADD
         SHOWARAY ENDADD,END_ADDRESS
* HAVING REACHED THE POINT OF THE MLPA, THINGS GET EASIER AGAIN
* AS START AND END ADDRESSES FOR MLPA, PLPA, FLPA AND NUCLEUS ITEMS
* CAN BE EXTRACTED DIRECTLY FROM THE CVT.
         SHOWARAY MLPALIT,MAP_NAME
         SHOWARAY CVTMLPAS,START_ADDRESS
         SHOWARAY CVTMLPAE,END_ADDRESS
         SHOWARAY FLPALIT,MAP_NAME
         SHOWARAY CVTFLPAS,START_ADDRESS
         SHOWARAY CVTFLPAE,END_ADDRESS
         SHOWARAY PLPALIT,MAP_NAME
         SHOWARAY CVTPLPAS,START_ADDRESS
         SHOWARAY CVTPLPAE,END_ADDRESS
* NOW WE NEED TO DO THE SQA
* IN A SIMILAR MANNER TO CSA.
         SHOWARAY SQALIT,MAP_NAME
         SHOWARAY GDASQA,START_ADDRESS
         L R1,GDASQA
         L R15,GDASQASZ
         AR R1,R15
         BCTR R1,Ø
         ST R1,ENDADD
         SHOWARAY ENDADD,END_ADDRESS
         SHOWARAY RWNUC,MAP_NAME
         SHOWARAY CVTRWNS,START_ADDRESS
         SHOWARAY CVTRWNE,END_ADDRESS
         SHOWARAY RONUC,MAP_NAME
         SHOWARAY CVTRONS,START_ADDRESS
         SHOWARAY BELOW,END_ADDRESS
         SHOWARAY ERONUC,MAP_NAME
         SHOWARAY BELOW1,START_ADDRESS
         SHOWARAY CVTRONE,END_ADDRESS
         SHOWARAY ERWNUC,MAP_NAME
         SHOWARAY CVTERWNS,START_ADDRESS
         SHOWARAY CVTERWNE,END_ADDRESS
* NOW WE NEED TO DO THE E-SQA
```

```
* IN A SIMILAR MANNER TO CSA.
         SHOWARAY ESQALIT,MAP_NAME
         SHOWARAY GDAESQA,START_ADDRESS
         L R1,GDAESQA
         L R15,GDAESQAS
         AR R1,R15
         BCTR R1,Ø
         ST R1,ENDADD
         SHOWARAY ENDADD,END_ADDRESS
         SHOWARAY EPLPALIT,MAP_NAME
         SHOWARAY CVTEPLPS,START_ADDRESS
         SHOWARAY CVTEPLPE,END_ADDRESS
         SHOWARAY EFLPALIT,MAP_NAME
         SHOWARAY CVTEFLPS,START_ADDRESS
         SHOWARAY CVTEFLPE,END_ADDRESS
         SHOWARAY EMLPALIT,MAP_NAME
         SHOWARAY CVTEMLPS,START_ADDRESS
         SHOWARAY CVTEMLPE,END_ADDRESS
         SHOWARAY ECSALIT,MAP_NAME
         SHOWARAY GDAECSA,START_ADDRESS
         L R1,GDAECSA
         L R15,GDAECSAS
         AR R1,R15
         BCTR R1,Ø
         ST R1,ENDADD
         SHOWARAY ENDADD,END_ADDRESS
*** NOW THE E-PVT
         SHOWARAY EPVTLIT,MAP_NAME
         SHOWARAY LDAESTRA,START_ADDRESS
         SHOWARAY GIG_LIMIT,END_ADDRESS
         SHOWBASE MAP_NAME
         B RETURN
         EJECT
*********************************************************************
***      RETURN TO CALLER                                         ***
***      RELEASING ALL STORAGE IN THE PROCESS                     ***
*********************************************************************
RETURN   DS ØH
         STORAGE RELEASE,LENGTH=AREALEN,ADDR=(2)
         PR
*********************************************************************
***      WORKING STORAGE, ETC                                     ***
*********************************************************************
         TITLE  'WORKING STORAGE / DSECTS'
         LTORG
BELOW    DC X'ØØFFFFFF'
BELOW1   DC X'Ø1ØØØØØØ'
PSALIT   DC C'PSA'
RCTLIT   DC C'RCT'
PVTLIT   DC C'PVT'
CSALIT   DC C'CSA'
```

```
MLPALIT DC C'MLPA'
FLPALIT DC C'FLPA'
PLPALIT DC C'PLPA'
SQALIT  DC C'SQA'
RWNUC   DC C'RW-NUCLEUS'
RONUC   DC C'RO-NUCLEUS'
ERONUC  DC C'ERO-NUCLEUS'
ERWNUC  DC C'ERW-NUCLEUS'
ESQALIT DC C'E-SQA'
EPLPALIT DC C'E-PLPA'
EFLPALIT DC C'E-FLPA'
EMLPALIT DC C'E-MLPA'
ECSALIT DC C'E-CSA'
EPVTLIT DC C'E-PVT'
ZERO    DC F'Ø'
GIG_LIMIT DC X'7FFFFFFF'
WORKAREA DSECT
*      IRXEXCOM PARAMETER AREA
         DS  ØD
COMS     DS  5AL4
COMID    DS  CL8
COMDUMMY DS  AL4              * NOT USED
COMSHVB  DS  (SHVBLEN)X       * IRXEXCOM SHVBLOCK (LENGTH FROM DSECT)
COMRET   DS  AL4              * IRXECOM RC
         DS  ØD
ENDADD   DS F
COMSLEN  EQU *-COMS
AREALEN  EQU *-WORKAREA
         IHAPSA
         IHAECVT
         IHAGDA
         IHAASCB
         IHALDA
         IRXEFPL
         IRXARGTB
         IRXEVALB
         IRXENVB
         IRXEXTE
         IRXSHVB
         CVT DSECT=YES
         END
```

SUPPORTING MACROS

The supporting macros are shown below.

REXREGS

```
MACRO
        REXREGS
```

```
        LCLA &CNT
&CNT    SETA Ø
.LOOP   ANOP
R&CNT   EQU &CNT
&CNT    SETA &CNT+1
        AIF (&CNT LT 16).LOOP
        MEND
```

SHOWSET

```
MACRO
        SHOWSET
        AIF (D'SHOW_START).NONEED
        B  BY_SHOW_START
SHOW_START DS ØH
        ST R1Ø,COMRET
        LA 6,COMSHVB
        USING SHVBLOCK,R6
        XC COMSHVB(SHVBLEN),COMSHVB
        XC SHVNEXT,SHVNEXT
        MVI SHVCODE,C'S'
        BR 14
ABENDØØ1 DS ØH
        ABEND 1   * REQUIRED FOR THE OTHER MACROS. SAVES SOME CODING.
BY_SHOW_START DS ØH
LITLOC  LOCTR
@_UNPACK DC CL16' '
         DC CL8' '  * FILL FIELD
         ORG @_UNPACK+8
@_UNPACKER DC CL8' '
         ORG
@_DWORD  DS CL8     * USED FOR THE DEBIN FUNCTION
&SYSECT LOCTR
.NONEED ANOP
        BAL 14,SHOW_START
        MEND
```

SHOWARAY

```
MACRO
  SHOWARAY &LABEL,&ASNAME,&ERR=ABENDØØ1,&LEN=,&SUBARRAY=,&DEBIN=,&LINK=
        PRINT NOGEN
*********************************************************************
* MACRO TO CREATE REXX ARRAY VARIABLES
*
* NOTE RESTRICTION: THIS MACRO IS LIMITED TO CREATING UP TO 9,999,999
*                   ENTRIES FOR EACH ARRAY.
* MACRO FORMAT:
*       SHOWARAY &LABEL,&ASNAME,&ERR=,&LEN=,&SUBARRAY=,&DEBIN=
* WHERE:
```

```
*       &LABEL IS THE NAME OF THE LABEL WHICH ADDRESS THE FIELD FROM
*              WHERE THE DATA TO BE DEFINED IN A REXX VARIABLE IS
*              LOCATED
*       &ASNAME IS THE NAME TO BE ASSIGNED TO THE DATA FOR USE IN REXX
*       &ERR= IS THE LABEL TO BRANCH TO SHOULD AN ERROR OCCUR WHILE
*             CREATING THE REXX VARIABLE. BY DEFAULT IT IS ABENDØØ1
*       &LEN= IF THE DATA AT &LABEL IS NOT DEFINED SUCH THAT THE LENGTH
*             OF THE DATA IS WHAT YOU WANT, SIMPLY ENTER A NUMBER HERE
*             THAT DEFINES THE LENGTH REQUIRED. CAN ALSO BE USEFUL IF
*             NECESSARY TO DUMP OUT A LARGE AREA.
*       &SUBARRAY= IF A MULTI LEVEL ARRAY IS REQUIRED EG A.1.1 THEN
*                  SET THIS VALUE ACCORDINGLY.
*       &DEBIN= IF THE DATA TO BE CREATED IS BINARY, SETTING THIS TO A
*               VALUE WILL CONVERT THE SPECIFIED NUMBER OF BYTES FROM
*               BINARY TO CHARACTER. THE DEFAULT LENGTH FOR THE
*               OUTPUT DATA IS 4 BYTES. IF THIS IS INSUFFICIENT, THEN
*               SPECIFY A SUITABLE &LEN VALUE TO OVERRIDE IT.
*       &LINK= THIS IS A REXX NAME LABLE TO WHICH THE ARRAY COUNT IS
*              LINKED. THE PURPOSE OF THIS IS TO ALLOW A BRANCH OUT
*              OF ARRAY LOOPS WHILE STILL MAINTAINING NUMERIC
*              CONSISTENCY.
*********************************************************************
        PRINT GEN
        LCLA &DEFLEN
&DEFLEN SETA 16
        SHOWSET
LITLOC  LOCTR
&LABCHECK SETC '@_&ASNAME&SUBARRAY'
&LINKNAME SETC '@_&LINK'
         AIF   (D'&LABCHECK).BYPASS
         AIF   (T'&SUBARRAY EQ 'O').NORMNAME
&LABCHECK DC C'&ASNAME..&SUBARRAY'
         AGO   .EOFARRAY
.NORMNAME ANOP
&LABCHECK DC C'&ASNAME'
.EOFARRAY ANOP
&LABCHECK._ARRAY DC C'.       '
&LABCHECK._COUNTER DC PL4'Ø' * COUNTER FIELD FOR THIS ITEM
.BYPASS  ANOP
&SYSECT LOCTR
        AIF (T'&LINK EQ 'O').DOADD
        MVC &LABCHECK._COUNTER,&LINKNAME._COUNTER
        AGO .DOUNPK
.DOADD  ANOP
        AP &LABCHECK._COUNTER,=P'1' * INCREMENT THE COUNTER THIS PASS
.DOUNPK ANOP
        UNPK @_UNPACKER,&LABCHECK._COUNTER * UNPACK THE VALUE
        OI   @_UNPACKER+7,X'FØ'       * REMOVE THE SIGN
* NOW NEED TO WORK OUT THE LENGTH OF THE COUNTER BIT TO ADD TO ARRAY
        L    R15,&LABCHECK._COUNTER   * LOAD THE COUNTER VALUE TO WORK
*                                       OUT THE LENGTH
```

