October 1997

In this issue

3 Disk unit information display
16 Listing APF-authorized libraries
18 Making global changes to PDS members
28 A volume mount analyser
33 Generating structured Assembler programs with ISPF edit macros – part 1
48 Enabling/disabling cache and NVS
51 Enhanced Assembler utilities – part 1
72 MVS news

© Xephon plc 1997
MVS Update

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

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

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

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

© Xephon plc 1997. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.
INTRODUCTION
In large shops, there is often a mix of disks from several vendors. Usually somewhere there is a table stating which string was manufactured by which vendor. There can be considerable problems if there is a hardware failure and the systems programmer or operators urgently require this information after-hours. It is therefore convenient to be able to access the material on-line.

The following program will provide this information. In prompting the disk control unit, it gets a considerable amount of information back, including the (3-byte) name of the manufacturer, the serial number, the subsystem number, etc. It runs as an authorized Assembler routine and for this purpose it needs to be added to IKJTSOxx. It uses a REXX routine and an ISPF panel as a user interface. Because the program runs authorized, it cannot perform any ISPF services. To overcome this, a temporary file is used between the program and the REXX program to pass data to and fro. If the program has been installed correctly and returns the message ‘Could not read controller info’, it means that the controller is probably an old 3880-type and does not support the ‘READ CONFIG DATA’ command.

Install the REXX EXEC, ISPF panel, and load module in the usual places for your site and then update the ISPF options panel to contain:

\[
\text{DT.'CMD(%DISKTYPE)'}
\]

SAMPLE OUTPUT

```
COMMAND ===> ____________ Disk unit information display

Enter the unit number or VOLSER of the device

Unit : 0995
VOLSER: WORK99

Device type : 3390     Control unit type : 3990
```
Device model: B9C  Control unit model: 003
Manufacturer: HTC  Manufacturer: HTC
Sequence num: 000033444502  Sequence num: 00000044323

Subsystem id: 0092
Unit address: 15  (Address director uses on channel interface)
Phys address: 15  (Address storage path uses)

ISPF PANEL

)ATTR
_TYPE(INPUT) CAPS(ON) PADC(_)
_TYPE(OUTPUT) COLOR(GREEN)
_TYPE(OUTPUT) COLOR(green) intens(low)
_TYPE(text) COLOR(white) intens(low)
_TYPE(text) COLOR(blue) intens(low)

)BODY WINDOW(72,23)

% >Disk unit information display%
%COMMAND --->_ZCMD
%
%
#Enter the unit number or VOLSER of the device
*
*Unit : _unit+
*VOLSER: _volser+
%
%Device type : @devtype
%Device model : @model
%Manufactuer: @manufac
%Sequence num: @serial

%Subsystem id: @subsysid
%Unit address: @unitad +(Address director uses on channel interface)
%Phys address: @physad +(Address storage path uses)
%
@error

)INIT
.CURSOR = unit
)PROC
IF (&ZCMD = ' ') .MSG = ISPZ001 /* INVALID COMMAND */
IF (&volser = '')
   &error = ''
   VER(&unit NB)
   .CURSOR = unit
)END

© 1997. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
REXX EXEC

/* REXX*/
/* This utility calls SYS4.LINKLIB(DISKINFO) to obtain info for one */
/* disk at a time. DISKINFO uses the READ CONFIGURATION CCW to obtain */
/* the info. The problem is that DISKINFO needs to be authorized to do*/
/* this, so it cannot use any ISPF services. An intermediate file is */
/* used for the purpose. Once the panel has been filled in by the user*/
/* a parm is put into the file in the format Uxxxx or Vyyyy, with */
/* Uxxxx indicating Unit followed by a 4-byte address or Vyyyy in- */
/* dicating Volser and a 6-byte volume. DISKINFO picks this parm up */
/* and returns the info in the following format: */
/* */
/* Bytes 1-7 : input parm, untouched */
/* Bytes 8-11 : 4-byte unit number */
/* Bytes 12-17 : 6-byte volser */
/* Bytes 18-272 : 255 bytes of info as returned by the controller */
/* Bytes 273-227: 55 bytes of error message (if required) */
/* Bytes 228-231: 4-byte printable subsystem id */
/* Bytes 232-233: unit address in printable format */
/* Bytes 234-235: physical device address in printable format */
/* */
/* ******************************************************* */
if sysvar(sysispf) = active then do
  address ispexec "CONTROL ERRORS RETURN"
  "ALLOC FI(disktype) NEW SPACE(l) TRACK LRECL(335) RECFM(F B)
   BLKSIZE(335) REUSE UNIT(VIO)"
  do while RC = 0
    oldunit = unit
    oldvol = volser
    address ispexec "DISPLAY PANEL(DISKTYPE)"
    if RC > 0 then leave
    if oldunit = unit & oldvol = volser
      then unit = ""
    else if oldunit = unit & unit = ""
      then volser = ""
    else
      error = ""
    if unit = "" then do
      unit = right(unit,4)
      parm = U || unit
    end
    else
      do
        volser = right(volser,6)
        parm = V || volser
      end
    queue parm
    "execio 1 diskw disktype (finis"
  address tso "CALL 'SYS2.LINKLIB(DISKINFO)'
  callrc = RC

if callrc = 16 then
  do
    error = "Could not open parm file - contact software support"
    iterate
  end
else if callrc = 12 then
  do
    error = "Input parm file empty - contact software support"
    iterate
  end
"execio 1 diskr disktype (STEM info. FINIS"
if callrc > 0 then
  do
    error = substr(info.1, 272)
    devtype = ""
    model = ""
    manufac = ""
    serial = ""
    cutype = ""
    cumodel = ""
    cmanufac = ""
    cserial = ""
    subsysid = ""
    unitad = ""
    physad = ""
    iterate
  end
  unit = substr(info.1, 8, 4)
  volser = substr(info.1, 12, 6)
  devtype = substr(info.1, 22, 6)
  model = substr(info.1, 28, 3)
  manufac = substr(info.1, 31, 3)
  serial = substr(info.1, 36, 12)
  cutype = substr(info.1, 88, 4)
  cumodel = substr(info.1, 92, 3)
  cmanufac = substr(info.1, 95, 3)
  cserial = substr(info.1, 100, 12)
  subsysid = substr(info.1, 328, 4)
  unitad = substr(info.1, 332, 2)
  physad = substr(info.1, 334, 2)
"FREE FI(disktype)"
end
else do
  address tso "ISPSTART CMD(%DISKTYPE)"
end
**ASSEMBLER ROUTINE**

------------------------------------------------------------------------
* This routine supports the DISKTYPE (DT) option under ISPF.
* It runs in supervisor state key 0 for part of the time, so it is not allowed to do any ISPF calls. It communicates to REXX via a file.
* When it starts up, it reads the file (DD-name PARMFILE) and looks at the first byte. If it contains a "U", it takes the next 4 bytes as the unit number; if it contains a "V" it takes the next 6 bytes as a volser. It then allocates the volume and issues an FA (Read Config Data) to the controller. This info (255 bytes) is then appended to the first 7 bytes (with a PUTX) and some of the fields are made printable - refer to the CALLPARM DSECT for the record layout. It then closes the PARMFILE, frees the VTOC and returns to the caller.

* INPUT:  File, refer CALLPARM DSECT (DD-name of PARMFILE)
* OUTPUT: PUTX to file, refer CALLPARM DSECT, ditto
* AMODE: 24
* RMODE: 24
* Program attr: RENT, AC=1
* Called Routines: None
* Macros: DYNALLOC, UCBLOOK, OPEN, CLOSE, GET, PUTX etc.
* DD-cards: PARMFILE (static) and VTOC of disk (dynamic)
* Special Regs: R4 - pointer to data read in from PARMFILE
* R12 - Base register
* R13 - Pointer to general Savearea and workareas
* All other registers general purpose
* Error messages: Error messages moved into PARMFILE at field name ERROR.
* Return Codes: RC=0 all normal completion, result in PARMFILE
* RC=8 specified unit address not on-line
* RC=8 specified volume not on-line
* RC=8 could not allocate the volume
* RC=12 PARMFILE does not contain data
* RC=16 could not open PARMFILE
* Reason for RC=8 returned in PARMFILE for caller
------------------------------------------------------------------------

DISKINFO CSECT
DISKINFO AMODE 24
DISKINFO RMODE 24

BAKR R14,0 .Save Caller's Status
BALR R12,0
USING Load,12

************************************************** Main driver routine
**************************************************

Load LA R3,GetMSize .Our storage requirement's length
Storage STORAGE OBTAIN,LENGTH=(3),LOC=BELOW,BNDRY=DBLWD
LR R2,R1 .Point to getmainted area
LA R3,GetMSize .Size of area we got
XR R9,R9 .Byte to propagate

MVCL R2,R8 .Propagate binary zeroes
USING GetMArea,R1
ST R13,SaveArea+4 .Backchain
DROP R1
LR R13,R1 .Point to getmain area
USING GetMArea,R13 .Addressability to getmain area
BAS R14,GetParm .Go read input unit/ volser
L R15,RetCode .Pick up the return code
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,GetVols .Go determine the VOLSER of the unit
L R15,RetCode .Pick up the return code
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
BNZ SetFile .No, get out
BAS R14,AllocVol .Go allocate & open the VTOC
LTR R15,R15 .Success?
*********************************************************************
* This routine gets the VOLSER for the supplied UCB/ device #
*********************************************************************

GetVols BAKR R14,0
L R4,Data@ .Where the passed parm is
CLI InParm,C'U' .Did we get a unit number?
BE MoveUnit .Yes

MoveVol MVC Volser,1(R4) .No, we were supplied with a volser
OI VolSpec,Yes .Turn the flag on
B GetUnit# .Obtain the unit number to return

MoveUnit MVC Unit,1(R4) .Pick up the (4-byte) unit number

MODESET MODE=SUP,KEY=ZERO
MVC UCBMacA(UCBMacL),UCBMac
LA R1,UCBMacA .Have unit, get volser
UCBLOOK MF=(E,(R1)),DEVMCHR=Unit,UCBPTR=UCB,LOC=ANY, X
   DYNAMIC=YES,NOPIN,RANGE=ALL
LTR R15,R15 .Successful?
BNZ NoVol .Yes
L R2,UCB@

USING UCBOB,R2
TM UCBSTAT,UCBONLI .Is the device on-line?
BNO NoVol .No
TM UCBTBYT3,UCB3DACC .Is it a disk?
BNO NoVol .No
MVC Volser,UCBVOLI .Pick up the volser
B GetVolSX .Yes

NoVol L R4,Data@ .Address where parm data is
MVC Error,-CL55'Unit address not on-line to this system'
LA R15,B .Set the return code to 8
ST R15,RetCode .Plug the return code
B GetVolSX .Get out

GetUnit# MODESET MODE=SUP,KEY=ZERO
LA R1,UCBMacA .Have volser, get unit
UCBLOOK MF=(E,(R1)),VOLSER=Volser,UCBPTR=UCB,LOC=ANY, X
   DYNAMIC=YES,NOPIN,RANGE=ALL
LTR R15,R15 .Successful?
BZ LocUnit .Yes, go move the unit number
MVC Error,-CL55'Cannot obtain the volser for the device'
B GetVolSX .Not a critical error

LocUnit L R1,UCB@

XC Double,Double .Clear our work area
MVC Double(2),4(R1) .Device number (halfword in hex)
MVC Double+4(2),4(R1) .Device number (halfword in hex)
NC Double(2),-2X'F0' .Turn second half of byte off
TR Double(2),FstByte .Make first half printable
NC Double+4(2),-2X'0F' .Turn first half of byte off
TR Double+4(2),SecByte .Make second half printable
MVC Unit(1),Double .First half of first byte
MVC Unit+1(1),Double+4 .Second half of first byte
MVC Unit+2(1),Double+1 .First half of second byte
MVC Unit+3(1),Double+5 .Second half of second byte

GetVolSX MODESET MODE=PROB,KEY=NZERO

This routine allocates the volume's VTOC and opens it.

**AllocVol**

BAKR R14,0
L R4,Data@
LA R5,RB0
ST R5,APRB0
OI APRB0,S99RBPN
USING S99RB,R5
MVI S99RBLN,X'14'
MVI S99VERB,S99VRBAL
OI S99FLG11,S99NOCNV
LA R6,DirPtr0
ST R6,S99TXTPP
DROP R5
USING S99UNIT,R5
LA R5,DirPrm0
ST R5,DirPtr0
MVI S99TUKEY+1,DALRTDDN
MVI S99TUNUM+1,X'01'
MVI S99TULNG+1,X'08'
MVC S99TUPAR(6),Volser
LA R5,DirPrm1
ST R5,DirPtr1
MVI S99TUKEY+1,DALVLSER
MVI S99TUNUM+1,X'01'
MVI S99TULNG+1,X'06'
MVC S99TUPAR(6),Volser
LA R5,DirPrm2
ST R5,DirPtr2
MVI S99TUKEY+1,DALSTATS
MVI S99TUNUM+1,X'01'
MVI S99TULNG+1,X'01'
MVI S99TUPAR,DA0BSHR
MVC S99TUPAR(8),='SYSALLDA'
LA R1,APRB0
DYNALLOC
LTR R15,R15
BZ OpenDsn
LA R15,8
ST R15,RetCode
* If user specified the unit, there is a system error because we
  already managed to read the UCB for the device.
* If the user specified the volser, it is simply not on-line.
TM VolSpec.Yes
BNO UError

© 1997. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
VolError MVC Error,-CL55'Volume not on-line to the system'
B AllocVoX .Get out
UError MVC Error,-CL55'System error - could not allocate the volume'
B AllocVoX .Get out
OpenDsn EQU *
MVC VTOCDCB(VTOCLeng),VTOC
LA R5,DirPrm0 .Text unit containing DDNAME
MVC VTOCDCB+40(8),S99TUPAR Update the DDNAME in the DCB
LA R14,PrepJFCB
BSM 0,R14 .Change addressing mode to 24
PrepJFCB LA R1,JFCBarea .Get address of JFCB area
STCM R1.7,PDJFList+1 .Plug into list
MVI RDJFList,X'07'
LA R1,PDJFList
STCM R1.7,VTOCDCB+37 .Update the JFCB list pointer
RDJFCB LA R2,VTOCDCB .Point to the VTOC
STCM R2.7,PDJFCBP+1
OI RDJFCBP+X'00'
LA R1,PDJFCBP .Point to DCB pointer
SVC 64 .Read the JFCB
LTR R15,R15 .Test if DD-card present
BZ DDOK .DD-card is present
MVC Error,-CL55'RDJFBCB error - contact software support'
LA R15,S .Indicate DD-card missing
ST R15,RetCode .Save return code
B AllocVoX .Return to our caller
DDOK EQU *
LA R2,JFCBAREA .Point to JFCB
USING INFMJFCB,R2 .Establish addressability
MVI JFCBDSNM,X'04' .Create DSNAME
MVC JFCBDSNM+1(43),JFCBDSNM - 44x'04'
OI JFCBTSNM,JFCNWRIT .Do not write JFCB during open
DROP R2
MVC OpenArea(OpenLeng),OpenMac
LA R1,OpenArea
OPEN VTOCDCB,TYPE=J,MF=(E,(1))
TM VTOCDCB+48,X'10' .Did the VTOC open?
BO AllocVoX .Yes
MVC Error,-CL55'Could not open VTOC - contact support'
LA R15,S .Set the return code to 8
ST R15,RetCode .Plug the return code
AllocVoX PR
*********************************************************************
* This routine gets the information from the controller
*********************************************************************
GetInfo BAKR R14,0
MODESET MODE=SUP,KEY=ZERO
L R4,Data@ .Where parm data is
MVC EXCPArea(EXCPLeng),EXCPInfo
* .Update the IOB in g'mained storage
LA R1,ReadECB .Address of ECB to post
ST R1,ECB@ .Plug into ECB pointer

LA R1,CCW .Address of CCW
ST R1,IOBCCWA .Plug into CCW pointer
LA R1,VTODCDB .Address of DCB
ST R1,IOBCCWA+4 .Plug into DCB pointer
LA R1,InArea .Address of output area
STCM R1,7,CCW+1 .Plug into CCW
LA R1,VTODCDB
USING IHADCBA,1 .Addressability to DCB
L R1,DCBDEBAD .Address of DEB basic section
DROP R1
USING DEBBASIC,R1
LA R1,DEBBASND .End of DEB basic section
DROP R1
USING DEBDAASD,R1 .Map DASD section
ICM R1,15,DEBSTRCC .Pick up start of first extent
STCM R1,15,MBBCCHHR+3 .Plug into IOB

**FILEDEB**

LA R2,VTODCDB .Point to DCB
L R1,X'2C'(R2) .DEB pointer
SH R1,-H'8' .DEB - 8
L R3,0(R1) .DEB Extention
OI 3(R3),X'40' .Bypass CCW prefixing
XC ReadECB,ReadECB .Clear the ECB
EXCP IOB .Get info off controller
WAIT ECB=ReadECB .Wait for controller
XR R15,R15
CLI IOB+4,X'7F' .Normal I/O completion?
BNE NoInfo .Yes, get out
OI GotInfo,Yes .Set the flag on
B GetInfoX .Get out

**NoInfo**

MVC Error,-CL55'Could not read controller info'
ICM 3,2,1OB+12
XR R2,R2
IC R2,1OB+4
LA R15,4 .Probably pre-3390 type controller

**GetInfoX**

ST R15,RetCode .Set the return code
MODESET MODE=PROB,KEY=NZERO
PR

*********************************************************************
* This routine closes the VTOC and frees it
*********************************************************************

**FreeVTOC**

BAKR R14,0
MVC ClosArea(ClosLeng),ClosMac
LA R1,ClosArea
CLOSE VTODCDB,ME-(E,(1)) .Re-entrant close

