209

# MVS update

February 2004

## In this issue

# *MVS Update*

## Published by

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

## North American office

Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

## Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1999 issue, are available separately to subscribers for £29.00 ($43.50) each including postage.

## *MVS Update* on-line

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

## Editor

Trevor Eddolls

## Disclaimer

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

## Contributions

When Xephon is given copyright, articles published in *MVS Update* are paid for at the rate of £100 ($160) per 1000 words and £50 ($80) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £20 ($32) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

*Printed in England.*

# Format and display a data field from Assembler

## BACKGROUND

Often during testing and for one-off applications, it is useful to have an easy means of displaying, from an Assembler program, the contents of a field, converted to a displayable format when necessary (such as for binary or hexadecimal fields) – ie something similar to the COBOL DISPLAY instruction or the C printf() function.

## SOLUTION

The DISPLAY macro described in this article outputs the contents of a specified field to the job log (routing code 11). The use of the WTO macro obviates the need to specify a DD statement and also has 31-bit capability. Furthermore, it can be used in two popular environments – batch and TSO.

To reduce the footprint of the generated code when the macro is used more than once in a program, the code used to perform the formatting and output is included just once. Similarly, the use of sparse translation tables (not all 256 bytes defined) reduces the size, but means that the first macro call should not be placed too near the start of the program, otherwise addressing errors may occur (if necessary, the appropriate padding must be included).

To improve the utility, the field name is also output (not for literals).

## RUN-TIME ENVIRONMENT

The DISPLAY macro can run in batch and TSO.

Note: the macro could easily be extended to run in other environments, for example, CICS.

The invocation syntax is:

```
[name] DISPLAY source[,length[,type]]
```

where:

- *name* – optional label. The label applies to the source.
- *source* – source field; field name (eg ALPHA), literal (eg ‘beta gamma’) or base-displacement address (eg 4(5) = 4 byte displacement from the address contained in general purpose register 5).
- *length* – explicit length (in bytes) of the source field; either as a self-defining (numeric) value (eg 8) or as a register (specified within parentheses, eg (9)) that contains the appropriate length at execution time. The length must be specified for a base-displacement address. N = numeric (=decimal) value for a register, otherwise the register content is displayed in hexadecimal notation. If no length is specified, the implicit length is used, ie the value returned by the L attribute.
- *type* – field type. If no type is specified, the implicit type is used, ie the value returned by the T attribute. Type may be one of the following:

  C – character

  Z – zoned (decimal)

  X – hexadecimal

  P – packed decimal (signed)

  B, H, F – binary (signed)

  A – address

  R – general purpose register (0,…,15 or appropriate equate specified as *field*).

Register usage: as usual for macros, DISPLAY uses registers 14-1.