```
        SRL  R15,4                    * REMOVE THE SIGN
        XR   R14,R14                  * CLEAR R14 FOR A COUNTER
LOOP&SYSNDX DS ØH
        SRA  R15,4                    * MOVE DIGIT-BY-DIGIT
        LTR  R15,R15
        BZ   COUNT&SYSNDX
        LA   R14,1(,R14)
        B    LOOP&SYSNDX
COUNT&SYSNDX DS ØH
* NOW ADD COUNT FIELD TO NAME
        LA  R15,@_UNPACKER+7        * POINT TO END OF FIELD
        SR  R15,R14                 * AND COME BACK TO FIRST DIGIT.
        MVC &LABCHECK._ARRAY+1(7),Ø(R15)
        LA 1,&LABCHECK
        ST 1,SHVNAMA
* NOW CALCULATE NEW LENGTH
        LA 1,L'&LABCHECK
        LA 1,2(R14,R1)
        ST 1,SHVNAML
        AIF (T'&DEBIN EQ 'O').NORMLAB
*** NOW ALLOW FOR A BINARY CONVERSION
*** FIST CALCULATE THE ICM VALUE
&ICM    SETA (1 SLL &DEBIN)-1
        XR R15,R15
        ICM R15,&ICM,&LABEL           * LOAD THE BINARY VALUE
        CVD R15,@_DWORD               * CONVERT TO PACKED
        OI  @_DWORD+7,X'ØF'
        UNPK @_UNPACK,@_DWORD
*** IF THE LEN VALUE IS SUPPLIED THIS OVERRIDES THE DEFAULT OF 16
        AIF (T'&LEN EQ 'O').SETDEF    * LENGTH NOT SUPPLIED USE DEFLEN
&DEFLEN SETA &LEN                     * RESET DEFLEN TO SUPPLIED LEN
.SETDEF ANOP
        LA R1,@_UNPACK+(16-&DEFLEN)
        ST R1,SHVVALA
        LA R1,&DEFLEN
        AGO .OK
.NORMLAB ANOP
        LA 1,&LABEL
        ST 1,SHVVALA
        AIF (T'&LEN NE 'O').DOLEN
        LA 1,L'&LABEL
        AGO .OK
.DOLEN  ANOP
        LA 1,&LEN
.OK     ANOP
        ST 1,SHVVALL
        LR Ø,1Ø
        LA 1,COMS
        L 15,IRXEXCOM
        BALR 14,15
        LTR 15,15
        BNZ &ERR
        MEND
```

SHOWBASE

```
MACRO
        SHOWBASE &LABEL,&ERR=ABEND001,&SUBARRAY=
*********************************************************************
* MACRO TO CREATE REXX BASE VARIABLES
* SHOULD BE USED IN ASSOCIATION WITH A SHOWARAY MACRO. NOTE THAT A
* SHOWBASE MACRO IS OPTIONAL IF YOU ALREADY KNOW THE NUMBER OF
* VARAIABLES BEING SET. THIS WILL CREATE THE A.0 ENTRY
* MACRO FORMAT:
*       SHOWBASE &LABEL,&ERR=,&SUBARRAY=
* WHERE:
*       &LABEL IS THE NAME OF THE REXX ARRAY LABEL WHICH HAS BEEN
*              CREATED. THIS WILL CREATE THAT LABEL.0 ENTRY
*       &ERR= IS THE LABEL TO BRANCH TO SHOULD AN ERROR OCCUR WHILE
*             CREATING THE REXX VARIABLE. BY DEFAULT IT IS ABEND001
*       &SUBARRAY= IF SUBARRAYS HAVE BEEN USED THIS WILL INSERT THE
*                  APPROPRIATE VALUE EG A.1.0
*********************************************************************
        SHOWSET
        AIF (T'&SUBARRAY EQ 'O').NORMNAME
&ASNAME   SETC '&LABEL..&SUBARRAY..0'
        AGO .CHECKER
.NORMNAME ANOP
&ASNAME   SETC '&LABEL..0'
.CHECKER  ANOP
&LABCHECK SETC '@_&LABEL&SUBARRAY._COUNTER'
        AIF (D'&LABCHECK).ITSOK
MNOTE   NO ARRAY ELEMENTS DEFINED.
        MEXIT
.ITSOK  ANOP
LITLOC  LOCTR
@_A&SYSNDX DC C'&ASNAME'
&SYSECT LOCTR
        LA 1,@_A&SYSNDX
        ST 1,SHVNAMA
        LA 1,L'@_A&SYSNDX
        ST 1,SHVNAML
        UNPK @_UNPACKER,&LABCHECK
        OI  @_UNPACKER+L'@_UNPACKER-1,C'0'
        LA 1,@_UNPACKER
        ST 1,SHVVALA
        LA 1,L'@_UNPACKER
        ST 1,SHVVALL
        LR 0,10
        LA 1,COMS
        L 15,IRXEXCOM
        BALR 14,15
        LTR 15,15
        BNZ &ERR
        MEND
```

An EXEC to search strings and list lines above and below the searched string

THE PROBLEM

There are a number of tools available as well as utilities published in *MVS Update* to search for 'strings' in a PDS or a flat file. On various occasions, I found that such lists were of limited use for my analysis, as I needed more information. I wanted to see lines above or below such searched strings to assist me in doing analysis. This is why I have written this REXX EXEC.

A SOLUTION

This EXEC will search a PDS or a flat file for a series of strings either keyed in or coded in a dataset and will output lines along with the line numbers that contain the searched strings as well as lines that are above / below the lines that have the search string(s). The lines will not be repeated in case the gap between the lines that have the search string(s) is equal to less than the number of lines that are above or below the searched lines. For example, if you want to see three lines above and two lines below the 'hit' line, and the search string is found in lines 4, 6, 7, 11, the EXEC will display the lines 1, 2, 3, 4, 5, 6, 7, 8, 9, 11,12,13. The line numbers for the lines that have the searched string are enclosed in '*', while line numbers for the lines that are above the searched string are enclosed in '<' and '>' and line numbers for the lines that are below the searched string are enclosed in '>' and '<'. The output of the EXEC is a flat file. You can use C'text' Or T'text' Or text to search for text and can use X'HexText' Or H'Hextext' to search for Hex search. The EXEC will delete the output dataset if already present without warning. You can modify the defaults and the output filename by changing the values suitably in the subroutine BuildDflts.

FORMAT

AbvBelow <Param1> <Param2> <Param3> <Param4>

- Param1 = PDS Name or Flat File Name to search. You can search an entire PDS or have a pattern to search specific members. TSO rules apply for dataset names.
- Param2 = can be a File Name that has all the strings that you want to search that are coded line by line, or a single search string. If you want to have multiple search strings keyed in from the terminal, leave this and further parameters blank. The EXEC will then prompt you for the search strings. If you are providing a dataset that has the search strings, start PARA2 with DSN=
- Param3 = number of lines that are to be displayed above the line that has the search string(s).
- Param4 = number of lines that are to be displayed below the line that has the search string(s).

You can leave any or all of the parameters blank and call the EXEC. The EXEC will prompt you to supply each parameter that you have skipped. If you have supplied at least one parameter through the terminal, you can also see the names of PDS members that the EXEC is processing on the terminal.

```
/* ------------------------- REXX -------------------------------- */
/* If this character is not | logical OR, please make a    */
/* a global change to modify this character to logical OR. */
/* ------------------------- REXX -------------------------------- */
/* REXX EXEC To Search for a series of strings and output  */
/* lines above / below the search strings                  */
/* Format is:                                              */
/*   SpSearch filename SearchDsn/String nAbove nBelow      */
/* Filename can be a flat file or an entire PDS or a       */
/* a partial selection of members                          */
/* ------------------------------------------------------- */
/* Naming Conventions Used:                                */
/* REXX Commands and Functions start with a Capital letter */
/* TSO  Commands are in UPPER CASE                         */
/* User data (Variables/Constants/Labels) is a combination */
/* Of Upper and Lower case and                             */
/*  Alphanumeric user data starts with x  (lower x)        */
/*  Numeric      user data starts with n  (lower n)        */
/*  Switches / Flags       start  with s  (lower s)        */
/* ------------------------------------------------------- */
Trace O
Arg xFromDsn xSrchDat nAbove nBelow
Call BuildDflts                          /* Defaults are here       */
Call InitVarbls
```

```
If xFromDsn='' Then Call GetInpDsn
Call CheckInpDsn
If xSrchDat='' Then Call GetSearchData
 Else Call CheckSrchData

If xSrchDsn<>'' Then Call LoadSrchData
Call CnvrtSrchDat
If nAbove='' | nBelow=''  Then Call GetAboveBelow
Call AllocOut
If xType='Pds' Then Call AnalyzePds
 Else Call Analyze
If sHit='Y' Then Call ViewHits
 Else Call Nothing2Disp
Exit

/*———————  Start Of Subroutines ——————————*/
GetInpDsn:
/********/
'CLRSCRN'
xMsg.=''
xMsg.1='Type The Input Dataset Name To Search — TSO Rules Apply'
xMsg.2='ie If The Name Is Not In Quotes, 'Userid() 'Will Be Suffixed'
xMsg.3=Copies('-',Length(xMsg.1))
xMsg.4='Input can be a Sequential File Or a PDS.'
xMsg.5='For PDS, Type just the PDS Name to analyze all members Or'
xMsg.6='Choose a pattern for partial selection - Use * ? As Wild
Characters'
xMsg.7='Example Of Pattern: Your selection can be SAM*ENA OR NB* OR *ASK
OR S?KH*'
Do I=1 By 1 Until xMsg.I=''
 Say Center(xMsg.I,74)
 End
Pull xFromDsn .
xFromDsn=Strip(Translate(xFromDsn,"'",'"'))
Select
 When xFromDsn='' Then Do
  Say '*Error* You Have To Type A Dataset Name To Search'
  Say '        Program Aborted'
  Exit
  End
 When Left(xFromDsn,1)<>"'" Then Do
  xFromDsn="'"Userid()'.'xFromDsn"'"
  End
 When Left(xFromDsn,1)="'" & Right(xFromDsn,1)<>"'" Then Do
  xFromDsn=xFromDsn"'"
  End
 Otherwise Nop
 End
sWatch='Y'
Return
```