**FreeIt**

LA R5,RB0
USING S99RB,R5
MVI S99VERB,S99VRBUN .UNALLOCATE verb
DROP R5
USING S99UNIT,R5 .SVC99 Text Unit DSECT
LA R5,DirPrm0 .Build DDNAME pointer
MVI S99TUKEY+1,DALDDNAM
O1 DirPtr0,S99RPBND .Mark as last in the list
LA R1,APRB0 .Load parameter pointer for SVC99
DYNALLOC .Do the allocation
LTR R15,R15 .Successful?
BZ FreeVTOX .Yes, get out
MVC Error,-CL55'Volume not freed, contact support'
LA R15,4
ST R15,RetCode

FreeVTOX PR
*********************************************************************
* This routine provides the wanted information
*********************************************************************

UpdtFile BAKR R14,0
LA R1,InArea .Info returned by controller
L R4,Data@ .Where file will be written from
MVC OutParm,O(R1) .Move all supplied info into file
BAS R14,MakePrt .Make hex fields printable
L R1,Data@ .Where the output data is
PUTX ParmDCB .Update the record
MVC ClosArea(ClosLeng),ClosMac
LA R1,ClosArea
CLOSE ParmDCB,ME=(E,(1))

WTOInfX PR
*********************************************************************
* This routine converts HEX fields to printable format
*********************************************************************

MakePrt BAKR R14,0
L R4,Data@ .Where the output data is
MVC Double(2),249(R4) .SSID at offset 232-233 (+ 17)
MVC Double+4(2),249(R4) .SSID at offset 232-233 (+ 17)
NC Double(2),-2X'F0' .Turn second half of byte off
TR Double(2),FrstByte .Make first half printable
NC Double+4(2),-2X'0F' .Turn first half of byte off
TR Double+4(2),SecByte .Make second half printable
MVC PrtSubs(I),Double .Pick up the first byte
MVC PrtSubs+1(I),Double+4 Pick up the second byte
MVC PrtSubs+2(I),Double+1 Pick up the third byte
MVC PrtSubs+3(I),Double+5 Pick up the fourth byte
MVC Double(1),252(R4) .Unit address at offset 235 (+ 17)
MVC Double+4(1),252(R4) .Unit address at offset 235 (+ 17)
NC Double(1),-X'F0' .Turn second half of byte off
TR Double(1),FrstByte .Make first half printable
NC Double+4(1),-1X'0F' .Turn first half of byte off
TR Double+4(1),SecByte .Make second half printable
MVC PrtUadd(I),Double .Pick up the first byte
MVC PrtUadd+1(I),Double+4 Pick up the second byte
MVC Double(1),254(R4) .Unit address at offset 237 (+ 17)
MVC Double+4(1),254(R4) .Unit address at offset 237 (+ 17)
NC Double(1),-X'F0' .Turn second half of byte off
TR Double(1),FrstByte .Make first half printable
NC Double+4(1),-1X'0F' .Turn first half of byte off
TR Double+4(1),SecByte .Make second half printable
MVC PrtPhAdd(I),Double .Pick up the first byte

MVC PrtPhAdd+1(1),Double+4 Pick up the second byte

* Constants follow

VTOC DCB DDNAME=VTOC,DSORG=PS,RECFM=U,KEYLEN=44,
MACRF=E,EXLST=VTOC
VTOCLeng EQU *-VTOC
PARMFILE DCB DDNAME=DISKTYPE,DSORG=PS,MACRF=GL,EODAD=NoData
ParmFill EQU *-PARMFILE
UCBMAC UCBLOOK MF=(L,UCBArea)
UCBMAC EQU *-UCBMAC
OpenMac OPEN (),MF=L
OpenMac EQU *-OpenMac
ClosMac CLOS (),MF=L
ClosMac EQU *-ClosMac
EXCPIInfo DS ØF
DC X'02000000' .Start of IOB
DC CL4 .Address of ECB to post
DC 8X'00' .CSW
DC 4F .IOBCCWA
DC 8X'00' .MBBCCHHR
DC X'FA',5X'00',X'0100'
EXCPIInfo EQU *-EXCPIInfo .Total length of area required
FrstByte DS ðCL240
DC X'F0',15X'00',X'F1',15X'00',X'F2',15X'00',X'F3'
DC 15X'00',X'F4',15X'00',X'F5',15X'00',X'F6',15X'00',X'F7'
DC 15X'00',X'F8',15X'00',X'F9',15X'00',X'C1',15X'00',X'C2'
DC 15X'00',X'C3',15X'00',X'C4',15X'00',X'C5',15X'00',X'C6'
SecByte DC X'F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6'
LTERG

* DSECTs follow

GetMArea DSECT
SaveArea DS 18F .General savearea
ParmStrt DS F .Start address of passed parms
Retcode DS F .Return code
FileRec DS CL335 .Workarea for file input/output
VTOCCB DS CL(VTOCLeng) .DCB area for volume VTOC
PARMDCB DS CL(ParmFill) .DCB area for file containing parms
RDJFCBPt DS F .Pointer used for RDJFCB
RDJFLst DS CL4
JFCBArea DS CL176 .Output area for RDJFCB
InArea DS CL256 .Returned data from control unit
ReadECB DS F
DS ØF
EXCPArea DS CL(EXCPIInfo)
ORG EXCPArea
IOB DS CL4 .Start of the IOB
ECB@ DS F .Address of ECB to post
CSW DS CL8 .Channel status word
.A(CCW,VTOC,0,0)

. Read config data' CCW

. Double word workarea

. UCB common segment address

. General workarea for WTOs

. Workarea for OPEN macro

. Workarea for CLOSE macro

. Flag

. Flag

. Workarea for UCBLOOK macro

. Pointer to request blocks

. Pointer to SVC99 text unit 0

. Pointer to SVC99 text unit 1

. Pointer to SVC99 text unit 2

. Pointer to SVC99 text unit 3

. SVC99 text unit 0 - DDNAME

. SVC99 text unit 1 - DNAME

. SVC99 text unit 2 - DS STATUS

. SVC99 text unit 3 - UNIT=SYSALLDA

. Address of file input/output area

. Parms we get from input parm file

. Has U(nit)xxxx or V(olser)yyyyyy

. Returned unit

. Returned volser

. Data obtained from disk controller

. Error message sent back to caller

. Subsystem id in printable format

. Unit add (offset 235) in prt fmt

. Phys. dev address in prt fmt

. DSECT for command exit fields

IEZVX101
IEFZB4DO
IEFZB4D2
IKJDAP08
IEFJFCBN
IEFUCB0B
DCBD DSORG=PS,DEVD=DA
IEZDEB
IECSDSL1 (1,4)

A A Keyser
 Systems Programmer
 Houghton Consulting Services Pty Ltd (Australia)
Listing APF-authorized libraries

The following program is a TSO command to list APF-authorized libraries. With dynamic APF, what we had at IPL time is not necessarily what we have ‘now’, and not everybody has access to a console. If the command is executed without a parameter list, it will list all APF-authorized libraries. If the command is executed with a parameter list, it will only list APF-authorized libraries that match the PARM (ie if one enters the command ‘AUTH SYS’ all APF-authorized starting with the prefix ‘SYS’ will be listed).

The program has been used successfully under MVS/ESA 5.1, MVS/ESA 5.2, and OS/390.

```
AUTH TITLE 'LIST AUTHORIZED LIBRARIES'
EQU *
STM R14,R12,12(R13) SAVE REGISTERS
LR R12,R15 GET NEW BASE
USING AUTH,R12 ESTABLISH IT
ST R13,SAVE+4 USUAL
LA R14,SAVE BORING
ST R14,8(R13) STUFF
LR R13,R14 FOR SYSTEM
LR R3,R1 LOAD CPPL PTR
USING CPPL,R3

*---- SET SYSTEM ID
L R9,16 CVTPTR
L R4,196(R9) CVTSMCA
MVC TITLSID(4),SMCASID-SMCABASE(R4)
L R14,CPPLCBUF GET COMMAND BUFFER
LH R15,2(R14) GET OPERAND OFFSET
LA R15,4(R15,R14) POINT TO OPERAND

GETPARMS DS ØH
CLI Ø(R15),C' ' ANY OPERANDS
BE Nonspec YES
LR R0,R15 REMEMBER
AH R14,Ø(R14) POINT TO END OF BUFFER
LA R1,REQDSN OUTPUT AREA

GETDSN DS ØH
CR R15,R14 END OF OF 1ST OP
BNL GETDSNXX OUT OF PARMS
MVC Ø(1,R1),Ø(R15) MOVE 1 BYTE OF DSN
LA R1,1(R1)
LA R15,1(R15)
CLI Ø(R15),C' ' END OF DSN
BE GETDSNXX YES-DO CALCS
```
CL1 Ø(R1),C' DONT OVERRUN OUTPUT
BE GETDSN SPACE AVAIL-CONTINUE
B INVPARM DSNNAME OVER 44 CHAR

GETDSNNX DS @H
LR R1,R15 L'DSN
SR R1,R0
LR R4,R1 L'JOBNAME
BCTR R1,R0 SUBTRACT FOR COMPARE
STH R1,REQDSNL STORE LENGTH
MVI SW,X'FF' SET SWITCH
OC REQDSN,REQDS1 SET TO UPPER CASE
MVC COMP+1(1),REQDSNL+1 SET LENGTH FOR COMPARE

NONSPEC EQU *
LA R13,SAVE1 STORE SAVE AREA FOR CSVAPF
LA R4,AREA GET AREA
MVC AREALEN,-AL4(28000) GET LENGTH
LA R5,AREALEN
CSVAPF REQUEST=LIST,ANSAREA=(4),ANSLEN=(5),
* RSCODE=REASON,RETCODE=RETCODE

TPUT CLEAR,L'CLEAR,FULLSCR CLEAR SCREEN
TPUT TITLE,TITLEL ADR PRINT TITLE AND TIME
TPUT BLANKS,L'BLANKS PRINT BLANK
TPUT HDR,HDRL PRINT
TPUT BLANKS,L'BLANKS PRINT BLANK
XC HDR(72),HDR CLEAR
L R6,Ø(R4) GET NUMBER OF ENTRIES
L R7,12(R4) GET OFFSET
LA R4,Ø(R7,R4) BUMP TO FIRST ENTRY
CLI SW,X'ØØ' HAVE WE PASSED A PARM
BE LOOP NO LOOP

COMP CLC REQDSN,10(R4) IS IT DSN WE WANT
BNE LOOP1 NO GET NEXT

LOOP MVC VOLSER(6),4(R4) GET VOLSER
MVC HDR(44),10(R4) GET DSNNAME
TPUT HDR,HDRL PRINT

LOOP1 XC HDR(72),HDR CLEAR
LH R7,Ø(R4) GET LENGTH
LA R4,Ø(R7,R4) BUMP TO NEXT
CLI SW,X'ØØ' HAVE WE PASSED PARM
BE LOOP2 NO LOOP
BCT R6,COMP YES GO TO COMPARE
B END GO HOME

LOOP2 BCT R6,LOOP

END EQU *
L R13,SAVE+4 RESTORE
LM R14,R12,12(R13) AND
XR R15,R15 GO
BR R14 HOME

INVPARM EQU *
SAVE DS 18F
SAVE1 DS 18F
Making global changes to PDS members

INTRODUCTION

The following utility will help to make changes to some or all members of a PDS file. To do this manually, would require editing the members one by one and issuing the change commands, or writing the commands to an edit macro, editing the files, and executing the macro. That is what this utility does automatically.

When TSO CHANGEG is invoked, you will be prompted with the screen seen below. A PDS name and also, optionally, a member can be passed to it as arguments. If this is done, the PDS and member fields in the panel will be filled. The PDS field is mandatory, the member field is optional. If you leave this last one blank, the change will affect
all members. If it is filled, it will be considered as the starting character
to select the members that will be affected. In the example panel, only
members starting by ‘M15’ will be processed. You can input up to
three strings to change. The CHANGE ALL option is assumed. You
just need to specify the ‘From’ and ‘To’ arguments, in any manner
acceptable to the ISPF editor Change command. In the above example,
you are asking to change ‘3380’ to ‘3390’ and ‘IEWL’ to ‘HEWLH096’.
When the command is executed, you will be informed of the total
number of changes for each member (eventually zero).

Figure 1: Example CHANGEG screen
HOW CHANGEG WORKS

This utility consists of the following programs:

- A REXX EXEC, CHANGEG, that is invoked at a TSO prompt. It sets up the two temporary files it needs (one to communicate with the COBOL program and another to create the editor macro). At the beginning of it, there is a variable ‘load’ that you must set to the name of the loadlib where the CHANGEGC module is located.

- A COBOL program, CHANGEGC, called by the EXEC to handle the full screen display and validate your input. This program, in turn, calls the PDISP Assembler module (see MVS Update, Issue 124, page 55). When you link-edit the COBOL program, you must have previously compiled and link-edited the PDISP program, and have its load library accessed as STEPLIB.

When the input is completed press ENTER, the COBOL program writes the input to the temporary file and exits. The EXEC reads the file and creates the appropriate edit macro. Then it lists the members of the PDS onto a REXX stem. Those that match your member’s specification (if any), will be edited using the macro. In the end, both temporary files will be deleted, completing the process.

CHANGEG SOURCE CODE

```rexx
/* REXX MVS */
/* CHANGEG - Global change of strings in PDS members through */
/* the ISPF editor command 'CHANGE' with the ALL option. */
/* This EXEC calls the module CHANGEGC */
/*---------------------------------------*/
load - "sis.loadlib" /* loadlib where module changegc is */
jobda - userid()".CHANGE1" /* names of temporary files */
jobda1= userid()".TPMACRO"
jobda2= userid()".TPMACRO(MACRO1)"
arg ficheiro .
fic = strip(ficheiro,"'"")
fic = translate(fic,"'\"\"\"")
fic = translate(fic,"\"\")
zz = msg(off)
call libertar
call alocar_1
dropbuf
address tso "call "load"(changegc) '""fic'""```

© 1997. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
execio 4 diskr tpchange "(finis"
if rc ← ø then signal saida
pull lin0
parse pull lin1; lin1 = strip(lin1)
pull lin2; lin2 = strip(lin2)
pull lin3; lin3 = strip(lin3)
if lin0 = "" then signal saida
if lin1 = "" then signal saida
ficheiro = space(left(lin0,50),0)
fic = strip(ficheiro,"")
mem1 = space(substr(lin0,51,7),0)
mem1 = strip(mem1,"""
1mem1 = length(mem1)
call lista_membros
call alocar_2
call alocar_3
/*---------- create editor macro and execute it ----------*/
txt0 = "/* REXX ISPF EDITOR */"
txt1 = " cha1=0; cha2=0; cha3=0"
txt2 = "ISREDIT MACRO"
txt2a = "ISREDIT CHANGE 'left(lin1,49)' 'ALL'"
txt2b = "ISREDIT (CHA1) = CHANGE_COUNTS"
txt3a = "ISREDIT CHANGE 'left(lin2,49)' 'ALL'"
txt3b = "ISREDIT (CHA2) = CHANGE_COUNTS"
txt4a = "ISREDIT CHANGE 'left(lin3,49)' 'ALL'"
txt4b = "ISREDIT (CHA3) = CHANGE_COUNTS"
txt5 = "ISREDIT SAVE"
txt5 = "ISREDIT (MEM1) = MEMBER"
txt6 = "ISREDIT END"
txt7 = "say 'Total changes"
txt7 = txt7 "in member 'left(mem1,8)':' cha1+cha2+cha3"
dropbuf
queue txt0
queue txt1
queue txt2
queue txt2a
queue txt2b
if lin2 = "" then do; queue txt3a; queue txt3b; end
if lin3 = "" then do; queue txt4a; queue txt4b; end
queue txt5
queue txt6
queue txt7
queue ""
"execio * diskw tpmacro (finis"
if rc ← ø then signal saida
"ALTLIB ACT APPLICATION(EXEC) DATASET( '''JOBDA1'')"
do y = 1 to cl
   "ISPEXEC EDIT DATASET( '''fic'"("tabl.y")') MACRO(MACRO1)"
end
/*---------------------------- exit ----------------------------*/
saida:
"ALTLIB RESET"
call libertar
"delete '"jobdal"'"
dropbuf
exit

/*------------------------- subroutines ---------------------------------------*/
lista_membros:
xx = outtrap(lista.)
address TSO "LISTDS ("ficheiro") MEMBERS ST"
xx = outtrap(off)
if lista.0 = 0 then signal saida
c1 = 0
membros = 0
do k = 1 to lista.0
    zzz = space(lista.k,0)
    zzz = strip(zzz,"·")
    if zzz = "MEMBERS" then do
        membros=1
        iterate k
    end
    if membros = 1 then do
        if left(zzz,lmeml) = meml | meml="" then do
            cl = cl+1
            tabl.cl = zzz
        end
    end
end
return

/*-----------------------------------------------*/
libertar:
"free da(''jobda'')"
"free da(''jobdal'')"
"free da(''jobda2'')"
return

/*-----------------------------------------------*/
alocar_1:
"alloc da(''jobda'') dd(tpchange) new reuse blkszie(8000),
  lrecl(80) recfm(f,b) dsorg(ps) space(1 1) tracks delete"
if rc=0 then do
    say "Error "rc" no allocation" jobda
    signal saida;
end ; else a=1
return
alocar_2:
"delete '"jobdal"'"
"free da(''jobdal'')"
"alloc da(''jobdal'') new reuse blkszie(8000) lrecl(80),
  recfm(f,b) dsorg(po) dir(1) space(1 1) tracks catalog"
if rc=0 then do
CHANGEGC SOURCE CODE

IDENTIFICATION DIVISION.
PROGRAM-ID. CHANGEGC.

* This program is invoked by the REXX EXEC "CHANGEG".          *
* This program calls the PDISP Assembler program.               *

ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
   SELECT TEMP0 ASSIGN TO TPCHANGE.

DATA DIVISION.
FILE SECTION.
FD TEMP0.
01 TEMP0-FD PIC X(80).

WORKING-STORAGE SECTION.
77 ALARME-ON PIC X VALUE "5".
77 ALARME-OFF PIC X VALUE SPACE.
77 PF3 PIC X VALUE "3".
77 PF15 PIC X VALUE "C".
01 CHANGEX0.
   03 TOTLENG PIC S9(8) COMP VALUE +959.
   03 CURRRT PIC S9(4) COMP VALUE +0.
   03 CURROUT PIC S9(4) COMP VALUE +651.
   03 AIDKEY PIC X VALUE SPACE.
   03 UPPER PIC X VALUE SPACE.
   03 CHARFIL PIC X VALUE SPACE.
   03 NUMFILL PIC X VALUE SPACE.
   03 FILLER PIC X(08) VALUE SPACE.
   03 ALARME PIC X VALUE SPACE.
   03 FILLER PIC X(04) VALUE X'11404013'.
   03 FILLER PIC X(05) VALUE X'1140401DF0'.
   03 FILLER PIC X(03) VALUE X'1140C9'.
   03 FILLER PIC X(02) VALUE X'1DF8'.
   03 FILLER PIC X(59) VALUE "CCCCC H H AAAAAA
- " NN N GGGGGG EEEEE GGGGGG".
   03 FILLER PIC X(03) VALUE X'1141D9'.

GLOBAL CHANGE FOR PDS MEMBERS VIA ISPF EDITOR 'CHANGE'.

Members:

Change From:

© 1997. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
03 FILLER PIC X(03) VALUE "To:".
03 FILLER PIC X(03) VALUE X'114DFD'.
03 TO1A PIC X(02) VALUE X'1DC0'.
03 TO1I PIC X(24) VALUE SPACES.
03 FILLER PIC X(03) VALUE X'114ED6'.
03 FILLER PIC X(02) VALUE X'1DF0'.
03 FILLER PIC X(05) VALUE "From:".
03 FILLER PIC X(03) VALUE X'1150D0'.
03 CH2A PIC X(02) VALUE X'1DC0'.
03 CH2I PIC X(24) VALUE SPACES.
03 FILLER PIC X(03) VALUE X'1150F6'.
03 FILLER PIC X(02) VALUE X'1DF0'.
03 FILLER PIC X(01) VALUE SPACES.
03 FILLER PIC X(03) VALUE X'1151E9'.
03 FILLER PIC X(02) VALUE X'1DF8'.
03 FILLER PIC X(03) VALUE "To:".
03 FILLER PIC X(03) VALUE X'1151ED'.
03 TO2A PIC X(02) VALUE X'1DC0'.
03 TO2I PIC X(24) VALUE SPACES.
03 FILLER PIC X(03) VALUE X'1152C6'.
03 FILLER PIC X(02) VALUE X'1DF0'.
03 FILLER PIC X(01) VALUE SPACES.
03 FILLER PIC X(03) VALUE X'1154C7'.
03 FILLER PIC X(02) VALUE X'1DF8'.
03 FILLER PIC X(05) VALUE "From:".
03 FILLER PIC X(03) VALUE X'1154CD'.
03 CH3A PIC X(02) VALUE X'1DC0'.
03 CH3I PIC X(24) VALUE SPACES.
03 FILLER PIC X(03) VALUE X'1154E6'.
03 FILLER PIC X(02) VALUE X'1DF0'.
03 FILLER PIC X(01) VALUE SPACES.
03 FILLER PIC X(03) VALUE X'1155D9'.
03 FILLER PIC X(02) VALUE X'1DF8'.
03 FILLER PIC X(03) VALUE "To:".
03 FILLER PIC X(03) VALUE X'1155DD'.
03 TO3A PIC X(02) VALUE X'1DC0'.
03 TO3I PIC X(24) VALUE SPACES.
03 FILLER PIC X(03) VALUE X'1155FE'.
03 FILLER PIC X(02) VALUE X'1DF8'.
03 MSGA PIC X(02) VALUE X'1DF0'.
03 MSGI PIC X(50) VALUE SPACES.
03 FILLER PIC X(03) VALUE X'1156DD'.
03 FILLER PIC X(46) VALUE "Enter: Execute
-PF3/15:Cancel".

01 MSG-1.
  02 FILLER PIC X(28) VALUE "Please enter the cursor area".

01 LINHA-0.
  02 PDS PIC X(50).
  02 MEM PIC X(7).

01 LINHA-1.
  02 CH1 PIC X(24).
  02 FILLER PIC X.
  02 TO1 PIC X(24).

01 LINHA-2.
  02 CH2 PIC X(24).
  02 FILLER PIC X.
  02 TO2 PIC X(24).

01 LINHA-3.
  02 CH3 PIC X(24).
  02 FILLER PIC X.
  02 TO3 PIC X(24).

LINKAGE SECTION.

01 ARGUMENTO.
  02 ARGLEN PIC S9(4) COMP.
  02 ARGVALUE PIC X(100).

PROCEDURE DIVISION USING ARGUMENTO.
  IF ARGLEN > 0
    UNSTRING ARGVALUE DELIMITED BY SPACE
    INTO PDSI MEMI.

INICIO.
  CALL "PDISP" USING CHANGEX0.
  MOVE SPACES TO LINHA-0 LINHA-1 LINHA-2 LINHA-3
  IF AIDKEY = PF3 OR AIDKEY = PF15
    GO TO ESCRITA.
  MOVE ALARME-OFF TO ALARME
  MOVE PDSI TO PDS
  MOVE MEMI TO MEM
  MOVE CH1I TO CH1
  MOVE TO1I TO TO1
  MOVE CH2I TO CH2
  MOVE TO2I TO TO2
  MOVE CH3I TO CH3
  MOVE TO3I TO TO3
  IF PDSI = SPACES
    MOVE 651 TO CUROUT
    MOVE MSG-1 TO MSGI
    MOVE ALARME-ON TO ALARME
    GO TO INICIO.
  IF CH1 = SPACES
    MOVE 814 TO CUROUT
    MOVE MSG-1 TO MSGI
    MOVE ALARME-ON TO ALARME
GO TO INICIO.
IF T01 = SPACES
  MOVE 894 TO CUROUT
  MOVE MSG-1 TO MSGI
  MOVE ALARME-ON TO ALARME
  GO TO INICIO.
IF CH2 = SPACES AND T02 NOT = SPACES
  MOVE 1054 TO CUROUT
  MOVE MSG-1 TO MSGI
  MOVE ALARME-ON TO ALARME
  GO TO INICIO.
IF CH2 NOT = SPACES AND T02 = SPACES
  MOVE 1134 TO CUROUT
  MOVE MSG-1 TO MSGI
  MOVE ALARME-ON TO ALARME
  GO TO INICIO.
IF CH3 = SPACES AND T03 NOT = SPACES
  MOVE 1294 TO CUROUT
  MOVE MSG-1 TO MSGI
  MOVE ALARME-ON TO ALARME
  GO TO INICIO.
IF CH3 NOT = SPACES AND T03 = SPACES
  MOVE 1374 TO CUROUT
  MOVE MSG-1 TO MSGI
  MOVE ALARME-ON TO ALARME
  GO TO INICIO.

ESCRITA.
OPEN OUTPUT TEMP0
WRITE TEMP0-FD FROM LINHA-0
WRITE TEMP0-FD FROM LINHA-1
WRITE TEMP0-FD FROM LINHA-2
WRITE TEMP0-FD FROM LINHA-3
CLOSE TEMP0.

SAIDA.
STOP RUN.
A volume mount analyser

INTRODUCTION

The volume mount analyser feature is provided in the form of two utilities within DFSMS Version 1.1. It is detailed in the manual *Volume Mount Analyser – SC26-4925*. The utilities can be used to determine the value of automating management of tape data. The reports produced by the utilities show how tapes are utilized and how many mounts occur. Previously, these services were provided via IBM software engineering services division. The reports can be used to:

- Study tape mount activity
- Monitor specific tape media usage
- Implement tape mount management processes.

The reports produced can be tailored for data classes, management classes, and auto-class selection. The whole purpose of performing a volume mount analysis is to maximize tape usage and reduce tape mounts, thereby removing an operator bottleneck.

The two programs used to implement the facility are GFTAXTR and GFTAVMA. The first program can be used to process SMF records and to correlate them into a format that can then be used as input to the second program. The second program analyses tape usage and tape mounts. The product also details how much space each tape dataset uses on volumes and how much space is being wasted. It details how many mounts can be saved and what DASD buffer space would be needed to hold datasets until transferred to tape. The concept of tape mount management is to reduce the number of mounts by:

- Allowing the system to manage the placement of data
- Utilizing hardware and software compaction
- Using new tape technology
- Filling tape volumes to capacity.
By using DFSMS/MVS to write full cartridges of data, optimum benefits can be achieved from compaction. Small datasets can be stacked on a single cartridge without being dependent on manual file markers and JCL changes. By using SMS to direct tape datasets to DASD no JCL changes are required. Mount reductions of 60 to 70% can be achieved, allowing integrated cartridge loaders, IDRC, and 3490Es to be fully exploited. This improves throughput and removes bottlenecks. DASD being used as a temporary buffer also allows faster access to data. The product requires DFSORT and MVS/ESA to run.

The recommended approach to perform an analysis is:

- Select a specific time period over which to monitor data
- Collect the relevant SMF records
- Run the GTFAXTR program to extract the data
- Run GTFAVMA to produce summary and detailed reports
- Analyse the reports.

Multi-system-type data needs to be merged or analysed separately.

The GTFAXTR program processes the SMF into a format that can be used for repeated runs of GTFAVMA. The program reads SMF type 14, 15, 21, and 30 (subtypes 4 and 5) records. It can also optionally read types 4, 5, 34, and 35 records. The types 4, 5, 21, 30 (subtypes 4 and 5), 34, and 35, records are discarded if they do not match the corresponding type 14 and 15 records.

The different SMF record types and their meanings are:

- Type 4 Step end
- Type 5 Job end
- Type 14 Open for input
- Type 15 Open for output
- Type 21 Volume demount
- Type 30 ASID record
- Type 34 Step end (TSO)
- Type 35 Job end (TSO).
To run the GTFaxTR job there is an example of the JCL included in SYS1.SAMPLIB in the member GTFaxTRP. I have included an example of the required JCL below. There are various parameters that can be coded as SYSIN data to control how data is extracted and to control output formatting. It is always best to check the percentage of records that have been dropped, and the percentage of type 21 records that occur. You should also review Chapter Three of the Volume Mount Analyser manual which details tape concepts. A return code of four from the program is acceptable.

GTFaxMA can be used to generate summary reports and optional detailed reports. The program has three processing phases:

1. GFTASRT1 Usage summary report
2. GFTASRT2 Volume summary report
3. GFTASRT3 Optional reports.

All input can be filtered if necessary using the filters that are detailed in Figure 1. An example of the GTFaxMAJCL which can be obtained from a supplied example in member GTFaxMAP in SYS1.SAMPLIB can be seen below.

<table>
<thead>
<tr>
<th>Limiting filters</th>
<th>Include/exclude filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>• MAXSIZE</td>
<td>Primary filters</td>
</tr>
<tr>
<td>• MINSIZE</td>
<td>• ACCOUNT</td>
</tr>
<tr>
<td>• MOUNT</td>
<td>• DATASET</td>
</tr>
<tr>
<td>• FILE</td>
<td>• EXPDT</td>
</tr>
<tr>
<td>• USAGE</td>
<td>• JOBNAME</td>
</tr>
<tr>
<td>• TIME</td>
<td>• PROGRAM</td>
</tr>
<tr>
<td></td>
<td>Secondary filters</td>
</tr>
<tr>
<td></td>
<td>• SYSTEMID</td>
</tr>
<tr>
<td></td>
<td>• UNIT</td>
</tr>
<tr>
<td></td>
<td>• UNITADDR</td>
</tr>
</tbody>
</table>

*Figure 1: Details of GTFaxMA filters*
The various reports that are produced give a number of different details. The general reports are used as follows:

- **GFTASRT** – input analysis report, which shows the times and dates of the SMF input data being sampled.
- **GFTASRT2** – volume analysis phase shows how many tapes have data on them.
- **GFTASRT3** – dataset analysis details.

The detailed reports that can be produced are controlled using different reporting parameters. The reports are:

- Dataset report: `REPORT(DATASET)`
- Maximum gigabytes: `REPORT(GBMAX)`
- Top report: `REPORT(TOP)`
- Usage report: `REPORT(USAGE)`
- Volume report: `REPORT(VOLUME)`.

The product also provides three simulation reports showing how tape usage can be changed.

**GTFAXTR JCL EXAMPLE**

```bash
//STS01A JOB (SDTS), 'JOHN BRADLEY', CLASS=A, MSGCLASS=Q,
// MSGLEVEL=(1,1)
//*
//DELETE EXEC PGM=IEFBR14
//OLDXTRCT DD DISP=(MOD,DELETE),
// DSN=STS01.BD.DATA.FTAXTR.D080296,
// UNIT=EPBD,
// SPACE=(TRK,(0))
//*
//XTRACT EXEC PGM=GTFAXTR,REGION=0M
//SMFIN DD DISP=SHR, DSN=SPROD.BD.SMFCUMA
//SYSOUT DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYSUOMP DD SYSOUT=*
//XTRCIN DD UNIT=EPBD,
// DSN=STS01.BD.DATA.GTFAXTR.D080296,
// DISP=(NEW,CATLG),
// RECFM=VB,
// SPACE=(1000,(3000,5000),RLSE),
// AVGREC=U
//XTRCOUT DD UNIT=EPBD.
```

// DSN-&XTRCOCU,
// RECFM=VB,
// SPACE=(1000,(30000,5000),RLSE),
// AVGREC=U
// XTRCWK01 DD UNIT=EPBD,
// SPACE=(300,(30000,5000),CONTIG,ROUND),
// AVGREC=U
// XTRCWK02 DD UNIT=EPBD,
// SPACE=(300,(30000,5000),CONTIG,ROUND),
// AVGREC=U
// XTRCWK03 DD UNIT=EPBD,
// SPACE=(300,(30000,5000),CONTIG,ROUND),
// AVGREC=U
// XTRCNTL DD DUMMY

**GTFAVMA JCL EXAMPLE**

//STS01A JOB (SDTS),'JOHN BRADLEY',CLASS=A,MSGCLASS=Q,
// MSGLEVEL=(1,1)
//*
//XTRACT EXEC PGM=GTFAVMA,REGION=8M,PARM='FILT#(1000)'
//XTRCIN DD UNIT=WORK,
// SPACE=(1000,(30000,1000)),
// AVGREC=U
//XTRCOUT DD UNIT=WORK,
// SPACE=(1000,(30000,1000)),
// AVGREC=U
//XTRCWK01 DD UNIT=EPBD,
// SPACE=(300,(30000,5000),CONTIG,ROUND),
// AVGREC=U
//XTRCWK02 DD UNIT=EPBD,
// SPACE=(300,(30000,5000),CONTIG,ROUND),
// AVGREC=U
//XTRCWK03 DD UNIT=EPBD,
// SPACE=(300,(30000,5000),CONTIG,ROUND),
// AVGREC=U
//XTRIN DD DISP=SHR,DSN=STS01.BD.DATA.GFTAXTR.D080296
//VMACHART DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYSOUT DD SYSOUT=* 
//VMAFLTRS DD UNIT=WORK,
// SPACE=(300,(30000,5000),CONTIG,ROUND),
// AVGREC=U
//VMAEXCL DD DUMMY,DCB=(DSORG=PS,RECFM=VB)
//VMAINCL DD DUMMY,DCB=(DSORG=PS,RECFM=VB)
//VMACNTL DD *
// REPORT(DATASET,GBMAX,TOP,USAGE,VOLUME

John Bradley  
Systems Programmer (UK)  © Xephon 1997
Generating structured Assembler programs with ISPF edit macros – part 1

INTRODUCTION

Most programmers have copied an existing program with similar functions and modified it to specific needs. This technique eliminates much of the effort in creating a program from scratch. This basic principle can be achieved with an ISPF edit macro that first copies a skeleton program and then uses additional ISPF edit macros to automatically add common functions and data structure definitions (objects, if you like). The basic techniques may be customized for other languages.

There are two basic skeletons: one for batch (ABATSKE)L and one for command level CICS (ACMDSKEL). To begin, you open an EDIT session with the member name you choose and issue either the ABAT or ACMD macro, for batch or CICS programs respectively. This initial macro copies the specific skeleton into the member and replaces the dummy program name with the current member name. The skeleton is obtained from the current library and, hence, can be customized by each TSO user.

The skeleton must contain specific comment or other statements which the macros use for placing generated code.

GENERAL MACROS

These macros are used for both skeletons. They are:

- Macro ADC is used to define constants. It requires two parameters. The first parameter is an Assembler label. The second parameter is the operand(s) for an Assembler DC statement. For example:

  \[
  \text{adc label1 c125'demonstration',c'text'}
  \]

  generates:

  \[
  \text{LABEL1 DC C125'DEMONSTRATION',C'TEXT'}
  \]
The above statement is placed above the first LTORG statement and the screen position remains unchanged. This allows constants to be defined as the logic is being developed, without the need to interrupt the programming process.

- Macro ADS is used to define data storage. It requires two parameters. The first parameter is an Assembler label. The second parameter is the operand(s) for an Assembler DS statement. For example:
  
  ads work1 f

  generates:

  WORK1  DS  F

  The above statement is placed above the first ‘* BEGIN STUB LINK’ comment statement and the screen position remains unchanged.

- Macro AEQU is used to define equate statements. It requires two parameters. The first parameter is an Assembler label. The second parameter is an Assembler expression. For example:

  aequ aminusb a-b

  generates:

  AMINUSB  EQU  A-B

  The above statement is placed above the first ‘* BEGIN STUB’ comment statement and the screen position remains unchanged.

- The ASTUB macro generates a branch and link instruction at the current line position, creates a stub for inserting functional code, and defines a space for storing the linkage address. It requires one parameter to define the stub and normally requires a second parameter to uniquely define the register storage work area. If the second parameter is not entered nulls are assumed. For example:

  astub subrut sr

  generates:

  BAL  RBAL, SUBROUT
which is placed at the current line and the following code (preceded by a comment box)

```
SUBROUT ST RBAL, BALSRSAV
  *
  L RBAL, BALSRSAV
  BR RBAL
```

is inserted above the ‘* END STUB DEFINE’ comment statement, and the definition:

```
SAVSRBAL DS F
```

is placed before the ‘* END STUB LINK’ comment statement. This simplifies the creation of structured programming ‘stubs’.

- The AJULGREG macro inserts a routine to convert a Julian date in the format retrieved from the MVS TIME macro and converts it to Gregorian formats. Operationally it functions in the manner specified by ASTUB.

- The AERR macro places the label of an error routine at the current cursor position, creates an error initialization routine, and branches to an error return procedure. It requires two parameters—an error message number and an error message. For example if the cursor is positioned as the ‘_’ in the following statement:

```
BNE _
aerr 16 invalid input
```

then the statement is completed as:

```
BNZ ERR16
```

and the following code:

```
ERR4 MVC ERRMSG(24),='ERROR 16 - INVALID INPUT'
LA RØ.16 SET ERROR CODE
LA R1.24 SET MESSAGE LENGTH
B ERROR GO SEND ERROR MESSAGE/EXIT
```

is inserted above the statement with label ‘ERROR ’.

MACROS FOR BATCH
The following macros are intended for generating batch programs.
The AINDCB macro generates an input DCB, statements to move the DCB to a work area allowing reenterable programming, OPEN and CLOSE lists and similar statements to move them to a work area, and OPENs and CLOSEs for the DCB. There are normally two parameters – the first is the DDNAME (and DCB name) and the second is to make the associated labels unique. For example:

```
aindcb infile if
```
generates:

```
MVC INFILE(INFILEL),INFILED INITIALIZE INFILE DCB
```
and is inserted before the ‘* END DCB INITIAL’ comment instruction.