```
CheckInpDsn:
/**********/
xMemPat=''
If Left(xFromDsn,1)<>"'" Then xFromDsn="'"||UserId()"."xFromDsn
If Right(xFromDsn,1)<>"'" Then xFromDsn=xFromDsn"'"
If Pos('(',xFromDsn)>Ø Then Do
  Parse Var xFromDsn  xFromDsn '(' xMemPat ')' .
  If Verify(xMemPat,'*')=Ø Then xMemPat=''
  xFromDsn=xFromDsn"'"
  End
xAvail=SYSDSN(xFromDsn)
If xAvail<>'OK' Then Do
  Say '*Error*' xFromDsn 'Is Not Found'
  Say '        Program Aborted'
  Exit
  End
xAvailRc=LISTDSI(xFromDsn NORECALL)
If Rc<4 Then Do
  Select
   When Left(SYSDSORG,2)='PO' Then Do
    nLrecl=SYSLRECL
    xDsName=xFromDsn
    xType='Pds'
    End
   When Left(SYSDSORG,2)='PS' Then Do
    nLrecl=SYSLRECL
    xDsName=xFromDsn
    xType='Seq'
    End
   When Left(SYSDSORG,2)='VS' & sVsamSupport='Y' Then Do
    nLrecl=255
    xDsName=xFromDsn
    xType='Vsam'
    End
   Otherwise Do
    Say '*Error* The Dataset Organization Of' xFromDsn
    Say '        Is Not Supported By This REXX'
    Say '        Program Aborted'
    Exit
    End
   End
  End
 Else Do
  Say '*Error* Fatal Error While Accessing ' xFromDsn
  Say '        Program Aborted'
  Exit
  End
Return

GetSearchData:
/************/
'CLRSCRN'
```

```
xMsg.=''
xMsg.1='You Can Type Your Search Strings On The Terminal One By One'
xMsg.2='OR you can supply a Dataset which has all the Search Strings'
xMsg.3="For Text Strings,Use C'xxx' Or T'xxxx' Or Text"
xMsg.4="For Hex  Strings,Use X'HHHH' Or H'HHHH'"
xMsg.5='Code each search string in a new line If the input is a DSN'
xMsg.6='Type DSN=Your.Dataset.Name Or Start Typing The Search String'
xMsg.7='(For DSN, TSO Rules Apply)'
Do I=1 By 1 Until xMsg.I=''
 Say Center(xMsg.I,72)
 End
J=Ø
Do Forever
 Pull xAns
 Select
  When xAns='' Then Leave
  When Left(xAns,4)='DSN=' Then Do
   Parse Var xAns 'DSN=' xSrchDsn
   Leave
   End
  Otherwise Do
   J=J+1
   xSrch.J=xAns
   Say 'Type Your Next Search String — Enter Blank To Go To Next Phase'
   End
  End
 End
If xSrchDsn='' & J=Ø Then Do
 Say '*Error* No Search String Entered'
 Say '        Program Aborted'
 Exit
 End
xSrch.Ø=J
sWatch='Y'
Return

CheckSrchData:
/************/
If Pos('DSN=',xSrchDat)>Ø Then Do
  Parse Var xSrchDat 'DSN=' xSrchDSn
  End
 Else Do
  xSrchDsn=''
  xSrch.Ø=1
  xSrch.1=xSrchDat
  End
Return
LoadSrchData:
/***********/
Address 'TSO'
xSamFir=MSG('OFF')
```



```
'FREE DD(xSrchDsn)'
xAskNb=MSG(xSamFir)
'ALLOC DD(xSrchDsn) DSN('xSrchDsn') SHR'
If Rc<>Ø Then Do
  Say '*Error* Unable To Allocate ' xSrchDsn
  Say '        Program Aborted - Return Code From Allocate = ' Rc
  Exit
  End
'EXECIO * DISKR xSrchDsn (STEM xSrch.'
If Rc<>Ø Then Do
  Say '*Error* Unable To Read'  xSrchDsn
  Say '        Program Aborted - Return Code From Read = ' Rc
  Exit
  End
'EXECIO Ø DISKR xSrchsn (FINIS'
nRec.Ø=xSrch.Ø
xSamFir=MSG('OFF')
'FREE DD(xInpDsn)'
xAskNb=MSG(xSamFir)
Return

CnvrtSrchDat:
/***********/
Do I=1 By 1 Until I=xSrch.Ø
 xText=Strip(xSrch.I)
 nLenText=Length(xText)
 Select
  When Left(xText,2)="C'" Then xText=Substr(xText,3,nLenText-3)
  When Left(xText,2)="T'" Then xText=Substr(xText,3,nLenText-3)
  When Left(xText,2)="H'" Then xText=X2C(Substr(xText,3,nLenText-3))
  When Left(xText,2)="X'" Then xText=X2C(Substr(xText,3,nLenText-3))
  Otherwise Nop
  End
 If xText<>xSrch.I Then xSrch.I=xText
 End
Return

GetABoveBelow:
/************/
xMsg.=''
Select
 When nAbove='' Then Do
  xMsg.1='Type The Number Of Lines Above & Below That You Want To See'
  xMsg.2='Alongwith The Line That Has The Search String'
  xMsg.3='The Default Number Of Above The Search Lines Are' nDfltAbove
  xMsg.4='The Default Number Of Below The Search Lines Are' nDfltBelow
  xMsg.5='Type Two Numbers To Override The Default'
  End
 When nBelow='' Then Do
  xMsg.1='Type The Number Of Lines Below That You Want To See'
  xMsg.2='Alongwith The Line That Has The Search String'
```

```
  xMsg.3='The Default Number Of Below The Search Lines Are' nDfltBelow
  xMsg.4='Type A  Number To Override The Default'
  End
 Otherwise Nop
 End
Do I=1 By 1 Until xMsg.I=''
 Say Center(xMsg.I,72)
 End
Select
 When nAbove='' Then Pull nAbove nBelow .
 When nBelow='' Then Pull nBelow .
 Otherwise Nop
 End
If Datatype(nAbove,'W') Then Nop
 Else nAbove=nDfltAbove
If Datatype(nBelow,'W') Then Nop
 Else nBelow=nBelow
sWatch='Y'
Return

AnalyzePds:
/*********/
Select
 When Pos('*',xMemPat)>Ø | Pos('?',xMemPat)>Ø Then Do
  xPattern=",PATTERN("Strip(xMemPat)")"
  End
 When xMemPat<>'' Then Do
  xPattern=",PATTERN("Strip(xMemPat)")"
  sOneMember='Y'
  End
 Otherwise xPattern=''
 End
'ISPEXEC LMINIT DATAID(xNameInp) DATASET('xFromDsn') ENQ(SHR)'
If Rc<>Ø Then Do
  Say 'Unable to Access' xFromDsn
  Exit
  End
'ISPEXEC LMOPEN DATAID('xNameInp') OPTION(INPUT)'
If Rc<>Ø Then Do
  Say 'Unable to Open' xFromDsn
  Exit
  End
'CLRSCRN'
Do Forever
 'ISPEXEC LMMLIST DATAID('xNameInp') OPTION(LIST) MEMBER(xNextMem)
STATS(YES)' xPattern
 If Rc<>Ø Then Leave
 xNextMem=Strip(xNextMem)
 xDsName=Strip(xFromDsn,'T',"'")'('xNextMem")'"
 Call Analyze
 End
```

```
'ISPEXEC LMCLOSE DATAID('xNameInp')'
Return

Analyze:
/******/
sMemPresent='Y'
If xType='Pds' & sWatch='Y' Then Do
  nMem=nMem+1
  If nMem>2Ø Then Do
   nMem=Ø
   'CLRSCRN'
   End
  Say 'Analyzing' xNextMem
  End
Address 'TSO'
xSamFir=MSG('OFF')
'FREE DD(xInpDsn)'
xAskNb=MSG(xSamFir)
'ALLOC DD(xInpDsn) DSN('xDsName') SHR'
If Rc<>Ø Then Do
  Say '*Error* Unable To Allocate ' xDSName
  Say '        Program Aborted - Return Code From Allocate = ' Rc
  Exit
  End
'EXECIO * DISKR xInpDsn (STEM XREC.'
If Rc<>Ø Then Do
  Say '*Error* Unable To Read'  xDSName
  Say '        Program Aborted - Return Code From Read = ' Rc
  Exit
  End
'EXECIO Ø DISKR xInpDsn (FINIS'
xSamFir=MSG('OFF')
'FREE DD(xInpDsn)'
xAskNb=MSG(xSamFir)
nRec.Ø=xRec.Ø
sFound='N'
xHit.=''
xNazHit.=''
nLen=Length(nRec.Ø)
If nLen<5 Then nLen=5

Do I=1 By 1 While I<=nRec.Ø
 xData=xRec.I
 Do J=1 By 1 Until J=xSrch.Ø
  If Pos(Translate(xSrch.J),Translate(xData))>Ø Then Do
    xHit.I='*'
    sFound='Y'
    Leave J
    End
  End
 End
```

```
If sFound='N' Then Return
Do I=1 By 1 Until I=nRec.Ø
 If xHit.I='' Then Iterate I
 xNazHit.I=xHit.I
 If nAbove>Ø Then Do
   Do J1=1 By 1 Until J1=nAbove
    K=I-J1
    If K<Ø Then Iterate J1
    If xHit.K='' Then xNazHit.K='<'Right(K,nLen,Ø)'>'
    End
  End
 If nBelow>Ø Then Do
   Do J2=1 By 1 Until J2=nBelow
    K=I+J2
    If K>nRec.Ø Then Iterate J2
    If xHit.K='' Then xNazHit.K='>'Right(K,nLen,Ø)'<'
    End
  End
 End
If xType='Pds' Then Do
  xMsg1='* Member Name = ' Left(xNextMem,8) '*'
  xMsg2=Copies('*',Length(xMsg1))
  xLine.Ø=3
  xLine.1=Center(xMsg2,72)
  xLine.2=Center(xMsg1,72)
  xLine.3=xLine.1
  Call RiteDetl
  End
xLine.=''
J=Ø
nSaveLine=Ø
nLineNuM=Ø
Do I=1 By 1 Until I=nRec.Ø
 If xNazHit.I<>'' Then Do
   If xNazHit.I='*' Then xLine='#'Right(I,nLen,Ø)'#' xRec.I
    Else xLine=xNazHit.I  xRec.I
   nLineNum=Substr(xLine,2,nLen)
   If nAbove>Ø | nBelow>Ø Then Do
     If nLineNum-nSaveLine>1 Then Do
       J=J+1
       xLine.J=Copies('-',nLen+2)
       End
     nSaveLine=nLineNum
     End
   If Length(xLine)>255 Then xLine=Left(xLine,255)
   J=J+1
   xLine.J=xLine
   End
 End
xLine.Ø=J
Call RiteDetl
```