```
MVC IFOPENL(IFOPENLN),OPEND SET INFILE OPEN LIST
OPEN (INFILE,(INPUT)),MF=(E,IFOPENL) OPEN INFILE
```
is inserted before the ‘* END DCB OPEN’ comment statement.

```
MVC IFCLOSL(IFCLOSLN),CLOSED SET INFILE CLOSE LIST
CLOSE (INFILE),MF=(E,IFCLOSL) CLOSE INFILE
```
is inserted before the ‘* END DCB CLOS’ comment statement.

```
INFILED DCB DDNAME=INFILE,DSORG=PS,MACRF=GM,EODAD=IFEOF
```
is inserted before the ‘* END DCB CONST’ comment statement.

```
INOPENL OPEN (,),MF=L
INOPENLN EQU */-INOPENL
INCLOSL CLOSE (,),MF=L
INCLOS LN EQU */-INCLOSL
```
is inserted before the ‘* END OPEN/CLOSE’ comment statement.

```
INFILE DCB DDNAME=INFILE,DSORG=PS,MACRF=GM,EODAD=IFEOF
INFILEL EQU */-INFILE
```
is inserted before the ‘* END DCB DSECT’ comment statement.

The AOUTDCB macro generates an output DCB, statements to move the DCB to a work area allowing re-enterable programming, OPEN and CLOSE lists and similar statements to move them to a work area, and OPENs and CLOSEs for the DCB. There are
normally two parameters – the first is the DDNAME (and DCB name) and the second is to make the associated labels unique. For example:

aoutdcb outile of

generates:

MVC OUTFILE(OUTFILEL),OUTFILED INITIALIZE OUTFILE DCB

which is inserted before the ‘* END DCB INITIAL’ comment instruction.

MVC OFOPENL(OFOPENLN),OPENEND SET OUTFILE OPEN LIST
OPEN (OUTFILE,(OUTPUT)),MF=(E,OFOPENL) OPEN OUTFILE

is inserted before the ‘* END DCB OPEN’ comment statement.

MVC OFCLOSIL(OFCLOSILN),CLOSED SET OUTFILE CLOSE LIST
CLOSE (OUTFILE),MF=(E,OFCLOSIL) CLOSE OUTFILE

is inserted before the ‘* END DCB CLOS’ comment statement.

OUTFILED DCB DDNAME-OUTFILE,DSORG-PS,MACRF-PM

is inserted before the ‘* END DCB CONST’ comment statement.

OFOPENL OPEN (,),MF=L
OFOPENLN EQU *-OFOPENL
OFCLOSIL CLOSE (,),MF=L
OFCLOSILN EQU *-OFCLOSIL

is inserted before the ‘* END OPEN/CLOSE’ comment statement.

OUTFILE DCB DDNAME-OUTPUT,DSORG-PS,MACRF-PM
OUTFILEL EQU *-OUTFILE

is inserted before the ‘* END DCB DSECT’ comment statement.

MACROS FOR COMMAND-LEVEL CICS

The following macros are intended for generating command-level CICS programs:

• The ACSA macro inserts a comment box and the following CSA definition:

COPY DFHCSADS
before the ‘* END DSECT’ comment statement and:

```assembly
EXEC CICS ADDRESS CSA(CSAREG)
CSAREG EQU 9
USING DFHCSADS
```

at the current cursor position.

- The ATCA macro inserts a comment box and the following TCA definition:

```assembly
DFHTCA CICSYST=CONFIG
```

after the ‘* END DSECT’ comment statement and

```assembly
L TCACBAR,CSACDTA-DFHCSADS(CSABAR)
USING DFHCSADS
```

at the current cursor position.

Note: the above assumes that the CSABAR register is pointing to the CSA. See macro ACSA if needed.

- The ATWA macro inserts a comment box and the following TWA definition:

```assembly
TWADS DSECT
TWA DS 0C
```

after the ‘* END DSECT’ comment statement, and:

```assembly
EXEC CICS ADDRESS TWA(YWAREG)
TWAREG EQU 8
USING TWADS,TWABAR
```

at the current cursor position.

**SAMPLE APPLICATION**

To create the trivial program (COPYFILE) that copies an input file (INPUT) to an output file (OUTPUT) the following steps are required:

1. Using ISPF edit, define member COPYFILE.
2. Issue the command (from the edit command line) to create the main program body: abat.
3. Define the input file by issuing the following from the Edit command line: aindcb input ip.
4 Define the output file by issuing the following edit command: aoutdcb output op.

5 Position source to begin program logic by issuing the following command: f first headpage.

   The batch version always assumes that a report is to be produced, if not you may remove this statement.

6 Create a stub to read a record from the input file by issuing: astub getrec gr.

7 Label the resulting BAL statement with MAINLOOP.

8 Create a stub to write a record to the output file by issuing: astub putrec pr.

9 Insert the following statements immediately after the generated BAL statement:

   B MAINLOOP
   IPEOFDS 0H

10 Create a work area by issuing: ads workarea cl100.

11 Go to the GETREC stub and insert the following statement:

   GET INPUT.WORKAREA READ INPUT RECORD

12 Go to the PUTREC stub and insert the following statement:

   PUT OUTPUT.WORKAREA WRITE OUTPUT RECORD

13 If you like, search for label ‘HEAD ’ and change the text to your preferred page heading. Note that there should be at least one blank character at the end of the text. The skeleton contains the logic to move this text to the storage defined by ‘HEADER ’, insert ‘PAGE’ page number id, etc for the ‘HEADPAGE’ routine.

14 Store the resulting program.

The above is intended to illustrate the general concepts and steps to generate Assembler language programs. Of course to generate a more complex program, more complex logic would be needed. Normally, this logic would be added with additional stubs, etc.

Note: IND$FILE has translated the NOT sign (hex '5F') to –. When translated back to EBCDIC it will self-correct.
**BATCH SKELETON**

```
LCLC &MYNAME

&MYNAME SETC '*************************************' CSECT NAME
RBASE EQU 12 BASE REGISTER FOR CSECT
RBAL EQU 10 BAL REGISTER
TITLE '&MYNAME' LISTING TITLE

***********************************************************************
*** LINKAGE CONVENTIONS ENTERING PROGRAM ***
***********************************************************************

&MYNAME CSECT .
STM R14,R12,12(R13) SAVE REGS TO CALLER S.A.
B (BEGIN-&MYNAME)(R15) BRANCH AROUND EYECATCHER
DC A(L'NAME) LENGTH OF CSECT NAME
NAME DC C'&MYNAME' CSECT NAME
DC C' &SYSDATE &SYSTIME ' ASSEMBLY DATE/TIME STAMP
BEGIN LR RBASE,R15 LOAD BASE REGISTER
USING &MYNAME,RBASE ADDRESSABILITY
PRINT NOGEN
GETMAIN R,LV=WORKDLEN GET SAVE/WORK AREA
ST R1,8(0,R13) MY S.A. ADDR INTO CALLER S.A.
ST R13,4(0,R1) CALLER S.A. ADDR INTO MY S.A.
LR R13,R1 R13 POINTS TO MY S.A.
USING WORKD,R13 ADDRESSABILITY OF SAVE AREA
L R1,4(0,R13) R1 POINTS TO CALLER S.A.
LM R15,R1,16(R1) R15 R0 AND R1 ARE RESTORED

***********************************************************************
*** MAINLINE ROUTINE ***
***********************************************************************

MAIN EQU *
BEGIN MAINLINE ROUTINE
ST R1,R1SAVE SAVE INITIAL R1
XC COMPCODE,COMPCODE CLEAR COMPLETION CODE

* BEGIN DCB INITIALIZATION
MVC PRINTER(PRINTERL),PRINTERD INITIALIZE OCB
* END DCB INITIALIZATION

* BEGIN DCB OPENS
MVC PROPENL(PROPENLN),OPENED INITIALIZE SET PRINTER OPEN LIST
OPEN (PRINTER,(OUTPUT)),MF=(E,PROPENL) OPEN PRINTER

* END DCB OPENS
MVC HEADER(L'HEADER),HEAD INITIALIZE HEADER
MVC HEADER+L'HEADER,L'HEADER+L'HEADER-1 CLEAR
MVC PAGENO-4(4),-C'PAGE' SET PAGE NUMBER ID
ZAP PAGES,-P'1' INITIALIZE PAGE COUNT
BAL RBAL,HEADPAGE PRINT PAGE HEADER