```
Return

RiteDetl:
/*******/
'EXECIO * DISKW xOutDsn  (STEM xLine.'
If Rc<>Ø Then Do
  Say '*Error* Write Failure On ' xFileOut 'RC='rc
  Say '        Program Aborted'
  xSamFir=MSG('OFF')
  'FREE DD(xOutDsn)'
  xAskNb=MSG(xSamFir)
  Exit 16
  End
nOutRecs=nOutRecs+xLine.Ø
sHit='Y'
Return

AllocOut:
/*******/
xOutName="'"xOutName"'"
xAvail=SYSDSN(xOutName)
If xAvail='OK' Then Do
  xSamFir=MSG('OFF')
  'DELETE' xOutName
  xAskNb=MSG(xSamFir)
  If Rc<>Ø Then Do
    Say '*Error*' xOutName 'Could Not Be Deleted - Return Code = ' Rc
    Say 'Program Aborted'
    Exit
    End
  End
nLen=nLrecl+1Ø
Select
 When nLen<81  Then  nLrecl=8Ø
 When nLen<133 Then nLrecl=132
 Otherwise nLrecl=255
 End
xDfltDisp='NEW UNIT(SYSDA) LRECL('nLrecl') SPACE(2Ø) DSORG(PS)
RECFM(F,B) TRACKS RELEASE'
xSamFir=MSG('OFF')
'FREE DD(xOutDsn)'
xAskNb=MSG(xSamFir)
'ALLOCATE DSN('xOutName') DD(xOutDsn)' xDfltDisp
If Rc<>Ø Then Do
  Say '*Error* Unable To Alloc' xOutName
  Say '        Return Code Is ' Rc
  Exit 16
  End
Return
ViewHits:
/*******/
'EXECIO Ø DISKW xOutDsn (FINIS'
```

```
xSamFir=MSG('OFF')
'FREE DD(xOutDsn)'
xAskNb=MSG(xSamFir)
'CLRSCRN'
If nOutRecs>Ø Then Do
  'ISPEXEC VIEW DATASET ('xOutName')'
  End
Return

Nothing2Disp:
/***********/
If sMemPresent='N' Then Do
  Say '*Error* No Members Found In Pds' xFromDsn 'That Could Match'
xMemPat
  Say '        Program Is Unable To Do Any Analysis'
  Return
  End
If nOutRecs=Ø Then Do
  Say '*Warning* There Were No Hits For Your Search String(s)'
  If xType='Pds' Then Say '          PDS Scanned :- ' xFromDsn
   Else Say '          DSN Scanned :- ' xFromDsn
  End
Return

InitVarbls:
/*********/
nLrecl=Ø
nMem=Ø
nOutRecs=Ø
sFirst='Y'
sHits='N'
sMemPresent='N'
xMsg=''
xSrchDsn=''
Return
/* ————————— Note ————————— Note ————————— Note ————————— */
/* The Defaults are defined here. Modify them to suit your          */
/* installations standards.                                         */
/* ———————————————————————————————————————————————————————————— */
BuildDflts:
/*********/
xOutName=Userid()'.AAAF.SEARCH.LIST'  /* Output Dataset Name         */
sWatch='N'                            /* Y = Display Members On Term */
sVsamSupport='N'                      /* Y = ISPF can read VSAM File */
nDfltAbove=2                          /* nAbove = Above Lines to disp*/
nDfltBelow=1                          /* nBelow = Below Lines to dis */
Return
```

Accessing cross-memory storage in REXX

INTRODUCTION

TSO/REXX provides a very useful function called STORAGE, which allows the retrieval of virtual storage from your own address space. However, it does not allow access to the virtual storage of another address space. This can be useful for developing system oriented-EXECs, that need to access control blocks in other address spaces. A previous article in Issue 93 of *MVS Update*, from June 1994, pages 12-19, describes the command XTSOMEM, which can be called from a REXX routine to access the storage of another address space.

It should be noted that this version of XTSOMEM has a major restriction: only non-swappable address spaces may be accessed. It is because XTSOMEM uses 'Access Registers' that only non-swappable address spaces may be accessed in this way. In order to bypass this restriction, I wrote a new utility called XTSOMES1, which uses SRBs to retrieve virtual storage from another address space

XTSOMES1

```
XTSOMES1 CSECT
XTSOMES1 AMODE 31
XTSOMES1 RMODE ANY
         SAVE  (14,12)
         BASR  R12,Ø
         USING *,R12                    R12 = BASE REGISTER
         L     R2,Ø(R1)             LOAD PARAMETER ADDRESS
         LH    R3,Ø(R2)             LENGTH
         GETMAIN R,LV=WORKL
         ST    R1,8(R13)
         ST    R13,4(R1)
         LR    R13,R1
         USING WORK,R13
*/////////////////////////////////////////////////////////////////////*
* GET INPUT ARGUMENTS: ASID - ADDR - LEN                               *
*/////////////////////////////////////////////////////////////////////*
*=====================================================================*
* GET IKJCT441 ADDRESS                                                 *
*=====================================================================*
```

```
         SR    R2,R2
         USING PSA,R2
         L     R3,FLCCVT
         USING CVTMAP,R3
         L     R4,CVTTVT
         USING TSVT,R4
         L     R5,TSVTVACC              LOAD IKJCT441 ENTRY POINT
         ST    R5,ADDR441
*=============================================================================*
* GET          A VARIABLE (TSVNOIMP) - VIA BALR - ASID                 *
*=============================================================================*
         L     R1,=A(TSVNOIMP)
         ST    R1,T_ECODE
* PREPARE VARIABLE ATTRIBUTES
         MVC   VARNAME,=CL2Ø"XTSOMEM_ASID'
         LA    R2,12                VARIABLE NAME LENGTH
         LA    R1,VARNAME
         ST    R1,T_NAMEA
         ST    R2,T_NAMEL
* PREPARE PARAMETER LIST
         LA    R2,T_ECODE
         ST    R2,T_P1
         LA    R2,T_NAMEA
         ST    R2,T_P2
         LA    R2,T_NAMEL
         ST    R2,T_P3
         LA    R2,T_VALUEA
         ST    R2,T_P4
         LA    R2,T_VALUEL
         ST    R2,T_P5
         LA    R2,T_TOKEN
         ST    R2,T_P6
         OI    T_P6,X'8Ø'           LAST ENTRY
         XC    T_TOKEN,T_TOKEN      ZERO
         L     R15,ADDR441
         LA    R1,T_PARML           ADDRESS OF PARAMETER LIST
         BALR  R14,R15
         L     R3,T_VALUEA
         MVC   WTOM,=CL8Ø"ASID:'
         MVC   WTOM+Ø8(L'VARVALUE),Ø(R3)
         L     R2,=A(WTOMLEN)
         STH   R2,WTOML
         LA    R2,WTOMSG
*        WTO   TEXT=(R2),MF=(E,WTOLIST)
         L     R1,T_VALUEL              LENGTH OF VARIABLE VALUE
         MVC   V_CL1Ø,=CL1Ø''
         L     R3,=A(L'V_CL1Ø)
         LA    R4,V_CL1Ø
         L     R5,T_VALUEL
         SR    R3,R5
         AR    R4,R3
         L     R6,T_VALUEA
```

```
         LR    R7,R5
         MVCL  R4,R6
         PACK  V_PL8,V_CL1Ø
         CVB   R3,V_PL8
         ST    R3,ASID
*        LINK  EP=SHOWREGS
*=====================================================================*
* GET          A VARIABLE (TSVNOIMP) - VIA BALR - ADDR                *
*=====================================================================*
         L     R1,=A(TSVNOIMP)
         ST    R1,T_ECODE
* PREPARE VARIABLE ATTRIBUTES
         MVC   VARNAME,=CL2Ø"XTSOMEM_ADDR'
         LA    R2,12                VARIABLE NAME LENGTH
         LA    R1,VARNAME
         ST    R1,T_NAMEA
         ST    R2,T_NAMEL
* PREPARE PARAMETER LIST
         LA    R2,T_ECODE
         ST    R2,T_P1
         LA    R2,T_NAMEA
         ST    R2,T_P2
         LA    R2,T_NAMEL
         ST    R2,T_P3
         LA    R2,T_VALUEA
         ST    R2,T_P4
         LA    R2,T_VALUEL
         ST    R2,T_P5
         LA    R2,T_TOKEN
         ST    R2,T_P6
         OI    T_P6,X'8Ø'           LAST ENTRY
         XC    T_TOKEN,T_TOKEN      ZERO
         L     R15,ADDR441
         LA    R1,T_PARML           ADDRESS OF PARAMETER LIST
         BALR  R14,R15
         L     R3,T_VALUEA
         MVC   WTOM,=CL8Ø"ADDR:'
         MVC   WTOM+Ø8(L'VARVALUE),Ø(R3)
         L     R2,=A(WTOMLEN)
         STH   R2,WTOML
         LA    R2,WTOMSG
*        WTO   TEXT=(R2),MF=(E,WTOLIST)
         L     R1,T_VALUEL          LENGTH OF VARIABLE VALUE
         MVC   V_CL1Ø,=CL1Ø''
         L     R3,=A(L'V_CL1Ø)
         LA    R4,V_CL1Ø
         L     R5,T_VALUEL
         SR    R3,R5
         AR    R4,R3
         L     R6,T_VALUEA
         LR    R7,R5
         MVCL  R4,R6
```

```
         PACK  V_PL8,V_CL1Ø
         CVB   R3,V_PL8
         ST    R3,ADDR
*=========================================================================*
* GET          A VARIABLE (TSVNOIMP) - VIA BALR - LEN                     *
*=========================================================================*
         L     R1,=A(TSVNOIMP)
         ST    R1,T_ECODE
* PREPARE VARIABLE ATTRIBUTES
         MVC   VARNAME,=CL2Ø"XTSOMEM_LEN'
         LA    R2,11                VARIABLE NAME LENGTH
         LA    R1,VARNAME
         ST    R1,T_NAMEA
         ST    R2,T_NAMEL
* PREPARE PARAMETER LIST
         LA    R2,T_ECODE
         ST    R2,T_P1
         LA    R2,T_NAMEA
         ST    R2,T_P2
         LA    R2,T_NAMEL
         ST    R2,T_P3
         LA    R2,T_VALUEA
         ST    R2,T_P4
         LA    R2,T_VALUEL
         ST    R2,T_P5
         LA    R2,T_TOKEN
         ST    R2,T_P6
*         OI    T_P6,X'8Ø'           LAST ENTRY
         XC    T_TOKEN,T_TOKEN      ZERO
         L     R15,ADDR441
         LA    R1,T_PARML           ADDRESS OF PARAMETER LIST
         BALR  R14,R15
         L     R3,T_VALUEA
         MVC   WTOM,=CL8Ø"LEN: "
         MVC   WTOM+Ø8(L'VARVALUE),Ø(R3)
         L     R2,=A(WTOMLEN)
         STH   R2,WTOML
         LA    R2,WTOMSG
*        WTO   TEXT=(R2),MF=(E,WTOLIST)
         L     R1,T_VALUEL          LENGTH OF VARIABLE VALUE
         MVC   V_CL1Ø,=CL1Ø''
         L     R3,=A(L'V_CL1Ø)
         LA    R4,V_CL1Ø
         L     R5,T_VALUEL
         SR    R3,R5
         AR    R4,R3
         L     R6,T_VALUEA
         LR    R7,R5
         MVCL  R4,R6
         MVC   WTOM,=CL8Ø"LEN: "
         MVC   WTOM+Ø4(L'V_CL1Ø),V_CL1Ø
         L     R2,=A(WTOMLEN)
```

```
         STH   R2,WTOML
         LA    R2,WTOMSG
*        WTO   TEXT=(R2),MF=(E,WTOLIST)
         PACK  V_PL8,V_CL1Ø
         CVB   R3,V_PL8
         ST    R3,LEN
*/////////////////////////////////////////////////////////////////////*
*                                                                     *
*/////////////////////////////////////////////////////////////////////*
         SR    R2,R2
         USING PSA,R2
         L     R3,FLCCVT
         USING CVTMAP,R3
         L     R4,CVTASVT
         USING ASVT,R4
         L     R5,ASVTMAXU             MAXIMUM NUMBER OF AS
         LA    R6,1                    AS COUNTER
         LA    R7,ASVTENTY             POINT TO FIRST ENTRY
ASLOOP   EQU   *
         L     R8,Ø(R7)
         USING ASCB,R8
         CLC   ASCBASID,ASID+2         TARGET ASID?
         BE    FOUND
NEXTAS   EQU   *
         LA    R6,1(R6)
         CR    R6,R5
         BH    NOTFOUND
         LA    R7,4(R7)
         B     ASLOOP
NOTFOUND EQU   *
         B     RETURN
FOUND    EQU   *
         ST    R8,ADDR_TAR             STORE ADDRESS OF TARGET ASCB
*=====================================================================*
* GET CURRENT ADDRESS SPACE PARAMETERS                                *
*=====================================================================*
         L     R8,PSAAOLD-PSA          GET ADDRESS OF CALLER ASCB
         USING ASCB,R8
         ST    R8,MYASCB
         MVC   MYASID,ASCBASID
         L     R4,CVTTCBP
         MVC   MYTCB,4(R4)             COPY CURRENT TCB ADDRESS
         CLC   MYASID,ASID+2           CURRENT AS = TARGET AS?
         BNE   NOTSAME
*=====================================================================*
* TARGET AS == CURRENT AS => SIMPLE MVCL                              *
*=====================================================================*
         L     R6,ADDR
         L     R7,LEN
         LA    R8,VARVALUE
         LR    R9,R7
         MVCL  R8,R6
```

```
         B     SETOUT
*=============================================================================*
* TARGET AS <> CURRENT AS => MUST SCHEDULE SRB                                *
*=============================================================================*
NOTSAME  EQU   *
         GETMAIN R,LV=SRBLEN,SP=241
         LTR   R15,R15
         BNZ   ERROR
         ST    R1,ADDR_SRB
         GETMAIN R,LV=PARM_LEN,SP=241
         LTR   R15,R15
         BNZ   ERROR
         ST    R1,ADDR_PRM
         LR    R9,R1
         USING PARM,R9
         MODESET KEY=ZERO,MODE=SUP
         XC    Ø(PARM_LEN,R9),Ø(R9)     ZERO PARM AREA SP241 => AUTH
         LA    R2,ADDR_MOD
         LOAD  EP=XTSOMES2,GLOBAL=YES,EOM=NO,LOADPT=(R2)
* PREPARE SRB FIELDS
         L     R8,PSAAOLD-PSA           GET ADDRESS OF CALLER ASCB
         USING ASCB,R8
         ST    R8,MYASCB
         MVC   MYASID,ASCBASID
         L     R4,CVTTCBP
         MVC   MYTCB,4(R4)              COPY CURRENT TCB ADDRESS
         L     R4,ADDR_SRB
         USING SRB,R4
         XC    Ø(SRBLEN,R4),Ø(R4)       ZERO SRB AREA SP241 => AUTH
* INITIALIZE SRB
         MVC   SRBID,=C'SRB "
         BASR  RØ,Ø                     GET ADDRESSING MODE
         N     RØ,=X'8ØØØØØØØ'          LEAVE ONLY ADDRESSING MODE
         L     R1,ADDR_MOD              GET ADDRESS OF SRB ROUTINE
         OR    R1,RØ                    PUT ADDRESSING MODE IN
         ST    R1,SRBEP                 STORE SRB ENTRY POINT
         ST    R9,SRBPARM               STORE PARM ADDRESS FOR SRB
         MVC   SRBASCB,ADDR_TAR         STORE TARGET ASCB ADDRESS
         MVC   SRBPASID,MYASID          STORE CALLER ASID
         MVC   SRBPTCB,MYTCB            STORE CALLER TCB ADDRESS
* COPY PARAMETERS FOR SRB ROUTINE
         MVC   M_ADDR,ADDR
         MVC   M_LEN,LEN
         MVC   M_ASCB,MYASCB            STORE CALLER ASCB ADDRESS
*                                        FOR POST ECB BY SRB
* SCHEDULE SRB ROUTINE IN TARGET ADDRESS SPACE
         SCHEDULE SRB=(R4),SCOPE=GLOBAL
         WAIT 1,ECB=M_ECB
         DELETE EP=XTSOMES2
         MODESET KEY=NZERO,MODE=PROB
         L     R2,ADDR_SRB
         FREEMAIN R,LV=SRBLEN,A=(R2),SP=241
```

```
         MVC   VARVALUE,M_FIELD
         L     R2,ADDR_PRM
         FREEMAIN R,LV=PARM_LEN,A=(R2),SP=241
SETOUT   EQU   *
*/////////////////////////////////////////////////////////////////////*
* SET OUTPUT VARIABLE: FIELD                                          *
*/////////////////////////////////////////////////////////////////////*
*=====================================================================*
* SET          A VARIABLE (TSVEUPDT) - VIA BALR - FIELD               *
*=====================================================================*
         L     R1,=A(TSVEUPDT)
         ST    R1,T_ECODE
* PREPARE VARIABLE ATTRIBUTES
         MVC   VARNAME,=CL2Ø"XTSOMEM_FIELD'
         LA    R2,13               VARIABLE NAME LENGTH
         LA    R1,VARNAME
         ST    R1,T_NAMEA
         ST    R2,T_NAMEL
         L     R2,LEN              VARIABLE VALUE LENGTH
         LA    R1,VARVALUE
         ST    R1,T_VALUEA
         ST    R2,T_VALUEL
* PREPARE PARAMETER LIST
         LA    R2,T_ECODE
         ST    R2,T_P1
         LA    R2,T_NAMEA
         ST    R2,T_P2
         LA    R2,T_NAMEL
         ST    R2,T_P3
         LA    R2,T_VALUEA
         ST    R2,T_P4
         LA    R2,T_VALUEL
         ST    R2,T_P5
         LA    R2,T_TOKEN
         ST    R2,T_P6
         OI    T_P6,X'8Ø'            LAST ENTRY
         XC    T_TOKEN,T_TOKEN       ZERO
         L     R15,ADDR441
         LA    R1,T_PARML            ADDRESS OF PARAMETER LIST
         BALR  R14,R15
RETURN   L     R13,4(R13)            RESTORE R13
         L     R1,8(R13)
         FREEMAIN R,LV=WORKL,A=(R1)
         L     R14,12(R13)
         LM    RØ,R12,2Ø(R13)
         SR    R15,R15               SET UP RC
         BSM   Ø,R14                 RETURN TO MVS AND USE RC=R15
ERROR    EQU   *
         MVC   WTOM,=CL8Ø"ERROR ???'
         L     R2,=A(WTOMLEN)
         STH   R2,WTOML
         LA    R2,WTOMSG
```

```
*        WTO   TEXT=(R2),MF=(E,WTOLIST)
         B     RETURN
WTOLIST  WTO   TEXT=,ROUTCDE=11,MF=L
WTOL     EQU   *-WTOLIST
MASKØ8   DC    X'4021202020202020'
T_HEX    DC    X'FØF1F2F3F4F5F6F7F8F9C1C2C3C4C5C6'
WORK     DSECT
SAVEAREA DS    18F
WTOA     DS    CL(WTOL)
CL3      DS    CL3
CL4      DS    CL4
CL6      DS    CL6
CL8      DS    CL8
DOUBLE   DS    D
ASTYPE   DS    CL1
ASNAME   DS    CL8
ASASIDX  DS    CL4                      ASID HEX
ASASIDD  DS    CL5                      ASID DECIMAL
WTOMSG   DS    ØF
WTOML    DS    H
WTOM     DS    CL8Ø
WTOMLEN  EQU   *-WTOM
ADDR_MOD DS    F
ADDR_SRB DS    F
ADDR_PRM DS    F
ADDR_TAR DS    F
MYASCB   DS    F
MYTCB    DS    F
MYASID   DS    XL2
ADDR441  DS    F                        ADDRESS OF IKJCT441
*                                       IKJCT441 - PARM LIST
T_PARML  DS    ØF
T_P1     DS    F
T_P2     DS    F
T_P3     DS    F
T_P4     DS    F
T_P5     DS    F
T_P6     DS    F
T_ECODE  DS    F                        ENTRY CODE
T_NAMEA  DS    F                        ADRRESS OF VARIABLE NAME
T_NAMEL  DS    F                        LENGTH  OF VARIABLE NAME
T_VALUEA DS    F                        ADRRESS OF VARIABLE VALUE
T_VALUEL DS    F                        LENGTH  OF VARIABLE VALUE
T_TOKEN  DS    F                        TOKEN
VARNAME  DS    CL2Ø
VARVALUE DS    CL64
V_CL1Ø   DS    CL1Ø
V_PL8    DS    PL8
ASID     DS    F
ADDR     DS    F
LEN      DS    F
WORKL    EQU   *-WORK
```

```
PARM     DSECT                          PARM FOR SRB ROUTINE
M_ECB    DS    F
M_ASCB   DS    F                        ASCB ADDRESS OF CALLER
M_SAVE   DS    18F                      SAVE AREA
M_ADDR   DS    F                        TARGET AREA START ADDRESS
M_LEN    DS    F                        TARGET AREA LENGTH
M_FIELD  DS    CL64                     RETURN FIELD
VAL_R14  DS    F
PARM_LEN EQU   *-PARM
         REGISTER
         IHAPSA
         CVT DSECT=YES
         IHAASVT
         IHAASCB
         IHAASXB
         IHASRB
SRBLEN   EQU   *-SRB
         IKJTSVT
         END
```

SRB ROUTINE: XTSOMES2