* BEGIN DCB CLOSE
MVC PRCLOS(L,PRCLOS),CLOSED INITIALIZE CLOSE LIST
CLOSE (PRINTER),MF=(E,PRCLOS) CLOSE IT

* END DCB CLOSE
END00 LA R15,0 SET COMPLETION CODE 00
ST R15,COMPCODE INTO STORAGE
```
*** LINKAGE CONVENTIONS EXITING PROGRAM ***

ENDING L R14,COMP CODE
LR R13,R14 R13 RESTORED
R14 SAVES ADDR OF MY S.A.
LR R13,4(R13) R13 SET TO COMP CODE
LR R15,R14 R14 RESTORED
FREEMAIN R,LV=(WORKDLEN,A=(R1)) FREE MY SAVE/WORK AREA
LR R15,R14 R15 SET TO COMP CODE
LM R0,R12,20(R13) R0-R12 RESTORED
L R14,12(R13) R14 RESTORED
MVI 12(R13),X'FF' SET COMPLETION SIGNAL
BR R14 RETURN TO CALLER

* BEGIN STUB DEFINE
*

* END STUB DEFINE

*** ERROR RETURNS ***

ERROR STH R0,COMP CODE
BAL RBAL,HEADPAGE EJECT PAGE
BAL RBAL,PRINT PRINT ERROR MESSAGE
B ENDING GO EXIT

*** PRINT ROUTINE ***

PRINT PUT PRINTER,LINE PRINT LINE
MVI LINE,C' ' SET SEED
MVC LINE+1(L'LINE),LINE CLEAR LINE
DOUBLESP BCTR R9,RBAL RETURN IF PAGE NOT FULL
HEADPAGE MVC PAGENO,-X'40202120' SET EDIT PATTERN
ED PAGENO,PAGES FORMAT PAGE NUMBER
AP PAGES,-P'1' INCREMENT PAGE COUNT
PUT PRINTER,HEADER PRINT PAGE HEADER
LA R9,56 SET LINES/PAGE
MVI LINE,C'0' SET TO DOUBLE SPACE AFTER HEADER
BR RBAL RETURN

*** FIXED DATA AREA ***

HEAD DC C'1&MYNAME'
OPEND OPEN (.),MF=L
CLOSED CLOSE (.),MF=L

* BEGIN DCB CONSTANTS

PRINTER DCB DDNAME=PRINTER,DEV=DA,DSORG=PS,LRECL=133,
BLKSIZE=133,MACRF=(PM),RECFM=FBA

* END DCB CONSTANTS

* END CONSTANTS

LTORG
*** DSECT FOR MY SAVE AREA AND VARIABLES. ***

WORKD  DSECT
MYSAVE DS 18F               MY REGISTER SAVE AREA
COMPCODE DS F               PROGRAM COMPLETION CODE
RETCD DS F                 INTERNAL RETURN CODE
RISAVE DS F                INITIAL VALUE IN R1
PAGES DS PL2
DOUBLE DS D

* BEGIN STUB LINK SAVE
*
*
* END STUB LINK SAVE
* BEGIN OPEN/CLOSE LIST
   DS 0D
PROPENL OPEN (.),MF=L
PROPENLN EQU *-PROPENL
PRCLOSLL CLOSE (.),MF=L
PRCLOSLLN EQU *-PRCLOSLL
* END OPEN/CLOSE LIST
* BEGIN DCB DSECTS
PRINTER DCB DDNAME=PRINTER,DEVD=DA,DSORG=PS,LRECL=133,
   BLKSIZE=133,MACRF=(PM),RECFM=FBA
PRINTERL EQU *-PRINTER
* END DCB DSECTS
* END DSECT INSERT
HEADER DS CL133
   ORG HEADER+L*HEADER/2-4
HEADDRATE DS CL8
   ORG HEADER+L*HEADER-5
PAGENO DS CL4
   ORG
LINE DS CL133
   DS 0D
WORKDLEN EQU *-WORKD

*******************************************************************************
*** REGISTER EQUATES ***
*******************************************************************************
R0  EQU 0
R1  EQU 1
   etc
R15 EQU 15
END
COMMAND-LEVEL CICS SKELETON

LCLC &MYNAME

&MYNAME SETC 'AAAAAAAA' CSECT NAME
RBASE EQU 11 BASE REGISTER FOR CSECT
RBAL EQU 10 BAL REGISTER
TITLE '&MYNAME' LISTING TITLE

***********************************************************************
*** LINKAGE CONVENTIONS ENTERING PROGRAM ***
***********************************************************************

&MYNAME DFHEIENT CODEREG=(RBASE),DATAREG=(R13),EIBREG=(R5)
B BEGIN BRANCH AROUND EYECATCHER
DC A(L'NAME) LENGTH OF CSECT NAME
NAME DC C'&MYNAME' CSECT NAME
DC C' &SYSDATE &SYSTIME ' ASSEMBLY DATE/TIME STAMP
BEGIN DS 0H
* BEGIN ADDRESS
*
*
* END ADDRESS

***********************************************************************
*** MAINLINE ROUTINE ***
***********************************************************************

MAIN EQU * BEGIN MAINLINE ROUTINE
ST R1,RISAVE SAVE INITIAL R1
XC COMPCODE,COMPCODE CLEAR COMPLETION CODE
* NORMAL EXIT
END00 LA R15,0 SET COMPLETION CODE 00
ST R15,COMPCODE INTO STORAGE
B ENDING GO TO ENDING

***********************************************************************
*** LINKAGE CONVENTIONS EXITING PROGRAM ***
***********************************************************************

ENDING L R15,COMPCODE R14 SAVES COMP CODE
EXEC CICS RETURN. RETURN TO CALLER
* BEGIN STUB DEFINE
*
*
* END STUB DEFINE

***********************************************************************
*** ERROR RETURNS ***
***********************************************************************

ERROR STH R1,TIOAL SET MESSAGE LENGTH
STH R0,COMPCODE SET COMPLETION CODE
EXEC CICS SEND FROM(ERRORMSG) LENGTH(TIOAL). SEND ERROR MESSG
EXEC CICS RETURN. EXIT TRANSACTION

***********************************************************************
*** FIXED DATA AREA ***
***********************************************************************

HEAD DC C'1&MYNAME'

LTORG

***********************************************************************
*** STORAGE FOR MY SAVE AREA AND VARIABLES. ***
***********************************************************************

DFHEISTG

WORKD DS 0C
MYSAVE DS 18F MY REGISTER SAVE AREA
COMPCODE DS F PROGRAM COMPLETION CODE
RETCDE DS F INTERNAL RETURN CODE
RISAVE DS F INITIAL VALUE IN R1
DOUBLE DS D

* BEGIN STUB LINK SAVE
*
* END STUB LINK SAVE
*

WORKDLEN EQU \*\-WORKD

***********************************************************************
*** CICS DSECTS ***
***********************************************************************

* BEGIN DSECTS
*
* END DSECTS

***********************************************************************
*** REGISTER EQUATES ***
***********************************************************************

R0 EQU 0
R1 EQU 1
etc

R15 EQU 15

END

ADC EDIT MACRO

PROC Ø
ISREDIT MACRO (LABEL OPERNDS) NOPROCESS
IF &SUBNAME = ? THEN DO
ISPEXEC DISPLAY PANEL(ADC)
EXIT
END
DO WHILE &LENGTH(&STR(&SPACES)) LT 65
SET &SPACES = &STR(&STR(&SPACES)&STR( ))
END
ISREDIT (RETX) = CURSOR
ISREDIT PROCESS DEST
IF &LASTCC = 0 THEN +
DO ISREDIT FIND FIRST "LTORG" 1
IF &LASTCC = 0 THEN +
DO
    SET ZEDMSG = &STR(COMMENT COMMAND PENDING)
    SET ZEDLMSG = &STR(ENTER AN 'A' OR 'B' LINE COMMAND +
        NO 'LTORG' CONSTANT)
    ISPEXEC SETMSG MSG(ISRZ001)
    EXIT CODE(12)
END
ELSE +
    DO
        ISREDIT (DEST) = CURSOR
        SET DEST = &EVAL(&DEST-2)
    END
    ELSE +
        ISREDIT (DEST) = LINENUM .ZDEST
        SET &NAME = &STR(&LABEL&SPACES)
        SET &NAME = &STR(&SUBSTR(1:9,&NAME)DC)
        SET &NAME = &STR(&NAME&OPERANDS&SPACES)
        ISREDIT LINE_AFTER &DEST = DATALINE "&NAME"
        ISREDIT LOCATE &RETX
        EXIT CODE(0)
    ADS EDIT MACRO
    PROC Ø
    ISREDIT MACRO (LABEL OPERNDS) NOPROCESS
    IF &SUBNAME = ? THEN DO
        ISPEXEC DISPLAY PANEL(ADS)
        EXIT
    END
    DO WHILE &LENGTH(&STR(&SPACES)) LT 65
        SET &SPACES = &STR(&STR(&SPACES)&STR( ))
    END
    ISREDIT (RETX) = CURSOR
    ISREDIT PROCESS DEST
    IF &LASTCC = Ø THEN +
        DO
            ISREDIT FIND FIRST "* BEGIN STUB LINK" 1
            IF &LASTCC = Ø THEN +
                DO
                    SET ZEDMSG = &STR(COMMENT COMMAND PENDING)
                    SET ZEDLMSG = &STR(ENTER AN 'A' OR 'B' LINE COMMAND +
                        NO '* BEGIN STUB LINK' CONSTANT)
                    ISPEXEC SETMSG MSG(ISRZ001)
                    EXIT CODE(12)
                END
            ELSE +
                DO
                    ISREDIT (DEST) = CURSOR
                    SET DEST = &EVAL(&DEST-2)
                END
        END
ELSE +
    ISREDIT (DEST) = LINENUM .ZDEST
SET &NAME = &STR(&LABEL&SPACES)
SET &NAME = &STR(&SUBSTR(1:9,&NAME)DS )
SET &NAME = &STR(&NAME&OPERNDS&SPACES)
ISREDIT LINE_AFTER &DEST = DATALINE "&NAME"
ISREDIT LOCATE &RETX
EXIT CODE(0)
AEQU EDIT MACRO
PROC Ø
ISREDIT MACRO (LABEL OPERNDS) NOPROCESS
IF &SUBNAME = ? THEN DO
ISPEXEC DISPLAY PANEL(AEQU)
EXIT
END
DO WHILE &LENGTH(&STR(&SPACES)) LT 65
SET &SPACES = &STR(&STR(&SPACES)&STR( ))
END
ISREDIT (RETX) = CURSOR
ISREDIT PROCESS DEST
IF &LASTCC = ø THEN +
    DO
    ISREDIT FIND FIRST "* BEGIN STUB" 1
    IF &LASTCC = ø THEN +
        DO
            SET ZEDMSMG = &STR(COMMENT COMMAND PENDING)
            SET ZEDLMSG = &STR(ENTER AN 'A' OR 'B' LINE COMMAND +
                NO '* BEGIN STUB' CONSTANT)
            ISPEXEC SETMSG MSG(ISRZØ01)
            EXIT CODE(12)
        END
    ELSE +
        DO
            ISREDIT (DEST) = CURSOR
            SET DEST = &EVAL(&DEST-2)
        END
    END
ELSE +
    ISREDIT (DEST) = LINENUM .ZDEST
SET &NAME = &STR(&LABEL&SPACES)
SET &NAME = &STR(&SUBSTR(1:9,&NAME)EQU )
SET &NAME = &STR(&NAME&OPERNDS&SPACES)
ISREDIT LINE_AFTER &DEST = DATALINE "&NAME"
ISREDIT LOCATE &RETX
EXIT CODE(0)

AERR EDIT MACRO
PROC Ø
ISREDIT MACRO (NO MESSAGE) NOPROCESS
IF &SUBNAME = ? THEN DO
ISPEXEC DISPLAY PANEL(AERR)
EXIT
END
DO WHILE &LENGTH(&STR(&SPACES)) LT 65
SET &SPACES = &STR(&SPACES)&STR( )
END
ISREDIT (R,C) = CURSOR
ISREDIT PROCESS DEST
IF &LASTCC = Ø THEN +
  DO
    ISREDIT FIND FIRST "ERROR " 1
    IF &LASTCC = Ø THEN +
      DO
        SET ZEDMSG = &STR(COMMENT COMMAND PENDING)
        SET ZEDLMSG = &STR(ENTER AN 'A' OR 'B' LINE COMMAND +
                                NO 'ERROR' CONSTANT)
        ISPEXEC SETMSG MSG(ISRZ001)
        EXIT CODE(12)
      END
    ELSE +
      DO
        SET ZEDMSG - &STR(COMMENT COMMAND PENDING)
        SET ZEDLMSG - &STR(ENTER AN 'A' OR 'B' LINE COMMAND +
                                NO 'ERROR' CONSTANT)
        ISPEXEC SETMSG MSG(ISRZ001)
        EXIT CODE(12)
      END
    END
  ELSE +
    DO
      ISREDIT (DEST) = CURSOR
      SET DEST = &EVAL(&DEST-2)
    END
    ISREDIT (DEST) - LINENUM .ZDEST
    SET &MSG = &STR(ERROR &NO - &MESSAGE)
    SET &L = &LENGTH(&STR(&MSG))
    SET &LAB = &STR(ERR&NO)
    SET &MOVE = &STR(&LAB&SPACES)
    SET &MOVE = &STR(&SUBSTR(1:9,&MOVE) MVC ERRORMSG(&L),-C'&MSG')
    SET &LA0 = &STR( LA R0,&NO&SPACES)
    SET &LA1 = &STR( LA R1,&L&SPACES)
    SET &B = &STR( B ERROR&SPACES)
    SET &LA0 Ø &STR(&SUBSTR(1:35,&LA0)SET ERROR CODE)
    SET &LA1 Ø &STR(&SUBSTR(1:35,&LA1)SET MESSAGE LENGTH)
    SET &B Ø &STR(&SUBSTR(1:35,&B)GO SEND ERROR MESSAGE)
    ISREDIT LINE_AFTER &DEST = DATALINE "&MOVE"
    ISREDIT LINE_AFTER &EVAL(&DEST+1) = DATALINE "&LA0"
    ISREDIT LINE_AFTER &EVAL(&DEST+2) = DATALINE "&LA1"
    ISREDIT LINE_AFTER &EVAL(&DEST+3) = DATALINE "&B"
    ISREDIT CURSOR = &R &C
    ISREDIT CHANGE P'-' &LAB
    EXIT CODE(Ø)

Keith H Nicaise
Technical Services Manager
Touro Infirmary (USA)
Enabling/disabling cache and NVS

Even with RAMAC and other sophisticated DASD, we still have a few 3390s lying around attached to a 3990 controller, in this case a 3990/03, which I think is one of the best pieces of equipment ever to come out of IBM laboratories. Recently we had to change batteries for the NVS feature and, once more, the IBM CE asked me to turn off the NVS subsystem for all the DASD on that particular controller. As it was, I was a bit tired of finding out the VOLIDs of all the DASD I had on those strings, and I am not very fond of the IEF244I message, you know, the one that tells you ‘at least 1 off-line unit(s) needed’.

In order to overcome this I developed a small REXX EXEC, which I called NVS, that enables or disables all the cache features for an entire controller. In order to use it you pass it three parameters:

1. ON or OFF
2. The first address (in hexadecimal) on which you want to act
3. The last address (in hexadecimal) on which you want to act.

For example:

NVS ON 0110 011F

The way it works is as follows: after the EXEC has validated the parameters that it has been passed, it will use the IDCAMS function DCOLLECT to generate a file for all on-line DASD, regardless of where they are attached. After this, it will read that file and discard all the addresses that are not within the specified range, and generate SETCACHE commands for each of the selected addresses.

After it has finished with the on-line file, and if there are no selected DASD to enable/disable, it will issue a message saying that there are no on-line volumes for the addresses specified (start and end), otherwise it will generate a file that will be used as SYSIN for IDCAMS, in order to issue the SETCACHE commands. The listing resulting from the IDCAMS execution will be displayed at your terminal, during the execution of IDCAMS. You can change the EXEC to send it to a file in order to browse it later, but I do not think it will be worth it (that is why I choose not to do it).
Before you use this EXEC you should check that you have the necessary authority clearance from RACF (or other security product in use at your site) because there are probably profiles in place for most of these functions (eg SETCACHE and DCOLLECT).