```
XTSOMES2 CSECT
         LR    R12,R15
         USING XTSOMES2,R12
         B     CONTØ1
         DC    CL8"XTSOMES2'
         ABEND 4Ø95,DUMP
CONTØ1   DS    ØH
*        LR    R9,R14                   SAVE RETURN ADDRESS
         LR    R4,R1                    SAVE PARM ADDRESS
         USING PARM,R4
         ST    R14,VAL_R14
         LA    R13,M_SAVE
         L     R6,M_ADDR
         L     R7,M_LEN
         LA    R8,M_FIELD
         LR    R9,R7
         MVCL  R8,R6
         L     R2,M_ASCB                LOAD CALLER ASCB ADDRESS
         POST  M_ECB,BRANCH=YES,,ASCB=(R2),ERRET=LABØ1
         L     R14,VAL_R14
*        LR    R14,R9                   RESTORE RETURN ADDRESS
         BR    R14                      RETURN
LABØ1    EQU   *
         ABEND 4Ø94,DUMP
PARM     DSECT
M_ECB    DS    F
M_ASCB   DS    F                        ASCB ADDRESS OF CALLER
M_SAVE   DS    18F                      SAVE AREA
M_ADDR   DS    F                        TARGET AREA START ADDRESS
```

```
M_LEN    DS    F                        TARGET AREA LENGTH
M_FIELD  DS    CL64                     RETURN FIELD
VAL_R14  DS    F
         REGISTER
         CVT DSECT=YES
         IHALDA
         END
```

IMPLEMENTATION OF XTSOMES1

For installation:

- XTSOMES1 must be linked edited with AC=1 in an authorized library.
- The IKJTSO*xx* PARMLIB member must be updated to include an XTSOMES1 entry in the AUTHCMD section.

USE OF XTSOMES1

Before calling XTSOMES1, you should define three REXX variables:

- XTSOMEM_ASID – the decimal ASID value of the target address space.
- XTSOMEM_ADDR – the decimal value of the starting address.
- XTSOMEM_LEN – the decimal value of area length (up to 64 bytes).

A typical use of XTSOMES1 is shown below:

```
/* REXX */
numeric digits 15
lda_addr = 7ff16ea0
addr = x2d(lda_addr)
say addr
len  =   4
asid =  52
field  = xtsomem(asid,addr,len)
say field
exit
xtsomem:
 arg a1,a2,a3
 xtsomem_asid = a1
 xtsomem_addr = a2
 xtsomem_len  = a3
 xtsomes1
return xtsomem_field
```

Sample ISPF application using XTSOMES1

In order to show how to use XTSOMES1, I wrote a sample REXX/ISPF application scanning various control blocks located in 'common' (PSA, CVT, ASVT, ASCB), but also in 'private' (ASXB, TCB, JSCB, JCT) storage.

ASINFO

```
/* REXX */
/*                                                                   */
/* REXX to scan all active address spaces                            */
/*                                                                   */
parse upper arg parm
if parm ¬= "" then call parse_parm parm
numeric digits(15)
/*                                                                   */
/* control blocks structure:                                         */
/* =========================                                         */
/*                                                                   */
/* PSA => CVT =>                                                     */
/*               ASVT => ASCB                                        */
/*               ASVT => ASCB => ASXB => TCB => JSCB => JCT          */
/*               ....                                                */
   /* default values                                 */
skey   =  N
   /* main routine                                   */
disp_rc = 0
do while disp_rc <= 4         /* until PF3 */
  "ispexec tbcreate tabasinf keys(name)
    names(asid uid pgm jsd jst ssd sst) nowrite"
  "ispexec control display lock"
  function = "Address Spaces Scan... "
  "ispexec display panel(asinfow)"
  call get_info
  call sort_table
  "ispexec tbdispl   tabasinf panel(asinfo)"
  disp_rc = rc
  "ispexec tbclose   tabasinf"
  "ispexec tberase   tabasinf"
end
exit
get_info:
  /*=================================================================*/
  /* PSA  - offsets are decimal                                      */
  /*      - subpool: 239       - common                              */
  /*=================================================================*/
  o_flccvt   =  00016                          /* psa   => cvt       */
  /*=================================================================*/
  /* CVT  - offsets are decimal                                      */
```

```
/*      - subpool: nucleus   - common                                 */
/*====================================================================*/
o_cvtprodn = -00040                          /* sp level           */
o_cvtprodi = -00032                          /* sp fmid            */
o_cvtasvt  =  00556                          /* cvt   => asvt      */
/*====================================================================*/
/* ASVT - offsets are decimal                                         */
/*      - subpool: 245       - common                                 */
/*====================================================================*/
o_asvtasvt =  00512                          /* asvt acronym       */
o_asvtmaxu =  00516                          /* max number of as   */
o_asvtenty =  00528                          /* asid entries       */
/*====================================================================*/
/* ASCB - offsets are decimal                                         */
/*      - subpool: 245       - common                                 */
/*====================================================================*/
o_ascbascb =  00000                          /* ascb acronym       */
o_ascbasid =  00036                          /* asid               */
o_ascbjbns =  00176                          /* pointer to jobname */
o_ascbasxb =  00108                          /* pointer to asxb    */
o_ascbrctp =  00124                          /* pointer to rct tcb */
/*====================================================================*/
/* ASXB - offsets are decimal                                         */
/*      - subpool: 255       - private                                */
/*====================================================================*/
o_asxbasxb =  00000                         /* asxt acronym         */
o_asxbftcb =  00004                         /* pointer to first tcb */
o_asxbltcb =  00008                         /* pointer to last  tcb */
o_asxbuser =  00192                         /* userid               */
/*====================================================================*/
/* TCB  - offsets are decimal                                         */
/*      - subpool: 253       - private                                */
/*====================================================================*/
o_tcbjscbb =  00181                          /* pointer to jscb     */
o_tcbtcbid =  00256                          /* tcbid acronym       */
/*====================================================================*/
/* JSCB - offsets are decimal                                         */
/*      - subpool: 253       - private                                */
/*====================================================================*/
o_jscbjcta =  00261                          /* pointer to jct     */
o_jscbpgmn =  00360                          /* job step pgm name  */
/*====================================================================*/
/* JCT  - offsets are decimal                                         */
/*      - subpool: 236       - private                                */
/*====================================================================*/
o_jctjname =  00008                           /* jobname           */
o_jctjtptn =  00016                           /* terminal name     */
o_jctjmrss =  00143                           /* step start time   */
o_jctjmrjt =  00146                           /* job start time    */
o_jctjmrjd =  00149                           /* job start date    */
o_jctssd   =  00157                           /* step start date   */
/*********************************************************************/
```

```
p_cvt       =  d2x(o_flccvt)
a_cvt       =  d2x( c2d(storage(p_cvt,4)) )
   /* get sp level                                  */
a_cvtprodn  = d2x( x2d(a_cvt) + o_cvtprodn )
v_cvtprodn  = storage(a_cvtprodn,8)
if debug = "Y" then say   v_cvtprodn
   /* get sp fmid                                   */
a_cvtprodi  = d2x( x2d(a_cvt) + o_cvtprodi )
v_cvtprodi  = storage(a_cvtprodi,8)
p_asvt      = d2x( x2d(a_cvt) + o_cvtasvt )
a_asvt      = d2x( c2d(storage(p_asvt,4)) )
a_asvtasvt  = d2x( x2d(a_asvt) + o_asvtasvt )
v_asvtasvt  = storage(a_asvtasvt,4)
   /* get max number of address spaces              */
a_asvtmaxu  = d2x( x2d(a_asvt) + o_asvtmaxu )
v_asvtmaxu  = storage(a_asvtmaxu,4)
max_as      = c2d(v_asvtmaxu)
   /* get asid entries                              */
a_asvtenty  = d2x( x2d(a_asvt) + o_asvtenty )
p_entry     = a_asvtenty
do i = max_as to 1 by -1
     /* get ascb for this asid                      */
  p_ascb      = p_entry
  v_p_ascb = right(d2x( c2d(storage(p_ascb,4))),8,'0')
  first_byte =  substr(v_p_ascb,1,1)
                                     /* high order bit off ? */
                                     /* yes = valid entry    */
  if c2d(bitand(first_byte,'8"x)) = 0 then
   do
     a_ascb      = d2x( c2d(storage(p_ascb,4)) )
     ascb_address = right(a_ascb,8,'0')
     /* get ascb acronym          */
     a_ascbascb  = d2x( x2d(a_ascb) + o_ascbascb )
     v_ascbascb  = storage(a_ascbascb,4)
     /* get ascb asid             */
     a_ascbasid  = d2x( x2d(a_ascb) + o_ascbasid )
     v_ascbasid  = storage(a_ascbasid,2)
     asid_dec = c2d(v_ascbasid)
     /* get jobname / stc name    */
     a_ascbjbns  = d2x( x2d(a_ascb) + o_ascbjbns )
     v_ascbjbns  = storage(a_ascbjbns,4)
     p_jobname   = d2x( c2d( v_ascbjbns ) )
     v_jobname   = storage(p_jobname,8)
     /* Point to asxb            */
     p_asxb      = d2x( x2d(a_ascb) + o_ascbasxb )
     a_asxb      = d2x( c2d(storage(p_asxb,4)) )
     asxb_address = right(a_asxb,8,'0')
     /* get asxb acronym in private area */
     /*                     ============ */
     a_asxbasxb  = d2x( x2d(a_asxb) + o_asxbasxb )
     addr_dec    = X2D(a_asxbasxb)
     len_dec     =   4
```

```
v_asxbasxb     = xtsomem(asid_dec,addr_dec,len_dec)
/* get userid       in private area */
/*                     ============ */
a_asxbuser  = d2x( x2d(a_asxb) + o_asxbuser )
addr_dec    = X2D(a_asxbuser)
len_dec     =   8
v_asxbuser  = xtsomem(asid_dec,addr_dec,len_dec)
/* get rct tcb pointer  in private area */
/*                         ============ */
a_asxbltcb  = d2x( x2d(a_asxb) + o_asxbltcb )
addr_dec    = X2D(a_asxbltcb)
len_dec     =   4
v_asxbltcb     = xtsomem(asid_dec,addr_dec,len_dec)
a_tcb       = d2x( c2d(v_asxbltcb) )
tcb_address = right(a_tcb,8,'Ø')
/* get tcb  acronym in private area */
/*                     ============ */
a_tcbtcbid  = d2x( x2d(a_tcb) + o_tcbtcbid )
addr_dec    = X2D(a_tcbtcbid)
len_dec     =   4
v_tcbtcbid     = xtsomem(asid_dec,addr_dec,len_dec)
/* get jscb pointer  in private area */
/*                      ============ */
a_tcbjscbb  = d2x( x2d(a_tcb) + o_tcbjscbb )
addr_dec    = X2D(a_tcbjscbb)
len_dec     =   3
v_tcbjscbb  = xtsomem(asid_dec,addr_dec,len_dec)
a_jscb      = d2x( c2d(v_tcbjscbb) )
jscb_address = right(a_jscb,8,'Ø')
/* get step program name in private area */
/*                          ============ */
a_jscbpgmn  = d2x( x2d(a_jscb) + o_jscbpgmn )
addr_dec    = X2D(a_jscbpgmn)
len_dec     =   8
v_jscbpgmn  = xtsomem(asid_dec,addr_dec,len_dec)
/* get jct  pointer  in private area */
/*                      ============ */
a_jscbjcta  = d2x( x2d(a_jscb) + o_jscbjcta )
addr_dec    = X2D(a_jscbjcta)
len_dec     =   3
v_jscbjcta  = xtsomem(asid_dec,addr_dec,len_dec)
a_jct       = d2x( c2d(v_jscbjcta) + 16 )
jct_address = right(a_jct,8,'Ø')
/* get jobname           in private area */
/*                          ============ */
a_jctjname  = d2x( x2d(a_jct ) + o_jctjname )
addr_dec    = X2D(a_jctjname)
len_dec     =   32
v_jctjname  = xtsomem(asid_dec,addr_dec,len_dec)
/* get terminal name     in private area */
/*                          ============ */
a_jctjtptn   = d2x( x2d(a_jct ) + o_jctjtptn  )
```

```
addr_dec     = X2D(a_jctjtptn )
len_dec      =   8
v_jctjtptn   = xtsomem(asid_dec,addr_dec,len_dec)
/* get job start date    in private area */
/*                          ============ */
a_jctjmrjd  = d2x( x2d(a_jct ) + o_jctjmrjd )
addr_dec    = X2D(a_jctjmrjd)
len_dec     =   3
v_jctjmrjd  = xtsomem(asid_dec,addr_dec,len_dec)
vv             = c2x(v_jctjmrjd)
job_start_date = substr(vv,1,2) || «.» || substr(vv,3,3)
/* get job start time    in private area */
/*                          ============ */
a_jctjmrjt  = d2x( x2d(a_jct ) + o_jctjmrjt )
addr_dec    = X2D(a_jctjmrjt)
len_dec     =   3
v_jctjmrjt  = xtsomem(asid_dec,addr_dec,len_dec)
vv              = c2d(v_jctjmrjt) % 100
ss              = right(vv // 60 , 2, "0")
vv              = vv % 60
mm              = right(vv // 60 , 2, "0")
hh              = right(vv % 60 , 2 , "0")
job_start_time = hh || «:» || mm || «:» || ss
/* get step start date    in private area */
/*                           ============ */
a_jctssd    = d2x( x2d(a_jct ) + o_jctssd   )
addr_dec    = X2D(a_jctssd)
len_dec     = 3
v_jctssd    = xtsomem(asid_dec,addr_dec,len_dec)
vv             = c2x(v_jctssd)
step_start_date = substr(vv,1,2) || «.» || substr(vv,3,3)
/* get step start time    in private area */
/*                           ============ */
a_jctjmrss  = d2x( x2d(a_jct ) + o_jctjmrss )
addr_dec    = X2D(a_jctjmrss)
len_dec     = 3
v_jctjmrss  = xtsomem(asid_dec,addr_dec,len_dec)
vv              = c2x(v_jctjmrss)
vv              = c2d(v_jctjmrss) % 100
ss              = right(vv // 60 , 2, "0")
vv              = vv % 60
mm              = right(vv // 60 , 2, "0")
hh              = right(vv % 60 , 2 , "0")
step_start_time = hh || «:» || mm || «:» || ss
/*                                       */
 name       = v_jobname
 asid       = asid_dec
 uid        = v_asxbuser
 pgm        = v_jscbpgmn
 jsd        = job_start_date
 jst        = job_start_time
 ssd        = step_start_date
```

```
          sst        = step_start_time
          «ispexec tbadd  tabasinf»
        end
       p_entry     = d2x( x2d(p_entry) + 4 )
     end
return

/*********************************************************************/
parse_parm:
  parse arg parm
  select
   when abbrev("DEBUG",parm,1) then DEBUG = "Y"
   otherwise say parm "invalid option"
  end
return
/*********************************************************************/
/* function to call xtsomes1 module                                  */
/*********************************************************************/
xtsomem:
 arg a1,a2,a3
 xtsomem_asid = a1
 xtsomem_addr = a2
 xtsomem_len  = a3
 xtsomes1
return xtsomem_field
/*********************************************************************/
/* function to sort ispf table                                       */
/*********************************************************************/
sort_table:
  select
   when skey = "N" then
     do
      skeyf = "name"
      sd = "A"
      fields = "fields("skeyf",c,"sd")"
     end
   when skey = "A" then
     do
      skeyf = "asid"
      sd = "A"
      st = "n"
      fields = «fields("skeyf","st","sd")"
     end
   when skey = "JD" then
     do
      skeyf1 = "jsd"
      sd1 = "A"
      skeyf2 = "jst"
      sd2 = "A"
      fields = "fields("skeyf1",c,"sd1","skeyf2",c,"sd2")"
     end
   when skey = "SD" then
```

```
     do
      skeyf1 = "ssd"
      sd1 = "A"
      skeyf2 = "sst"
      sd2 = "A"
      fields = "fields("skeyf1",c,"sd1","skeyf2",c,"sd2")"
     end
   otherwise
  end
  "ispexec tbsort    tabasinf " fields
return
```

ASINFO PANEL

```
)Attr Default(%[_)
 ' type(text) intens(low)  color(green) hilite(reverse)
 ? type(output) intens(low) caps(off) color(turq)
 # type(output) intens(high) caps(off) color(yellow)
 £ type(output) intens(low) caps(off) color(green)
 } type(output) intens(high) color(green)
 { type(text) intens(high)  color(turq)
 | type(text) intens(low) color(green)
 ] type(text) intens(low) color(red)
)Body Expand(çç) Width(&ZSCREENW)
%-o-o-'Address Spaces Info%-o-o-
%Command ===>_ZCMD                                  o o%Scroll ===>_amt [
%Sort Key%===>_z | N[/|A[/|JD[/|SD
%
  [                                Job    Job      Step   Step
  [Name     Asid Userid   Program  Start  Start    Start  Start
  [         Dec                    Date   Time     Date   Time
%  ———— —— ———— ———— ——— ———— ——— ————
)Model
  #name    ?asid?uid     ?pgm     ?jsd   ?jst     ?ssd   ?sst
)Init
  &tsel = ""
  .HELP = asinfoh
  .ZVARS = "(skey)"
)Proc
)End
```

ASINFOW WAIT PANEL

```
)attr
 £ TYPE(TEXT)   INTENS(LOW)  COLOR(TURQ)
 [ TYPE(TEXT)   INTENS(HIGH) COLOR(GREEN)  HILITE(BLINK)
 } TYPE(TEXT)   INTENS(LOW)  COLOR(BLUE)
 ( TYPE(TEXT)   INTENS(HIGH) COLOR(RED)
 ) TYPE(OUTPUT) INTENS(HIGH) COLOR(PINK) hilite(blink)
 { type(text) intens(high) color(yellow)
 | type(text) intens(low)  color(turq)
)Body Expand(??) Width(&ZSCREENW)
```

```
[WW        WW  OOOOOOOOOOOO  RRRRRRRRRRR   KK        KK    {Time     -¨&ZTIME
[WW        WW  OOOOOOOOOOOO  RRRRRRRRRRRR  KK       KK     {Date     -¨&ZDATE
[WW        WW  OO        OO  RR        RR  KK      KK      {Julian   -¨&ZJDATE
[WW        WW  OO        OO  RR        RR  KK     KK       {System   -¨&ZSYSID
[WW        WW  OO        OO  RR        RR  KK    KK
[WW        WW  OO        OO  RRRRRRRRRRR   KKKKKKK       ______
[WW   WW   WW  OO        OO  RRRRRRRRRRR   KKKKKKK       ______
[WW  WWWW  WW  OO        OO  RR    RR      KK    KK
[WW WW  WW WW  OO        OO  RR     RR     KK     KK
[WWWW    WWWW  OO        OO  RR      RR    KK      KK
[WWW      WWW  OOOOOOOOOOOO  RR       RR   KK       KK
[WW        WW  OOOOOOOOOOOO  RR        RR  KK        KK
                                          [IIIIIIIIII   NN        NN   GGGGGGGGGG
                                         [IIIIIIIIII   NNN       NN  GGGGGGGGGGGG
                                            [II       NNNN      NN  GG        GG
(Please be patient, performing:           [II       NN NN     NN  GG
                                          [II       NN  NN    NN  GG
                                         [II       NN   NN   NN  GG
)FUNCTION                               [II       NN    NN  NN  GG     GGGGG
                                       [II       NN     NN NN  GG     GGGGG
                                      [II       NN      NNNN  GG        GG
                                     [II       NN       NNN  GG        GG
                                [IIIIIIIIII   NN        NN  GGGGGGGGGGG
{SMP/E SMPOUT processing       [IIIIIIIIII   NN         N   GGGGGGGGGG
)init
)proc
)end
```

ASINFOH (HELP) PANEL

```
)Attr Default(%[_)
 ' type(text) intens(low)  color(green) hilite(reverse)
 ? type(output) intens(low) caps(off) color(turq)
 # type(output) intens(high) caps(off) color(yellow)
 £ type(output) intens(low) caps(off) color(green)
 } type(output) intens(high) color(green)
 { type(text) intens(high)  color(turq)
 | type(text) intens(low) color(green)
 ] type(text) intens(low) color(red)
)Body Expand(oo) Width(&ZSCREENW)
%-o-o-'Adress Spaces Info{-o-o-
%Command ===>_ZCMD                                    o o%Scroll ===>_amt [

 ¨This panel shows «Address Spaces Info» for all active address spaces.