```rexx
/* REXX
                  ***********************
                  * N V S       *
                  *********************** */
parse upper arg opt start_addr end_addr .
if opt="ON" & opt="OFF" then
    say"You MUST specify the ON or OFF option for the NVS subsystem"
else
    do
        if start_addr="" then
            say"You MUST specify the START ADDRESS for the NVS subsystem"
        else
            do
                if end_addr="" then
                    say"You MUST specify the END ADDRESS for the NVS subsystem"
                else
                    call verify_addr
            end
        end
    end
exit
/* - - - - - - - - - */
verify_addr:
if "datatype(start_addr,",X") then
    say "You specified an invalid hexadecimal address (", 
         start_addr") for the NVS START Address"
else
    do
        if end_addr="" then
            say"You specified an invalid hexadecimal address (", 
                 end_addr") for the NVS END Address"
        else
            call set_cache
    end
end
return
/* - - - - - - - - - */
set_cache:
start_addr=x2c(start_addr)
end_addr=x2c(end_addr)
dsn=userlD()."D"date("J")'"T"time("S")
"alloc f(sysprint) dat('NULLFILE') shr reuse"
"alloc f(sysin) new lrec(80) reuse space(1) tracks da('dsn')"
"alloc f(outds) new lrec(264) recfm(v b) reuse space(11) tracks"
queue " DCOLLECT VOLUMES(*)  OUTFILE(OUTDS) NODATAINFO"
"execio "queued()" diskw sysin (finis)"
"call 'sys1.linklib(idcams)"
```

"execio * diskr outds (finis stem linha.)"
"free f(outds)"
"alloc f(sysprint) da(*) reuse"
k=0
do a=1 to linha.0
    parse value linha.a with 25 valid 31 . 69 dvctp 77 unit 79 .
    if unit>=start_addr & unit<=end_addr then
        do
            if opt="ON" then
                do
                    k=k+1
                    nvs.k=" SETCACHE UNIT("dvctp") VOL("valid") SUBSYSTEM ON"
                    k=k+1
                    nvs.k=" SETCACHE UNIT("dvctp") VOL("valid") NVS ON"
                    k=k+1
                    nvs.k=" SETCACHE UNIT("dvctp") VOL("valid") DEVICE ON"
                    k=k+1
                    nvs.k=" SETCACHE UNIT("dvctp") VOL("valid") DFW ON"
                end
            else
                do
                    k=k+1
                    nvs.k=" SETCACHE UNIT("dvctp") VOL("valid") DFW OFF"
                    k=k+1
                    nvs.k=" SETCACHE UNIT("dvctp") VOL("valid") DEVICE OFF"
                    k=k+1
                    nvs.k=" SETCACHE UNIT("dvctp") VOL("valid") NVS OFF"
                    k=k+1
                    nvs.k=" SETCACHE UNIT("dvctp") VOL("valid") SUBSYSTEM OFF"
                end
            end
        end
    end
    if k=0 then
        do
            say"There are no ONLINE volumes for the addresses (", ||c2x(start_addr)"","c2x(end_addr)"") that you specified"
        end
    else
        do
            "alloc f(sysin) da("dsn") old delete reuse"
            "execio "k" diskw sysin (finis stem nvs.)"
            "call 'sys1.linklib(idcams)"
        end
    end
"alloc f(sysin) da(*) reuse"
return
/* - - - - - - - - - */
Enhanced Assembler utilities – part 1

The following macros, INITR, EXITR, BSM24, BSM31, TESTXA, TESTESA, BALRXA, and CALLXA, first appeared in *MVS Update* in December 1987 and January 1988 *Assembler macros in XA 31-bit mode*. Additional features supporting an ESA environment will be presented easing the conversion of Assembler programs to execute in 31-bit code. The first part of this two part series will look at the INITR macro which has been considerably updated and now has increased functionality and additionally the TESTXA macro.

The macros developed are:

- **INITR** – for program initialization
- **EXITR** – for program return
- **BALRXA** – a substitute for Assembler BALR R14,R15
- **CALLXA** – a substitute for CALL
- **BSM24** – for setting addressing mode to 24-bit mode
- **BSM31** – for setting addressing mode to 31-bit mode
- **TESTXA** - for testing for 370 or ESA and 24- or 31-bit mode.

The remaining updated macros can be found in part 2 of this series. This will additionally include the macros AUTHON and AUTHOFF, which are used inside the macros below. They will dynamically turn on/off authorization through SVC authorization. In most cases the authorization can be set using the normal APF authorization methods, for example authorizing the library via IEAAPFnn/PROGnn, linking program as authorized and in case the program should be used under TSO, authorizing the program in IKJTSOnn.

In many cases it is easier to dynamically get authorization than to use the above, more correct methods, but there are even cases where I have found it very practical or even essential to switch back and forth using a dynamic method.
INITR MACRO

* INITR MACRO CREATES SUBPROGRAM LINKAGE IN FULL REENTRANT CODE:

* THIS MACRO REQUIRES RETURN TO CALLER VIA MACRO EXIRTR.
* WORKS IN MVS/370, MVS/XA OR MVS/ESA ENVIRONMENT.
* SAVES REGISTERS, GETMAINS NEW SAVE AREA (BELOW 16M TO ENSURE
* FREE FLOW BETWEEN 24 AND 31-BIT MODE PROGRAMS) IN SUBPOOL 0.
* THE SIZE OF THE GETMAINED DEFAULTS TO 72 FOR SAVE AREA BUT CAN
* VIA SIZE PARAMETER BE BIGGER IF THE PROGRAM SHOULD GETMAIN A
* WORK AREA FOR REENTRANCY.
* GETMAIN CAN TAKE PLACE IN OTHER AREAS BY LOC-ANY OR LOC-RES OR
* BY USING EXPLICIT SUBPOOL VIA THE PARAMETER SUBPOOL-.
* GETMAINED CAN BE CLEARED WITH ZEROS IF CLEAR-YES IS SPECIFIED.
* SETS A BASE REGISTER (DEFAULT TO R12), BASEREG POINTS TO PROGRAM
* ENTRY FOR EASE OF DEBUGGING.
* SECONDARY, TERTIARY, AND QUARTERNARY BASEREGISTERS CAN BE SELECTED
* USING PARAMETERS SECBASE-, TERBASE-, AND QARBASE-.
* CREATES REGISTER EQUATES (R0 TO R15)
* CREATES PROGRAM NAME AND COMPILATION TIMESTAMP EYECATCHER.
* CREATES AMODE, RMODE, AND SYSSPLV.
* GENERATES CODE THAT CHECKS THAT ASSEMBLED ADDRESSING MODE
* CORRESPONDS TO EXECUTION ADDRESSING MODE:
* IF ASSEMBLED IN AMODE 24 AND ENTERED IN 31 MODE, MODE IS CHANGED
* TO 24; IF ASSEMBLED IN AMODE 31 AND ENTERED IN 24 MODE (CAN ONLY
* HAPPEN WHEN RESIDING UNDER 16M) MODE IS CHANGED TO 31; IF ASSEMBLED
* IN AMODE ANY IT CAN BE ENTERED IN ANY ADDRESSING MODE.
* CODE FOR SUPPORT OF NON-XA (MVS/370) WILL ONLY BE GENERATED IF
* PARAMETER MVS370-SUP IS SPECIFIED OR &SLEVEL=1.
* CODE FOR SUPPORT OF XA/ESA WILL ONLY BE GENERATED IF &SLEVEL > 1.
* GENEREL SUPPORT CODE CONTAINING SETUP OF ESTAE AND STAX-ENVIRON-
* MENT, LOCATION OF PARAMETER DATA, HOW PROGRAM IS INVOKED IE TSO-
* COMMAND, EXECUTED FROM JCL OR TSO CALL, OR AS SUBROUTINE. THE
* CALLER CAN PICK UP THIS INFORMATION FROM THE DATA PLACED IN THE
* GETMAINED WORKAREA.
* ALIAS CAN BE USED TO INDICATE TO MACRO THAT THIS PROGRAM CAN ALSO
* BE CALLED THROUGH AN ALIAS NAME; THE INDICATED ALIAS NAME WILL BE
* USED INTERNALLY IN THE MACRO TO LOOK UP PARAMETERS AND PROGRAM
* WILL BE INFORMED IN OPTION INDICATION THAT IT WAS INVOKED UNDER
* ALIAS.
* REGISTER EQUATES ARE GENERATED UNLESS REGEUQ-NO IS SPECIFIED.
* THIS CODE WILL BE GENERATED IF PARAMETER GENCODE-YES IS SUPPLIED.
* IN THIS CASE SIMILAR CODE WILL BE GENERATED BY THE EXIRTR MACRO.
* IN THIS CASE R15 WILL CONTAIN THE LENGTH OF DATA IF ANY, AND R14
* THE ADDRESS OF DATA IF ANY.
* IF GENCODE-YES TRANSLATION OF PARAMETER DATA WILL BY DEFAULT BE
* TRANSLATED TO UPPERCASE; THIS CAN BE SUPPRESSED BY PARAMETER
* XLATE-NO.
* IF GENCODE-YES ESTAE EXIT WILL BE ACTIVATED BUT CAN BE SUPPRESSED
* BY PARAMETER ESTAE-NO.

© 1997. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
* IF GENCODE-YES STAX EXIT WILL BE ACTIVATED BUT CAN BE SUPPRESSED
* BY PARAMETER STAX-NO.
* IF GENCODE-YES APF AND STATE WILL BE SAVED, AND RESTORED AT EXIT;
* THIS FUNCTION CAN BE SUPPRESSED BY PARAMETER TESTAUTH-NO.
* IF GENCODE-YES HANDLING AND SCANNING OF OPTIONAL DATA-PARAMETER
* FIELD (1ST PARAMETER) CAN BE SUPPRESSED BY PARAMETER SCANDATA-NO
* IN WHICH CASE DATAADDR AND PARMLEN ARE INITIALIZED TO ZERO.
* IF GENCODE-YES RETURNCODE FROM ESTAT EXIT WILL DEFAULT TO 16 BUT
* CAN BE SET TO ANY VALUE.
* THE FOLLOWING DATA AREAS AND EOUS WILL BE EXTERNALLY AVAILABLE WHEN
* GENCODE-YES IS USED:
* UCBPFLEN     LENGTH OF UCB PREFIX
* WORKAREA DSECT GETMAINED WORKAREA
* RETCODE DS A RETURN CODE
* PARMADDR DS A ADDR OF PARMLIST
* DATAADDR DS A ADDR OF PARAMETER DATA
* PARMLEN DS H LENGTH OF PARAMETER DATA
* ADSPNAME DS CL8 NAME OF ADDRESS SPACE
* OPTIONS DS C EXECUTION OPTIONS
* ATTN EQU X'80' ATTN FLAG SET
* TSOCMD EQU X'40' INDICATE CALLED AS TSO COMMAND
* EXECCALL EQU X'20' INDICATE JCL-EXEC OR TSO-CALL
* SUBRUTIN EQU X'10' INDICATE CALLED AS SUBROUTINE
* STC EQU X'08' STARTED TASK ADDR SPACE
* TSO EQU X'04' TSO ADDR SPACE (FOREGROUND)
* APPC EQU X'02' APPC TYPE ADDR SPACE
* JOB EQU X'01' JOB TYPE ADDR SPACE
* OPTIONR DS C EXECUTION OPTIONS
* APFON EQU X'80' APF AUTHORIZED AT ENTRY
* SUPVSTAT EQU X'40' IN SUPERVISOR STATE AT ENTRY
* SYSKEY EQU X'20' IN SYSTEM KEY AT ENTRY
* ALIASINV EQU X'10' PROGRAM INVOKED USING ALIAS NAME
* IKJEFT01 EQU X'01' TSO IN FOREGROUND OR BATCH
* USERWORK EQU * ADD USER DEFINITIONS HERE
* DS CL256 ALLOCATE MINIMUM 256 FOR USER
* WORKLEN EQU ^WORKAREA LENGTH TO GETMAIN
* THE FOLLOWING SYSTEM DATA AREAS MACROS WILL BE INVOKED
* GENCODE-YES IS USED:
* CVT
* ASXB
* ASCB
* ASXB
* PSA
* RB
* TCB
* ACEE
* JSCB
* PSCB
* JCT
EXAMPLE OF INVOCATION:

GBLC &ID
GBLA &IDLEN
TEMPNAME INITR AMODE=31,RMODE=ANY,GENCODE=YES,SIZE=GETSIZE
WORKAREA DSECT
ORG USERWORK
OWNDATA1 DS CLB
OWNDATA2 DS CL16
GETSIZE EQU *-WORKAREA

&ID CSECT

* OWN CODE

RETURN EQU *
EXITR
LTORG
END

MACRO
&NAME INITR &BASEREG=12,
&SECBASE=0,
&TERBASE=0,
&QARBASE=0,
&SUBPOOL=0,
&SIZE=72,
&LOC=BELLO,
&CLEAR=YES,
&SPLEVEL=4,
&MVS370=NOTSUP,
&GENCODE=NO,
&XLA=ES,
&ESTAE=ES,
&STAX=ES,
&TESTAUTH=ES,
&ABENDRET=16,
&SCANDATA=ES,
&REGEQU=ES,
&AMODE=31,
&ALIAS=,
&RMODE=ANY
INITR PARAMETERS: BASEREG. DEFAULTS TO R12; NO SECONDARY AND TERTIARY
BASEREG IS DEFAULT.
SIZE DEFAULTS TO 72, SUBPOOL DEFAULTS TO 0 FOR SAVE/WORK GETMAIN.
SPELEVEL, DEFAULTS TO 4 FOR MVS/ESA MACRO EXPANSIONS; CAN BE SET
TO 1 FOR 370 MACRO EXPANSIONS PROVIDED 370 MACRO LIBRARY IS USED,
OR TO 2 FOR XA OR
3 FOR ESA V3 MACRO EXPANSIONS PROVIDED ESA V3 MACRO LIBRARY IS USED.
AMODE, DEFAULTS TO 31. RMODE, DEFAULTS TO ANY.
AMODE RMODE SPELEVEL COMBINATIONS VALID:
24 24 1, 2, 3 OR 4
ANY 24 1, 2, 3 OR 4
31 ANY 2, 3 OR 4
31 24 2, 3 OR 4
MHELP 2+4
MHHELP 4
LCLC &DT,&TM,&AM,&RM,&SP
LCLC &NONCHK,&START
LCLC &NONXA,&COMM,&ALEXIST,&ALIASNM,&SIZEC,&SIZE3L
LCL &ALIASLN
&NONCHK SETC 'IN1'. '&SYSNDX'
&START SETC 'IN2'. '&SYSNDX'
&NONXA SETC 'IN3'. '&SYSNDX'
&COMM SETC 'IN4', '&SYSNDX'
&YY SETC '&SYSDATE'(7,2)
&MM SETC '&SYSDATE'(1,2)
&DD SETC '&SYSDATE'(4,2)
&TM SETC '&SYSTIME'
&ALEXIST SETC 'NO'
GBLC &GETPOOL
GBLC &MSIZE
GBLC &MVS370S
GBLC &SYSSPLV
GBLC &GENCO FROM INITR
GBLC &XLATEF FROM INITR
GBLC &ESTALST FROM EXITR
GBLC &ESTAEND FROM EXITR
GBLC &STAXLST FROM EXITR
GBLC &STAXEND FROM EXITR
GBLC &RETRYR1 FROM EXITR
GBLC &RETRYR2 FROM EXITR
GBLC &SECBS FROM INITR
GBLC &TERBS FROM INITR
GBLC &QARBS FROM INITR
GBLC &TRLATE FROM EXITR
GBLC &TRTAB FROM EXITR
GBLC &ABRET FROM EXITR
GBLC &ESTAER FROM INITR
GBLC &STAXR FROM INITR
GBLC &TSTAUT FROM INITR

&PGMLONG SETC 'IMB'. '&SYSNDX'
&ALNAME SETC 'IMC'. '&SYSNDX'
&NOMORRB SETC 'IMD'. '&SYSNDX'

AIF ('&CLEAR' EO 'NO'). NOBS23
AIF ('&BASEREG' NE '2' AND '&BASEREG' NE 'R2'). BASER2
MNOTE 8, 'BASEREG CAN NOT BE R2 WHEN CLEAR=YES'

BASER2 ANOP
AIF ('&BASEREG' NE '3' AND '&BASEREG' NE 'R3'). BASER3
MNOTE 8, 'BASEREG CAN NOT BE R3 WHEN CLEAR=YES'

BASER3 ANOP
NOBS23 ANOP
AIF ('&TESTAUTH' EO 'NO'). NOBS23A
AIF ('&BASEREG' NE '2' AND '&BASEREG' NE 'R2'). BASER2A
MNOTE 8, 'BASEREG CAN NOT BE R2 WHEN TESTAUTH=YES'

BASER2A ANOP
AIF ('&BASEREG' NE '3' AND '&BASEREG' NE 'R3'). BASER3A
MNOTE 8, 'BASEREG CAN NOT BE R3 WHEN TESTAUTH=YES'

BASER3A ANOP
NOBS23A ANOP
AIF ('&CLEAR' EO 'NO'). NOSE23
AIF ('&SECBASE' NE '2' AND '&SECBASE' NE 'R2'). SECBR2
MNOTE 8, 'SECBASE CAN NOT BE R2 WHEN CLEAR=YES'

SECBR2 ANOP
AIF ('&SECBASE' NE '3' AND '&SECBASE' NE 'R3'). SECBR3
MNOTE 8, 'SECBASE CAN NOT BE R3 WHEN CLEAR=YES'

SECBR3 ANOP
NOSE23 ANOP
AIF ('&TESTAUTH' EO 'NO'). NOSE23A
AIF ('&SECBASE' NE '2' AND '&SECBASE' NE 'R2'). SECBR2A
MNOTE 8, 'SECBASE CAN NOT BE R2 WHEN TESTAUTH=YES'

SECBR2A ANOP