  The list can be sorted by:

    ]N [: Address space name (ascending).
    ]A [: Asid decimal value (ascending).
    ]JD[: Job Start Date (ascending).
    ]SD[: Step Start Date (ascending).
```

```
)Init
)Proc
)End
```

ASINFO SAMPLE DISPLAY

```
──────────────── Address Spaces Info ──────── Row 1 to 16 of 52
Command ===>                                                Scroll ===> PAGE
Sort Key ===> N   N / A / JD / SD

                                  Job    Job      Step   Step
  Name     Asid Userid   Program  Start  Start    Start  Start
           Dec                   Date   Time     Date   Time
  ──── ── ──── ──── ─── ──── ─── ────
  *MASTER* 1    *BYPASS* IEEMB86Ø Ø1.1Ø1 13:Ø9:15 Ø1.1Ø1 13:Ø9:15
  ALLOCAS  18                     Ø1.1Ø1 13:Ø9:15 Ø1.1Ø1 13:Ø9:15
  ANTAS ØØØ 13   +ANTAS ØØ ANTXAINI Ø1.1Ø1 13:1Ø:52 Ø1.1Ø1 13:1Ø:52
  ANTMAIN  12   +ANTMAIN ANTMAIN  Ø1.1Ø1 13:1Ø:41 Ø1.1Ø1 13:1Ø:41
  APPC     42   *BYPASS* ATBINITM Ø1.1Ø1 13:12:15 Ø1.1Ø1 13:12:15
  APSWPR72 39   TCP      APSPPIEP Ø1.1Ø1 16:14:53 Ø1.1Ø1 16:14:53
  ASCH     43   *BYPASS* ASBSCHIN Ø1.1Ø1 13:12:15 Ø1.1Ø1 13:12:15
  ASCHINT  47   *BYPASS* IEFIIC   Ø1.1Ø1 13:12:21 Ø1.1Ø1 13:12:21
  BPXAS    32   TCP      BPXPRFC  Ø1.1Ø1 13:13:24 Ø1.1Ø1 13:13:24
  BPXOINIT 4Ø1  OMVSKERN BPXPINPR Ø1.1Ø1 13:11:3Ø Ø1.1Ø1 13:11:3Ø
  CATALOG  3Ø   +CATALOG IGGØCLXØ Ø1.1Ø1 13:1Ø:42 Ø1.1Ø1 13:1Ø:42
  CMF      48   *BYPASS* BBM9DAØØ Ø1.11Ø 16:58:Ø1 Ø1.11Ø 16:58:Ø1
  CMFCAS   41   *BYPASS* BBM9ZAØØ Ø1.11Ø 16:57:32 Ø1.11Ø 16:57:32
  CONSOLE  1Ø   *BYPASS*          Ø1.1Ø1 13:Ø9:15 Ø1.1Ø1 13:Ø9:15
```


SMP/E for z/OS and OS/390 Version 3 Release 1

IBM has announced SMP/E Version 3 Release 1, the software installation and maintenance tool for z/OS and OS/390 that maintains an inventory of the installed software and service. It is now available under its own product number as well as remaining a base element of z/OS, allowing sites with a currently-supported release of z/OS or OS/390 to order and install the latest release of SMP/E without having to upgrade the OS. It will cost nothing for customers with licences for OS/390 Version 2 Release 6 or later, or z/OS Version 1 Release 1 or later (*http://www.ibm.com/servers/eserver/zseries*).

Java basics

IBM has devoted considerable resources to promoting Java both on the mainframe and throughout its product range. In doing so IBM has, in essence, established Java as one of the principal programming languages for both OS/390 and z/OS. Furthermore, Java performance has been radically improved with the effort IBM is putting into its Just-In-Time (JIT) compiler technology.

Users can of course use NetREXX to bypass Java programming to a certain extent, but Java skills are highly sought after and can be a major asset in the employment market. For those of you who wish to explore Java for the mainframe there are innumerable resources available to get you started. The IBM DeveloperWorks Web site is possibly the most helpful: *http://www-106.ibm.com/developerworks/java/*.

Here you can find some excellent free tutorials. To use these you will need to register and will need Netscape 4.*x* or higher, or Internet Explorer 4.*x* or higher, with JavaScript enabled. It is possible to use the tutorials on-line, or they can be downloaded and used off-line.

Two of the most important tutorials for beginners to Java will be 'Java language essentials' and 'Building a Java applet'. They assume no prior knowledge in Java. These tutorials introduce the Java programming language. They include examples that demonstrate the syntax of the language in an object-oriented framework, along with standard programming practices such as defining instance methods, working with the built-in data types, creating user-defined data types, and working with reference variables.

Once you have gained confidence with Java users should register at IBM's Visual Age Developer Domain Web site (*http://www.ibm.com/software/vadd*), where it is possible to download a free copy of Visual Age for Java Entry Professional Edition.

New IBM Redbooks

INTRODUCTION

The last six months has seen the publication of a wide range of IBM Redbooks which are a 'must read' for those considering z/OS and mainframe Linux in their enterprises. A brief summary of some of the more important mainframe-oriented Redbooks is provided below.

Linux for zSeries and S/390: Distributions (SG24-6264-00)

This Redbook, published in September 2001, describes the different Linux distributions available for zSeries and S/390 hardware. It provides useful information to help users install, customize, and maintain Linux on a mainframe. The Redbook covers the following Linux distributions: SuSE Linux Enterprise Server for S/390, Turbolinux server for zSeries and S/390, Red Hat Linux for S/390, Marist file system, Caiman Linux, and Think Blue Linux. The Redbook covers the installation and use of Linux images in a logical partition (LPAR), under z/VM and under the Virtual Image Facility (VIF). Additionally, the Redbook provides information on managing DASD and file systems, the Logical Volume Manager (LVM), debugging, LDAP, Systems management, security, back-up, and restore.

z/OS Intelligent Resource Director (SG24-5952-00)

This Redbook, published on 15 August 2001, describes the new LPAR clustering technology, available on the IBM zSeries processors, and z/OS. The book is composed of three parts: dynamic CHIPD management, I/O priority queueing, and CPU management. Each part has an introduction to the new function, planning information to help you assess and implement the function, and management information to help you monitor, control, and tune the function in your environment. Considering the importance of the Intelligent Resource Director for both z/OS and the zSeries and the absence of much information, this Redbook fills a crucial gap (ISBN: 0738417904).

NETWORKING

For those users who are involved in mainframe networking there are a number of interesting new Redbooks, as follows.

FICON Native Implementation and Reference Guide (SG24-6266-00)

This Redbook, published on 7 August 2001, covers the planning and implementation of FICON channels, operating in FICON native (FC) mode for the z900 and 9672 Generation 5 (G5) and Generation 6 (G6) processors. It discusses the FICON and Fibre Channel architectures, terminology, and supported topologies. This book provides information about the principal FICON native products, system and I/O device set-up, availability and recovery considerations, and migration recommendations. It focuses on installing the new FICON Directors and FICON native control units in both a new and existing ESCON and FICON channel environment. There is also some really useful information about monitoring and managing a FICON native (FC) environment (ISBN: 0738422630).

TCP/IP in a Sysplex (SG24 5235 02)

This Redbook, published on 2 April 2001, focuses primarily on the Sysplex Distributor. It is designed to allow readers to produce an OS/390-based IP network which will gain the maximum benefit from the features available with the Sysplex. The book discusses three Sysplex-specific solutions that help to meet these demands, Sysplex Distributor, Domain Name Service/Workload Manager, and Network Dispatcher. All of these solutions, to some extent, make use of the MVS Workload Manager (WLM), so there is some analysis of WLM. Also, because the Sysplex high availability is closely tied with Virtual IP Addressing (VIPA), this is covered in some detailed including a detailed routing discussion as we deal with VIPAs in the Sysplex (ISBN: 0738421219).

Secure e-business in TCP/IP Networks on OS/390 and z/OS (SG24-5383-01)

This Redbook, published on 12 June 2001, shows users how to secure network access and TCP/IP applications on an OS/390 or z/OS platform, with practical examples of various configurations and

implementations. Unlike many of the other Redbooks, this volume does not focus heavily on detailed implementation information, but provides general solutions that will allow users to establish a secure e-business environment using OS/390 or z/OS. Some of the issues covered include: Unix System Services security, TCP/IP stack security, TCP/IP application security, SecureWay Security Server Firewall Technologies, and simplified security scenarios. There are also six useful appendices (ISBN: 0738421561).

CONCLUSION

These Redbooks can be purchased from IBM in hardcopy or CD ROM format, but they are also available in Adobe PDF format and may be viewed online or downloaded for offline viewing and printing for free. Do remember that the PDF files can range from 3-6 MB so can take a while to download if you have just simple dial-up Internet access. If you have not done so already, visit the IBM Redbook Web site at the following URL, it is well worth it: *http://www.redbooks.ibm.com*

COBOL Unit Tester

If you need to test System/390 COBOL applications outside of your work environment, where you do not have access to a System/390 box then this free software from IBM's Alphaworks Web site is a must.

Originally released in May 2001, with some bug fixes added in July, the 'COBOL Unit Tester' enables users to perform I/O data simulation of System/390 COBOL applications on a PC without a real run-time environment or a connection with a System/390 host. This software is specifically designed to test individual compilation units before they are linked together and executed on the System/390 platform.

The COBOL Unit Tester is available for Windows NT and Windows 2000 for free download at the following URL:*http://www.alphaworks.ibm.com/tech/cobol*

November 1998 – October 2001 index

Items below are references to articles that have appeared in *MVS Update* since November 1998. References show the issue number followed by the page number(s). Back issues of *MVS Update* are available back to Issue 136 (January 1998).

Computer Associates has announced availability of enhanced versions of Unicentre NetSpy Network Performance and Unicentre NetMaster Network Management for TCP/IP for ensuring network availability and performance of mainframe applications. NetSpy Network Performance Version 6.0, formerly known as CA-NetSpy and NetworkIT NetSpy, provides an integrated view of both TCP/IP and SNA mainframe networks, putting out detailed performance metrics that help detect and correct problems before they can impact activity. The new release provides broad TCP/IP performance instrumentation to help understand network utilization, and monitor service levels.

Meanwhile, NetMaster for TCP/IP 6.2, formerly known as SOLVE:NetMaster for TCP/IP and NetworkIT NetMaster for TCP/IP, includes an integrated set of diagnostics, performance reporting, and access control capabilities to help maintain network connections to mainframe applications and data through proactive monitoring, diagnosis, and performance management of TCP/IP network devices and events.

For further information contact:

Computer Associates International, Inc, One Computer Associates Plaza, Islandia, NY 11749, USA.
Tel: (631) 342 6000
Fax: (631) 342 6800

Computer Associates plc, Ditton Park, Riding Court Road, Datchet, Slough, Berkshire, SL3 9LL, UK.
Tel: 01753 577733
Fax: 01753 825464

http://www.ca.com

* * *

Network Associates has announced it has ported its PGP E-Business Server to OS/390. The PGP E-Business Server for OS/390 version 7.1, protects data in storage, transit and during access. It is aimed at sites that process batch transactions.

The product provides strong encryption, authentication and compression technology to protect data and help improve throughput of data transfer. Said to be easily installed and configured, it is transparent to users. It encrypts information using public key or symmetric encryption and includes a digital signature for authentication and data integrity.

Self-Decrypting Archives enable recipients that do not use PGP encryption to decrypt and access the files. For added security, it supports two-factor SmartCards and tokens, providing two-factor security.

Besides smart card support, users get interoperability with PGP Security or x.509 certificate-based products. The product also leverages IBM's S/390 crypto acceleration technology.

For further information contact:

Network Associates, 227 Bath Road, Slough Berkshire, SL1 5PP, UK.
Tel: 01753 217500
Fax: 01753 217520

Corporate Headquarters
3965 Freedom Circle
Santa Clara, CA 95054, USA.
Tel: 972 308 9960
Fax: 408 970 9727

http://www.pgp.com

* * *