AIF ('&SECBASE' NE '3' AND '&SECBASE' NE 'R3'). SECBR3A
MNOTE 8, 'SECBASE CAN NOT BE R3 WHEN TESTAUTH=YES'

SECBR3A ANOP
NOSE23A ANOP
AIF ('&CLEAR' EO 'NO'). NOTE23
AIF ('&TERBASE' NE '2' AND '&TERBASE' NE 'R2'). TERBR2
MNOTE 8, 'TERBASE CAN NOT BE R2 WHEN CLEAR=YES'

TERBR2 ANOP
AIF ('&TERBASE' NE '3' AND '&TERBASE' NE 'R3'). TERBR3
MNOTE 8, 'TERBASE CAN NOT BE R3 WHEN CLEAR=YES'

TERBR3 ANOP
NOTE23 ANOP
AIF ('&TESTAUTH' EO 'NO'). NOTE23A
AIF ('&TERBASE' NE '2' AND '&TERBASE' NE 'R2'). TERBR2A
MNOTE 8, 'TERBASE CAN NOT BE R2 WHEN TESTAUTH=YES'

TERBR2A ANOP
AIF ('&TERBASE' NE '3' AND '&TERBASE' NE 'R3'). TERBR3A
MNOTE 8, 'TERBASE CAN NOT BE R3 WHEN TESTAUTH=YES'

TERBR3A ANOP
NOTE23A ANOP
AIF ('&CLEAR' EQ 'NO').NOOA23
AIF ('&OARBASE' NE '2' AND '&OARBASE' NE 'R2').QARBR2
MNOTE 8,'OARBASE CAN NOT BE R2 WHEN CLEAR=NO'

QARBR2 ANOP
AIF ('&OARBASE' NE '3' AND '&OARBASE' NE 'R3').QARBR3
MNOTE 8,'OARBASE CAN NOT BE R3 WHEN CLEAR=NO'

QARBR3 ANOP
NOOA23 ANOP
AIF ('&TESTAUTH' EQ 'NO').NOOA23
AIF ('&OARBASE' NE '2' AND '&OARBASE' NE 'R2').QARBR2A
MNOTE 8,'OARBASE CAN NOT BE R2 WHEN TESTAUTH=NO'

QARBR2A ANOP
AIF ('&OARBASE' NE '3' AND '&OARBASE' NE 'R3').QARBR3A
MNOTE 8,'OARBASE CAN NOT BE R3 WHEN TESTAUTH=NO'

QARBR3A ANOP
NOOA23A ANOP
AIF ('&BASEREG' EQ '0' AND '&BASEREG' NE 'R0').BASER0
MNOTE 8,'BASEREG CAN NOT BE R0'

BASER0 ANOP
AIF ('&BASEREG' NE '1' AND '&BASEREG' NE 'R1').BASER1
MNOTE 8,'BASEREG CAN NOT BE R1'

BASER1 ANOP
AIF ('&BASEREG' NE '14' AND '&BASEREG' NE 'R14').BASER14
MNOTE 8,'BASEREG CAN NOT BE R14'

BASER14 ANOP
AIF ('&BASEREG' NE '15' AND '&BASEREG' NE 'R15').BASER15
MNOTE 8,'BASEREG CAN NOT BE R15'

BASER15 ANOP
AIF ('&SECBASE' NE '1' AND '&SECBASE' NE 'R1').SECBR1
MNOTE 8,'SECBASE CAN NOT BE R1'

SECBR1 ANOP
AIF ('&SECBASE' NE '14' AND '&SECBASE' NE 'R14').SECBR14
MNOTE 8,'SECBASE CAN NOT BE R14'

SECBR14 ANOP
AIF ('&SECBASE' NE '15' AND '&SECBASE' NE 'R15').SECBR15
MNOTE 8,'SECBASE CAN NOT BE R15'

SECBR15 ANOP
AIF ('&TERBASE' NE '1' AND '&TERBASE' NE 'R1').TERBR1
MNOTE 8,'TERBASE CAN NOT BE R1'

TERBR1 ANOP
AIF ('&TERBASE' NE '14' AND '&TERBASE' NE 'R14').TERBR14
MNOTE 8,'TERBASE CAN NOT BE R14'

TERBR14 ANOP
AIF ('&TERBASE' NE '15' AND '&TERBASE' NE 'R15').TERBR15
MNOTE 8,'TERBASE CAN NOT BE R15'

TERBR15 ANOP
AIF ('&OARBASE' NE '1' AND '&OARBASE' NE 'R1').QARBR1
MNOTE 8,'OARBASE CAN NOT BE R1'

QARBR1 ANOP
AIF ('&QARBASE' NE '14' AND '&QARBASE' NE 'R14').QARBR14
MNOTE 8,'QARBASE CAN NOT BE R14'

QARBR14 ANOP
AIF ('&QARBASE' NE '15' AND '&QARBASE' NE 'R15').QARBR15
MNOTE 8,'QARBASE CAN NOT BE R15'

QARBR15 ANOP
AIF ('&QARBASE' EQ '0' OR '&QARBASE' EQ 'R0').QARBO
AIF ('&TERBASE' NE '0' AND '&TERBASE' NE 'R0').QARBO
MNOTE 8,'TERBASE MUST BE SET IF QARBASE IS SET'

QARBO ANOP
AIF ('&TERBASE' EQ '0' OR '&TERBASE' EQ 'R0').TERBO
AIF ('&SECBASE' NE '0' AND '&SECBASE' NE 'R0').TERBO
MNOTE 8,'SECBASE MUST BE SET IF TERBASE IS SET'

TERBO ANOP
AIF ('&BASEREG' NE '&SECBASE').PSBR
MNOTE 8,'PRIMARY BASE AND SECONDARY MAY NOT BE EQUAL'

PSBR ANOP
AIF ('&BASEREG' NE '&TERBASE').PTBR
MNOTE 8,'PRIMARY BASE AND TERTIARY MAY NOT BE EQUAL'

PTBR ANOP
AIF ('&BASEREG' NE '&QARBASE').PQBR
MNOTE 8,'PRIMARY BASE AND QARTERNARY MAY NOT BE EQUAL'

PQBR ANOP
AIF ('&TERBASE' EQ '0' OR '&TERBASE' EQ 'R0').STBR
AIF ('&SECBASE' NE '&TERBASE').STBR
MNOTE 8,'SECBASE AND TERBASE MUST BE UNEQUAL'

STBR ANOP
AIF ('&QARBASE' EQ '0' OR '&QARBASE' EQ 'R0').SQBR
AIF ('&SECBASE' NE '&QARBASE').SQBR
MNOTE 8,'SECBASE AND QARBASE MUST BE UNEQUAL'

SQBR ANOP
AIF ('&QARBASE' EQ '0' OR '&QARBASE' EQ 'R0').TQBR
AIF ('&TERBASE' NE '&QARBASE').TQBR
MNOTE 8,'TERBASE AND QARBASE MUST BE UNEQUAL'

TQBR ANOP
&SIZEC SETC '&SIZE'
&SIZEL SETC '&SIZE'(1,1)
AIF ('&SIZE' GE '72').SSIZE
AIF ('&SIZEL' LT '0').SSIZE
MNOTE 8,'SIZE OF SAVE/WORK MUST A LEAST BE 72 BYTES'

SSIZE ANOP
AIF ('&AMODE' NE '24').AM31
AIF ('&RMODE' EQ '24').AMMR24
MNOTE 8,'WHEN AMODE=24, RMODE MUST BE 24 ALSO'

AMRM24 ANOP
AGO .AMANYSP

AM31 ANOP
AIF ('&AMODE' NE '31').AMANY
AIF ('&RMODE' EQ '24').AM31RM
AIF ('&RMODE' EQ 'ANY').AM31RM
MNOTE 8, 'WHEN AMODE=31, RMODE MUST BE 24 OR ANY'

AM31RM ANOP
AIF ('&SLEVEL' GT '1').AMRM31
MNOTE 8, 'WHEN AMODE=31, SLEVEL MUST BE 2, 3 OR 4'

AMRM31 ANOP
AGO .AMANYSP

AMANY ANOP
AIF ('&AMODE' EQ 'ANY').AMANYRM
MNOTE 8, 'AMODE IS NOT 24, 31 OR ANY'

AMANYRM ANOP
AIF ('&RMODE' EQ '24').AMANY24
MNOTE 8, 'WHEN AMODE=ANY, RMODE MUST BE 24'

AMANY24 ANOP
AIF ('&SLEVEL' EQ '1').AMANYSP
AIF ('&SLEVEL' EQ '2').AMANYSP
AIF ('&SLEVEL' EQ '3').AMANYSP
AIF ('&SLEVEL' EQ '4').AMANYSP
MNOTE 8, 'WHEN AMODE=ANY, SLEVEL MUST BE 1, 2, 3 OR 4'

AMANYSP ANOP
AIF ('&CLEAR' EQ 'YES').CLEAR
AIF ('&CLEAR' EQ 'NO').CLEAR
MNOTE 8, 'CLEAR GETMAINDD MUST BE EITHER YES OR NO'

CLEAR ANOP
AIF ('&GENCODE' EQ 'YES').GENCODE
AIF ('&GENCODE' EQ 'NO').GENCODE
MNOTE 8, 'GENCODE MUST BE EITHER YES OR NO'

GENCODE ANOP
&GENCO SETC '&GENCODE'
AIF ('&GENCO' NE 'YES').NGENCO
GBLC &ID
GBLA &IDLEN

&ID SETC '&NAME'
&IDLEN SETA
AIF ('&IDLEN' EQ '0').IDLENER
AIF ('&IDLEN' GT '8').IDLENER
AGO .BYPGENC

IDLENER ANOP
MNOTE 8, 'NAME OF PROGRAM MUST BE PRESENT AND MAX 8 CHARS LONG'

BYPGENC ANOP
AIF ('&ALIAS' EQ '').ALIASZR
&ALIASLN SETA K&ALIAS
AIF ('&ALIASLN' EQ '0').ALIASZR
AIF ('&ALIASLN' GT '8').ALIASER
&ALEXIST SETC 'YES'
&ALIASNM SETC '&ALIAS'
AGO .ALIASNZ

ALIASER ANOP
MNOTE 8, 'ALIAS OF PROGRAM CAN BE MAX 8 CHARS LONG'

ALIASNZ ANOP
&ALIASNM SETC '&ID'
&ALIASLN SETA K'&ID
ALIASNZ ANOP
   AIF ('&XLATE' EQ 'YES').XLATE
   AIF ('&XLATE' EQ 'NO').XLATE
   MNOTE 8, 'XLATE MUST BE EITHER YES OR NO'

XLATE ANOP
   AIF ('&ESTAE' EQ 'YES').ESTAE
   AIF ('&ESTAE' EQ 'NO').ESTAE
   MNOTE 8, 'ESTAE MUST BE EITHER YES OR NO'

ESTAE ANOP
   AIF ('&STAX' EQ 'YES').STAX
   AIF ('&STAX' EQ 'NO').STAX
   MNOTE 8, 'STAX MUST BE EITHER YES OR NO'

STAX ANOP
   AIF ('&TSTAUT' EQ 'YES').TSTAUT
   AIF ('&TSTAUT' EQ 'NO').TSTAUT
   MNOTE 8, 'TESTAUTH MUST BE EITHER YES OR NO'

TSTAUT ANOP
   AIF ('&SCANDATA' EQ 'YES').SCANDT
   AIF ('&SCANDATA' EQ 'NO').SCANDT
   MNOTE 8, 'SCANDATA MUST BE EITHER YES OR NO'

SCANDT ANOP

NGENCO ANOP
   &AM  SETC '&AMODE'
   &RM  SETC '&RMODE'
   &SP  SETC '&SPLEVEL'

   &NAME AMODE &AM          ADDRESSING MODE
   &NAME RMODE &RM          RESIDENCY MODE
   SPLEVEL TEST SET SYSSPLV
   AIF ('&SYSSPLV' EQ '&SP').SPEO
   AIF ('&SYSSPLV' LT '&SP').SPLW
   AIF ('&SYSSPLV' GT '&SP').SPHI

SPLW ANOP
** MNOTE 0,'GENERATED FOR MACRO LEVEL &SP ON MVS &SYSSPLV'
** AGO .SPEQ
    AGO .SPHI

SPHI ANOP
   MNOTE 4,'MACRO LEVEL &SP ATTEMPTED ON MVS &SYSSPLV'
   MNOTE 4,'MACRO LEVEL RESET TO &SYSSPLV'
   &SP  SETC '&SYSSPLV'
    AGO .SPEQ

SPEQ ANOP
   SPLEVEL SET-&SP        SET MACRO LEVEL 370, XA OR ESA
   SPLEVEL TEST SET SYSSPLV

&MVS370S SETC '&MVS370'
   AIF ('&MVS370S' EQ 'NOTSUP').SUPP
   AIF ('&MVS370S' EQ 'SUP').SUPP
   MNOTE 8,'MVS370 MUST BE INDICATED AS NOTSUP OR SUP'

SUPP ANOP
   AIF ('&SYSSPLV' GT '1').XASUPP XA/ESA-MACRO LEVEL

&MVS370S SETC 'SUP' FORC MVS370 SUPPORT

XASUPP ANOP
AIF ('&GENCO' EQ 'NO').NOGENC0
PRINT NOGEN
CVT DSECT-YES,PREFIX-YES,LIST-NO CVT

CVT EQU CVTMAP
PRINT NOGEN
IHAASCB . ASCB
IHAAXSB . ASXB
IHAPSA . PSA
USING PSA,R0 DUMMY USING TO PSA
IHARB . R8
IKJTCB . TCB
IHAACEE . ACEE
IEJSCB . JSCB
IKJPSCLB . PSCB
IEFJAJCTB . JCT
IEFTCT . TCT
IKJTSB . TSB
IEESMCA . SMCA
IEFUCBOB PREFIX-YES . UCB
UCBPFLN EQU UCBCMSEG-UCB UCB PREFIX LENGTH
IEFJESECT . JESCT
IEFJSCVT . JSCVT (SSCT)
IHASDW A DSECT-YES GETMAIN WORKAREA
PRINT GEN

WORKAREA DSECT GETMAIN WORKAREA
DS CL72 SAVE AREA
AIF ('&STAXR' EQ 'NO').NOSTAXI

&STAXD DS XL(&STAXEND-&STAXLST) STAX LIST AREA
NOSTAXI ANOP
AIF ('&ESTAER' EQ 'NO').NOESTAI

&ESTAEW DS XL(&ESTAEND-&ESTALST) ESTAE PARM LIST AREA
&ESTAPM DS 4F PARM LIST TO RETRY ROUTINE
AIF ('&SECBASE' EQ '0').ESPNSEC
DS F SECONDARY BASE FOR RETRY RTN

ESPNSEC ANOP
AIF ('&TERBASE' EQ '0').ESPNTER
DS F TERTIARY BASE FOR RETRY RTN

ESPNTER ANOP
AIF ('&QARBASE' EQ '0').ESPNQAR
DS F QUARTERNARY BASE FOR RETRY RTN

ESPNQAR ANOP

RETCODE DS A RETURN CODE
PARMADDR DS A ADDR OF PARMLIST
DATAADDR DS A ADDR OF PARAMETER DATA
PARMLEN DS H LENGTH OF PARAMETER DATA
ADSPNAME DS CL8 NAME OF ADDRESS SPACE
OPTIONS DS C EXECUTION OPTIONS
ATTN EQU X'80' ATTN FLAG SET
TSOCMD EQU X'40' INDICATE CALLED AS TSO COMMAND
EXECCALL EQU X'20' INDICATE JCL-EXEC OR TSO-CALL
SUBRUTIN EQU X'10' INDICATE CALLED AS SUBROUTINE
STC EQU X'08' STARTED TASK ADDR SPACE
TSO EQU X'04' TSO ADDR SPACE (FOREGROUND)
APPC EQU X'02' APPC TYPE ADDR SPACE
JOB EQU X'01' JOB TYPE ADDR SPACE
OPTIONR DS C EXECUTION OPTIONS
   AIF ('&TESTAUTH' EQ 'NO').NOATBIT
APFON EQU X'80' APF AUTHORIZED AT ENTRY
SUPVSTAT EQU X'40' IN SUPERVISOR STATE AT ENTRY
SYSKEY EQU X'20' IN SYSTEM KEY AT ENTRY
ALIASINV EQU X'10' PROGRAM INVOKED USING ALIAS
NOATBIT ANOP
IKJEFT01 EQU X'01' TSO IN FOREGROUND OR BATCH
USERWORK EQU * ADD USER DEFINITIONS HERE
   DS CL256 ALLOCATE MINIMUM 256 FOR USER
WORKLEN EQU *-WORKAREA LENGTH TO GETMAIN
&SIZEC SETC 'WORKLEN'
   AIF ('&SIZE' EQ '72').DEFSIZE
&SIZEC SETC '&SIZE'
DEFSIZE ANOP
* NOGENCO ANOP
&NAME CSECT
   AIF ('&ALEXIST' EQ 'NO').BYPALIØ
ENTRY &ALIASNM ALIAS ENTRY POINT
&ALIASNM DS 0H ALIAS ENTRY POINT
BYPALIØ ANOP
   AIF ('&REGEQU' EQ 'NO').BYPREGS
R0 EQU 0 EQUATE REGISTER 0
R1 EQU 1 EQUATE REGISTER 1
R2 EQU 2 EQUATE REGISTER 2
R3 EQU 3 EQUATE REGISTER 3
R4 EQU 4 EQUATE REGISTER 4
R5 EQU 5 EQUATE REGISTER 5
R6 EQU 6 EQUATE REGISTER 6
R7 EQU 7 EQUATE REGISTER 7
R8 EQU 8 EQUATE REGISTER 8
R9 EQU 9 EQUATE REGISTER 9
R10 EQU 10 EQUATE REGISTER 10
R11 EQU 11 EQUATE REGISTER 11
R12 EQU 12 EQUATE REGISTER 12
R13 EQU 13 EQUATE REGISTER 13
R14 EQU 14 EQUATE REGISTER 14
R15 EQU 15 EQUATE REGISTER 15
BYPREGS ANOP
   CNOP 0,4 . ALIGNMENT
   STM 14,12,12(13) . SAVE REGISTERS
PUSH USING SAVE PREVIOUS BASEREGS
USING &NAME,.15 MAKE INITIAL ADDRESSABILITY
AIF ('&SYSSPLV' LT '2').NOCHK NON-XA-MACRO LEVEL
AIF ('&AMODE' EQ 'ANY').NOCHK ACCEPT ANY AMODE
BALR 1,0 GET ADDRESSING MODE
LTR 1,1 TEST FOR 31 OR 24 BIT MODE
AIF ('&AMODE' EQ '31').CHK31
BNM &NONCHK ADDRESSING MODE OK
BSM24 (1) CHANGE TO 24 BIT MODE
AGO .NOCHK

CHK31 ANOP
BM &NONCHK ADDRESSING MODE OK
BSM31 (1) CHANGE TO 31 BIT MODE
AGO .NOCHK

&NONCHK ANOP
DS 0H SET UP BASE REG FOR EVT MODE SW
&TEMPBS DS 0H MAKE TEMPORARY ADDRESSABILITY
USING *,1 DROP 15 DROP INITIAL BASEREG
DROP &BASEREG,1 BUILD FINAL BASEREG
S &BASEREG,,&OFFSET LET BASE POINT TO ENTRY (DEBUG)
DROP 1 DROP TEMPORARY BASEREG
POP USING RESTORE PREVIOUS BASEREGS
USING &NAME,&BASEREG MAKE ADDRESSABILITY
B &START BRANCH AROUND ID AND DATE

&PNAME DC CLB'&NAME'. PROGRAM NAME
AIF ('&ALEXIST' EQ 'NO').BYPALI1
DC C'. DELIM FOR ALIAS TEST

&ALNAME DC CLB'&ALIASNM'. ALIAS PROGRAM NAME
DC C'. DELIM FOR ALIAS TEST

BYPALI1 ANOP
DC CL9'&YY..&MM..&DD'. COMPILATION DATE
DC CL5'&TM'. COMPILATION TIME

&MSIZE DC A(&SIZEC). SIZE FOR GETMAIN
&OFFSET DC A(&TEMPBS-&NAME). OFFSET FOR BASERG COMPUTATION
&START DS 0H BEGIN OF OWN CODE
AIF ('&SECBASE' EQ '0').SECN2
USING &NAME+4096,&SECBASE MAKE SECONDARY BASE REG
LA &SECBASE,2048(&BASEREG) MAKE SECONDARY BASE REG
LA &SECBASE,2048(&SECBASE) MAKE SECONDARY BASE REG

SECN2 ANOP
AIF ('&TERBASE' EQ '0').TERN2
USING &NAME+8192,&TERBASE MAKE TERTIARY BASE REG
LA &TERBASE,2048(&SECBASE) MAKE TERTIARY BASE REG
LA &TERBASE,2048(&TERBASE) MAKE TERTIARY BASE REG

TERN2 ANOP
AIF ('&QARBASE' EQ '0').QARN2
USING &NAME+8192+4096,&QARBASE MAKE QARTERNARY BASE REG
LA &QARBASE,2048(&TERBASE) MAKE QARTERNARY BASE REG
LA &QARBASE,2048(&QARBASE) MAKE QARTERNARY BASE REG
QARN2 ANOP
L 0,&MSIZE . SET UP FOR GETMAIN OF SAVE/WORK
AIF ('&MVS370S' EQ 'NOTSUP').BYPNON1
AIF ('&SYSSPLV' LT '2').SPIGETM BYPASS IF NOT XA/ESA MACLVL
TESTXA (15) . FIND OUT WHICH MODE
LTR 15,15 . TEST MODE
BP &NONXA . THEN NON XA/ESA MODE
BYPNON1 ANOP
GETMAIN RU,LV=(0).SP=&SUBPOOL,LOC-&LOC GETMAIN SAVE/WORK
AIF ('&MVS370S' EQ 'NOTSUP').BYPNON2
B &COMM . GO SET UP SAVE AREA
SPIGETM ANOP
&NONXA DS 0H.
LA 1,&SUBPOOL . INDICATE SUBPOOL NO
SLL 1,24 . INDICATE SUBPOOL NO
OR 0,1 . SET UP FOR GETMAIN
GETMAIN R,LV=(0) . GETMAIN SAVE/WORK
&COMM DS 0H.
BYPNON2 ANOP
AIF ('&SIZEC' EQ '72').NOCLEAR
AIF ('&CLEAR' EQ 'NO').NOCLEAR
LA 15,72 . LENGTH OF SAVE AREA
LA 14,0(15,1) . POINT AFTER SAVE AREA
LCR 15,15 . SET UP FOR SUBTRACT OF LENGTH
A 15,&MSIZE ADD LENGTH OF GETMAINED
XR 2,2 . SET UP FOR CLEAR OF GETMAINED
XR 3,3 . SET UP FOR CLEAR OF GETMAINED
MVCL 14,2 . CLEAR GETMAINED
NOCLEAR ANOP
ST 13,4(1) . SAVE POINTER TO PREV SAVE
ST 1,8(13) . POINT FROM PREV
LM 14,3,12(13) . RESTORE WORK REGISTERS
L 13,8(13) . SET UP STANDARD SAVE POINTER
AIF ('&GENCO' EQ 'NO').NOGENCP
USING WORKAREA,R13 ADDRESS WORKAREA
ST R1,PARMADDR SAVE ADDR OF PARMLIST
AIF ('&ESTAER' EQ 'NO').NOESTA2
LA R0,&RETRYR1 RETRY ROUTINE - NO SDWA
ST R0,&ESTAPM STORE IN PARAMETER LIST
LA R0,&RETRYR2 RETRY ROUTINE WITH SDWA
ST R0,&ESTAPM+4 STORE IN PARAMETER LIST
STM R12,R13,&ESTAPM+8 STORE BASE & DATA REG IN PARM
AIF ('&SECBASE' EQ '0').ESTNSEC
ST &SECBASE,&ESTAPM+8+8 STORE SECONDARY BASE IN PARM
ESTNSEC ANOP
AIF ('&TERBASE' EQ '0').ESTNTER
ST &TERBASE,&ESTAPM+8+12 STORE TERTIARY BASE IN PARM
ESTNTER ANOP
AIF ('&QARBASE' EQ '0').ESTNOAR

ST &QARBASE,&ESTAPM+8+16 STORE QUARTERNARY BASE IN PARM

ESTNQAR ANOP
MVC &ESTAEW,&ESTALST MOVE IN ESTAE PRMLST
ESTAE ,CT,PARAM=&ESTAPM,MF=(E,&ESTAEW) SETUP RCVRY

NOESTA2 ANOP
AIF ('&STAXR' EQ 'NO').NOSTAX2
MVC &STAXD,&STAXLST MOVE IN STAX LIST
STAX ,USADDR=WORKAREA,MF=(E,&STAXD) SET ATTN EXIT

NOSTAX2 ANOP
L R15,PSATOLD GET TCB-ADDRESS
L R15,TCBJSCB-TCB(R15) GET JSCB-ADDRESS
ICM R15,15,JSCBPSCB-IEZJSCB(R15) GET PSCB-ADDRESS
BZ &NTFGBG IF NOT TSO IN FG OR BATCH
OI OPTIONR,IKJEFT01 SET TSO FOREGROUND OR BATCH

&NTFGBG DS ØH.
JOBTYPE . GET TYPE OF ADDR SPACE
EX R15,&ORTYPE INSERT TYPE OF ADDR SPACE
LTR R15,R15 TEST FOR JOB
BNZ &NTJBTP NOT JOB ADDR SPACE
OI OPTIONS,JOB SET JOB ADDR SPACE

&NTJBTP DS ØH.
JOBNAME . GET NAME OF ADDR SPACE
MVC ADSPNAME,Ø(R15) SAVE NAME OF ADDR SPACE
ICM R15,15,PARMADDR GET ADDR TO INPUT PARM
BZ &SUBRTN NO PARAMETERS AT ALL
L R2,Ø(R15) GET PARM ADDR
AIF ('&SCANDATA' EQ 'NO').NOSCNI
XR R15,R15 CLEAR BEFORE INSERT
ICM R15,3,Ø(R2) GET PARM LENGTH
STH R15,PARMLEN SAVE LENGTH OF INPUT
LA R14,4+&IDLEN GET LENGTH OF PGM NAME + HDR
AIF ('&ALEXIST' EQ 'NO').BYPALI5
LA R1,4+&ALIASLN GET LENGTH OF ALIAS NAME + HDR
CR R14,R1 IS PGM OR ALIAS SHORTEST
BNH &PGMNAME PGMNAME WAS SHORTEST
LR R14,R1 USE ALIAS NAME AS SHORTEST

&PGMLONG DS ØH.
BYPALI5 ANOP
LA R1,4(R2) POINT TO EVENTUAL CMD-NAME

&CMDSCN DS ØH.
CR R15,R14 ANY ROOM FOR LEN + CMDNAME
BL &SUBRTN IF NOT, TRY SUBROUTINE
AIF ('&ALEXIST' NE 'NO').BYPALI7
CLC Ø(&IDLEN,R1),&PGNAME TSO COMMAND
BE &CMDFND FOUND CMD-NAME

BYPALI7 ANOP
AIF ('&ALEXIST' EQ 'NO').BYPALI2
CLC Ø(&IDLEN+1,R1),&PGNAME TSO COMMAND
BE &CMDFND FOUND CMD-NAME
CLC Ø(&ALIASLN+1,R1),&ALNAME TSO COMMAND
BNE &NCMDFND NOT FOUND CMD-NAME
OI OPTIONR,ALIASINV INDICATE INVOKED UNDER ALIAS
B &CMDFND FOUND CMD-NAME

&NCMDFND DS ØH .
BYPALI2 ANOP
CLI Ø(R1),C ' ' BLANK BEFORE CMD-NAME
BNE &SUBRTN TRY SUBROUTINE
LA R1,1(R1) POINT TO NEXT IN INPUT
LA R14,1(R14) COUNT UP LENGTH OF PREFIX
B &CMDSCN RECYCLE

&CMDFND DS ØH .
XR R1,R1 CLEAR WORK REGISTER
ICM R1,3,2(R2) GET OFFSET TO DATA
LA R14,&IDLEN GET LENGTH OF CMDNAME
AIF ('&ALEXIST' EQ 'NO').BYPALI6
TM OPTIONR,ALIASINV INVOKED UNDER ALIAS
BZ &NOINVAL NOT UNDER ALIAS
LA R14,&ALIASLN GET LENGTH OF CMDNAME

&NOINVAL DS ØH .
BYPALI6 ANOP
CR R1,R14 OFFSET TOO LOW
BNL &OFFNTL OFFSET NOT LESS THAN CMDNAME
STH R14,2(R2) ADJUST OFFSET TO LENGTH OF CMD
B &OFFADJ GO COMPUTE REAL DATA OFFSET

&OFFNTL DS ØH .
LA R1,4(R1) ADJUST FOR LENGTH OF HDR
CR R1,R15 OFFSET TOO HIGH
BL &OFFNTH OFFSET LESS THAN LENGTH OF BUFR
STH R14,2(R2) ADJUST OFFSET TO LENGTH OF CMD

&OFFADJ DS ØH .
LA R14,4(R14) ACCOUNT FOR LENGTH FIELDS
LA R1,0(R14,R2) GET ADDR OF DATA
SR R14,R15 COMPUTE LENGTH OF DATA
BNM &OFFNTH NO DATA AT ALL
LCR R14,R14 GET POSITIVE LENGTH OF DATA

&OFFSCN DS ØH .
CLI Ø(R1),C ' ' LEADING BLANK
BNE &ENDOFF END SCAN FOR LEADING BLANK
LA R1,1(R1) POINT TO NEXT IN PARAMETER
BCT R14,&OFFSCN RECYCLE SCAN

&ENDOFF DS ØH .
STH R14,PARMLEN SAVE REAL LENGTH OF INPUT
ST R1,DATAADDR SAVE ADDR OF DATA
LR R1,R15 GET BUFFER LENGTH
LA R14,4(R14) ADJUST FOR HDR
SR R1,R14 COMPUTE REAL OFFSET
STH R1,2(R2) SAVE REAL OFFSET

&OFFNTH DS ØH .
XR R1,R1 CLEAR WORK REGISTER
ICM R1,3,2(R2) GET OFFSET TO DATA
LA R1,4(R1) ACCOUNT FOR LENGTH FIELDS
SR R15,R1 REDUCE BY LENGTH OF HEADER
STH R15,PARMLEN SAVE LENGTH OF INPUT
BP &TSODTA IF ANY DATA
XC PARMLEN,PARMLEN CLEAR LENGTH OF DATA
B &TSOCMN IF ZERO (OR NEGATIVE) NO DATA

&TSODTA DS $0H.

LA R14.0(R1,R2) GET ADDR OF DATA
ST R14,DATAADDR SAVE ADDR OF DATA
OI OPTIONS,TSOCMD INDICATE CALLED AS TSOCOMMAND
B &TSOCMN PROCEED

NOSCN1 ANOP
&ORTYPE OI OPTIONS,0 EXECUTED OR OF ADDR SPACE TYPE
&SUBRTN DS $0H.

OI OPTIONS,SUBRTIN INDICATE CALLED AS SUBROUTINE
ICM R15,15,PARMADDR GET ADDR TO INPUT PARM
BZ &TSOCMN NO PARAMETERS AT ALL
AIF ('$SCANDATA' EQ 'NO').NOSCN2

LA R14,2(R2) GET ADDR OF DATA
XR R15,R15 CLEAR WORK REGISTER
ICM R15,3,PARMLEN GET MAX PARM LENGTH
BZ &ENDPSN DONT SCAN IF NO DATA

&PRMSCN DS $0H.

CLI 0(R14),C' ' LEADING BLANK
BNE &ENDPSN END SCAN FOR LEADING BLANK
LA R14,1(R14) POINT TO NEXT IN PARAMETER
BCT R15,&PRMSCN RECYCLE SCAN

&ENDPSN DS $0H.

ST R14,DATAADDR SAVE ADDR OF DATA
STH R15,PARMLEN SAVE REAL LENGTH OF INPUT

NOSCN2 ANOP

L R1,PARMADDR GET ADDR TO INPUT PARM
ICM R1,15.0(R1) GET FIRST PARM ADDR
BZ &TSOCMN IT WAS NOT LAST PARM

* DECIDE IF PROGRAM IS DIRECTLY EXECUTED VIA JCL-EXEC OR TSO-CALL;
* AND SET OPTIONS ACCORDINGLY:

L R1,PSATOLD GET TCB ADDR
USING TCB,R1 ADDRESS TCB
L R1,TCBRBP GET RB POINTER
USING RBASIC,R1 ADDRESS REQUEST BLOCK
AIF ('&ALEXIST' NE 'NO').BYPALI3

CLC RBXSAVE(&IDLEN),&PGNAME DIRECT EXECUTE OF PGM-NAME
BNE &TSOCMN NOT EXEC PGM- OR TSO CALL

BYPALI3 ANOP

AIF ('&ALEXIST' EQ 'NO').BYPALI4
CLC RBXSAVE(&IDLEN+1),&PGNAME DIRECT EXECUTE OF PGM-NAME
BE &SONEPM EXEC PGM- OR TSO CALL
CLC RBXSAVE(&ALIASLN+1),&ALNAME DIRECT EXEC OF ALIAS-NAME
BNE &TSOCMN NOT EXEC PGM- OR TSO CALL
OI OPTIONR,ALIASINV INDICATE INVOKED UNDER ALIAS
&SONEPM DS $0H.
BYPALI4 ANOP
TM RBSTAB2,RBTCBNXT DOES RBLINK POINT TO TCB
BO &NOMORRB NO MORE RBS
ICM R1,15,RBLINK POINT TO NEXT RB
BZ &NOMORRB NO MORE RBS
C R1,PSATOLD RBLINK POINTS TO TCB
BE &NOMORRB NO MORE RBS
B &TSOCMN THERE WAS A HIGHER PRB
&NOMORRB DS $0H.
O1 OPTIONS.EXECALL INDICATE JCL-EXEC OR TSO-CALL
NI OPTIONS,255-SUBRUTIN TURN OFF SUBROUTINE
&TSOCMN DS $0H.
*
* NORMAL PROCESSING
TM OPTIONS,ATTN IS ATTN FLAG SET
BO EXIT RETURN IF ATTN
AIF ('&SCANDATA' EQ 'NO').NOSCN3
XR R15,R15 CLEAR WORK REGISTER
ICM R15,3,PARMLEN GET LENGTH OF DATA
L R14,DATAADDR GET ADDR OF 1ST DATA PARAMETER
* REMOVE TRAILING BLANKS
LA R1,0(R14,R15) POINT AFTER DATA
&BACKSC DS $0H.
BCTR R1,0 POINT TO PREVIOUS BYTE
LTR R15,R15 ANY DATA LEFT
BNP &ENDBCS NO MORE BACKSCAN
CLI @R1),C' ' LOOK FOR TRAILING BLANK
BNE &ENDBCS NO MORE BACKSCAN
BCT R15,&BACKSC RESCAN FOR TRAILING BLANK
&ENDBCS DS $0H.
STH R15,PARMLEN SAVE REAL LENGTH OF INPUT
AIF ('&XLATEF' EQ 'NO').NOXLATE
LA R1,256 GET MAX LENGTH TO XLATE
&TRCYCL DS $0H.
CR R15,R1 MORE THAN 256 TO XLATE
BNH &XLTLST GO TO LAST XLATE
TR @(256,R14),&TRTAB XLATE A BATCH OF 256 BYTES
SR R15,R1 COUNT DOWN ALREADY XLATED
LA R14,256(R14) STEP BEHIND
B &TRCYCL AND RECYCLE
&XLTLST DS $0H
LTR R15,R15 TEST FOR ZERO LENGTH
BZ &ENDXLTX NO MORE TO TRANSLATE
BCTR R15,0 REDUCE FOR EXECUTE
EX R15,&TRLATE TRANSLATE TO UPPER
&ENDXLTX DS $0H
NOXLATE ANOP
NOSCN3 ANOP
AIF ('&TSTAUT' EQ 'NO').NOTSTAT
TESTAUTH FCTN=1,BRANCH=NOT,TEST FOR APF
LTR R15,R15 TEST FOR APF ON
BNZ &NTSTAT1 APF NOT ON
OI OPTIONR,APFON INDICATE APF AUTHORIZED

&NTSTAT1 DS 0H.
TESTAUTH KEY=NO,STATE=STATE,RELEVEL=1,BRANCH=NON TEST FOR STATE
LTR R15,R15 TEST FOR SUPERVISOR STATE
BNZ &NTSTAT2 NOT IN SUPERVISOR STATE
OI OPTIONR,SUPVSTAT INDICATE IN SUPERVISOR STATE

&NTSTAT2 DS 0H.
TESTAUTH STATE=NO,KEY=STATE,RELEVEL=1,BRANCH=NON TEST FOR KEYS
LTR R15,R15 TEST FOR KEYS 0 - 7
BNZ &NTSTAT3 IN KEY 0 TO 7
OI OPTIONR,SYSKEY INDICATE IN SYSTEM KEY

&NTSTAT3 DS 0H.

NOTSTAT ANOP
L R15,R13 GET SAVE POINTER TO PREV SAVE
AIF ('&TSTAUT' EO 'YES').YTSTAT
LM R0,R2.12+8(R15) RESTORE WORK REGISTERS

YTSTAT ANOP
AIF ('&TSTAUT' NE 'YES').STSTAT
LM R0,R3.12+8(R15) RESTORE WORK REGISTERS

STSTAT ANOP
XR R15,R15 CLEAR WORK REGISTER
ICM R15,3,PARMLEN GET LENGTH OF DATA
L R14,DATAADDR GET ADDR OF 1ST DATA PARAMETER

NOTSTAT ANOP
AIF ('&SYSSPLV' LT '2').ONLY370 ONLY MVS/370 CODE
AIF ('&MVS370S' EQ 'SUP').BOTH BOTH XA/ESA AND 370 CODE

ONLY370 ANOP
MNOTE 4,'NOTE —> ONLY MVS/370 CODE GENERATED'
MEXIT

BOTH ANOP
MNOTE 4,'NOTE —> BOTH MVS/XA/ESA AND MVS/370 CODE GENERATED'
MEND

TESTXA MACRO
* TESTS FOR RUNNING UNDER XA/ESA, AND TEST FOR ADDRESSING MODE
* RETURNS RX = 4 FOR MVS/370
* RETURNS RX = 0 FOR RUNNING UNDER XA/ESA IN 24 BIT MODE
* RETURNS RX = X'80000000' (MINUS) FOR RUNNING IN 31 BIT MODE
* RX DEFAULTS TO R15
* ANOTHER REGISTER CAN BE USED AS TESTXA (RY)
* CODE FOR SUPPORT OF NON-XA (MVS/370) WILL ONLY BE GENERATED IF
* GLOBAL VARIABLE FROM INITR &MVS370S—SUP IS SPECIFIED OR &SPELEVEL=1;
* IF MACRO INITR IS NOT USED AND &SLEVEL > 1, IT IS STILL POSSIBLE * TO FORCE GENERATION OF MVS/370 VIA THE PARAMETER MVS370-SUP. * CODE FOR SUPPORT OF XA/ESA WILL ONLY BE GENERATED IF &SLEVEL > 1. *

MACRO
&NAME TESTXA &REG,&MVS370=NOTSUP
GBLC &MVS370S COMES FROM INITR IF THIS MACRO IS USED
GBLC &SYSXPLV MACRO LEVEL
SLEVEL TEST SET SYSXPLV
LCLC &NONXXA
&NONXXA SETC 'TAA'.'&SYSXPLV' AIF('&MVS370S' NE '').INTSUPP
&MVS370S SETC 'MVS370'. SET ONLY FROM PARAMETER IF INITR IS NOT USED
INTSUPP ANOP
AIF('&MVS370S' EQ 'NOTSUP').SUPP
AIF('&MVS370S' EQ 'SUP').SUPP
MNOTE 0 'MVS370 MUST BE INDICATED AS NOTSUP OR SUP'
MEXIT
SUPP ANOP
AIF('&SYSXPLV' GT '1').XASUPP XA-MACRO LEVEL
&MVS370S SETC 'SUP' FORCE MVS370 SUPPORT
XASUPP ANOP
AIF('&REG' EQ '').RNUL
AIF('&REG'(1,1) EQ ' ').AREG
AREG ANOP
&AREG SETC 'AREG(1)' AGO .RNUL
AREG ANOP
&AREG SETC 'AREG(1)' AGO .REG
RNUL ANOP
&AREG SETC '15'
REG ANOP
&NAME DS OH
AIF('&MVS370S' EQ 'NOTSUP').XA
AIF('&SYSXPLV' LT '2').NONXA BYPASS IF NOT XA/ESA MACLEVEL
LA &REG,X'10'. GET CVT ADDK
TM X'74'(AREGR),X'80'. TEST CVTDCB FOR UNDER XA CTVMVSSE
NONXA ANOP
LA &AREG,4 . INDICATE MVS/370
AIF('&SYSXPLV' LT '2').BYPNON2 BYPASS IF NOT XA/ESA MACLEVEL
BN0 &NONKA . NOT UNDER XA
XA ANOP
SR &AREG,&AREG ZERO OUT REGISTER
BMS &AREG,0 . SAVE ADDRESSING MODE IN REG
AIF('&MVS370S' EQ 'NOTSUP').BYPNON2
&NONKA DS OH
BYPNON2 ANOP
MEXIT
MEND

Nils Plum
Systems Programmer (Denmark) © Xephon 1997

Sterling Software has begun shipping SAMS:Recover. SAMS:Recover for OS/390 integrates disaster recovery and normal daily back-up into a single, non-redundant process.

For further information contact:
Sterling Software Inc, 11050 White Rock Road, Suite 100, Rancho Cordova, CA 95670-6095, USA.
Tel: (916) 635 5535
Fax: (916) 635 5604 or
Sterling Software Ltd, 1 Longwalk Road, Stockley Park, Uxbridge, Middlesex, UB11 1DB, UK.
Tel: (01895) 8678000
Fax: (01895) 8678001.

***

Macro 4 has launched a new version of Dumpmaster. Dumpmaster MVS Version 4.3 is a dump analysis and fault diagnosis product with specific emphasis on addressing year 2000 and single European currency issues. The software is out now, but no prices have been announced.

For further information contact:
Macro 4, The Orangery, Turners Hill Road, Worth, Crawley, West Sussex, RH10 4SN, UK.
Tel: (01293) 886060 or
Macro 4, 35 Waterview Blvd, PO Box 292, Parsippany, NJ 07054-0292, USA.
Tel: (201) 402 8000.

***

TIBCO Software has announced that its industrial-strength messaging software, TIB/Rendezvous 4.0, now supports IBM and plug-compatible systems running the MVS or OS/400 operating systems. It features push-based message queuing, publish/subscribe, subject-based addressing, and fully-distributed queuing. It provides a platform for large-scale information or content distribution over private networks or the Internet. TIB/Rendezvous 4.0 is available now. Pricing starts at $3600 per client or desktop at runtime and $2,500 for a developer’s kit. Volume discounts are available.

For further information contact:
TIBCO Software Inc, 3165 Porter Drive, Palo Alto, CA 94304, USA.
Tel: (415) 846 5900
Fax: (415) 846 5905.

***

Asi has announced PKZIP MVS PLUS! The product features year 2000 compliance, HyperZip, enhanced file handlers, magnetic tape handlers, and an ISPF panel.

For further information contact:
Asi, 9009 Springboro Pike, Miamisburg, OH 45342, USA.
Tel: (937) 847 2374 Ext. 14
Fax: (937) 847 2375.

***