## MACRO DEFINITION

```
         MACRO
&NAME    DISPLAY &P1,&LP1,&TP1
```

```
.**
.* Format and display a data field
.**
.*  Parameters:
.*   P(1) - source field start (or literal)
.*   P(2) - source field length, either numeric literal or register (n)
.*          (if omitted, default length used)
.*          ('N' = numeric (decimal) conversion for register)
.*   P(3) - source field type (optional)
.**
.* The following field types are supported:
.*  C - character
.*  Z - zoned (decimal)
.*  X - hexadecimal
.*  P - packed decimal (signed)
.*  B, H, F - binary (signed)
.*  A - address
.*  R - register
.*  Literal (field enclosed within quotes)
.*  Explicit address (e.g. Ø(R1)), length must be specified
.**
         GBLB  &FD
         LCLA  &L
         LCLC  &LN
         LCLC  &C,&W,&MK
.* label
         AIF   (T'&NAME EQ 'O').AØ
&NAME    DS    ØH
.AØ      ANOP
         MVC   ##WK,##WK-1 clear
.* 1st CALL?
         AIF   (&FD).A1
&FD      SETB  1
         B     ##GO1
         SPACE 1
##FD     DS    PL8
##MK1    DC    X'Ø1Ø3Ø7ØF'
##MK2    DC    X'2Ø4Ø7Ø9Ø'
         SPACE 1
         DC    C' '
##OUT    DS    ØCL8Ø
##LEN    DS    HL2
##NAME   DS    CL8
         SPACE 1
         DC    C' '
##WK     DS    ØCL71
##WKS    DC    C' ' SIGN
##WKFLD  DS    CL7Ø
         SPACE 1
##FTR    DC    CL16'Ø123456789ABCDEF'
```

```
         SPACE 2
##GO1    DS    ØH
.A1      MVC   ##OUT,##OUT-1
.*
&TP      SETC  T'&P1
         AIF   (T'&TP1 EQ 'O').A1A
&TP      SETC  '&TP1'
.A1A     ANOP
.*
         SPACE 1
&C       SETC  '&P1'(1,1)
         AIF   ('&C' EQ '''').B4
&L       SETA  K'&P1
         MVC   ##NAME,=CL8'&P1'
         AIF   ('&TP' EQ 'H').B7
         AIF   ('&TP' EQ 'F').B7
         AIF   ('&TP' NE 'B').B1
.B7      ANOP
.* binary
&LN      SETC  'L''&P1-1'
         AIF   (T'&LP1 EQ 'O').B7A
&LN      SETC  '&LP1-1'
.B7A     SR    Ø,Ø
         LA    1,&LN
         IC    1,##MK1(1)
         ICM   Ø,Ø,&P1
         EX    1,*-4
         CVD   Ø,##FD
         LA    1,&LN
         IC    1,##MK2(1)
         AGO   .A2
.B1      AIF   ('&TP' NE 'P').B5
.* packed decimal
         ZAP   ##FD,&P1
&LN      SETC  'L''&P1*2-1'
         AIF   (T'&LP1 EQ 'O').B1A
&LN      SETC  '&LP1*2-1'
.B1A     LA    1,(&LN)*16
.A2      MVI   ##WKS,C'+'
         CP    ##FD,=P'Ø'
         BNL   *+8
         MVI   ##WKS,C'-'
         OI    ##FD+7,X'ØF'
         UNPK  ##WKFLD(Ø),##FD
         EX    1,*-6
         AGO   .MPUT
.B5      ANOP
         AIF   ('&TP' EQ 'C').B5B
         AIF   ('&TP' NE 'Z').B6
.B5B     ANOP
```

```
.* character or zoned decimal
&LN      SETC  'L''&P1'
         AIF   (T'&LP1 EQ 'O').B5A
&C       SETC  '&LP1'(1,1)
         AIF   ('&C' NE '(').B5D
         LR    1,&LP1
         BCTR  1,Ø
         AGO   .B5C
.B5D     ANOP
&LN      SETC  '&LP1'
.B5A     LA    1,&LN-1
.B5C     LA    Ø,(L'##WKFLD-1)
         CR    1,Ø
         BNH   *+6
         LR    1,Ø
         MVC   ##WKFLD(Ø),&P1
         EX    1,*-6
         AGO   .MPUT
.B6      AIF   ('&TP' NE 'X').B8
.* hexadecimal
.B6C     ANOP
&LN      SETC  'L''&P1'
&P       SETC  '&P1'
         AIF   (T'&LP1 EQ 'O').B6A
&C       SETC  '&LP1'(1,1)
         AIF   ('&C' NE '(').B6D
         LR    Ø,&LP1
         AGO   .B6E
.B6D     ANOP
&LN      SETC  '&LP1'
.B6A     LA    Ø,&LN
.B6E     LA    1,(L'##WKFLD/2)
         CR    Ø,1
         BNH   *+6
         LR    Ø,1
         LA    1,&P
.B6B     LA    15,##WKFLD
         UNPK  ##FD(3),Ø(2,1)
         TR    ##FD(2),##FTR-X'FØ'
         MVC   Ø(2,15),##FD
         LA    1,1(1)
         LA    15,2(15)
         BCT   Ø,*-26
         AGO   .MPUT
.B4      ANOP
.* literal
&L       SETA  K'&P1-2
         MVC   ##WKFLD(&L),=C&P1
         AGO   .MPUT
.MPUT    SPACE
```

```
         MVC   ##LEN,=AL2(L'##WK)
         WTO   TEXT=((##OUT,D)),ROUTCDE=11
         MEXIT
.B8      AIF   ('&TP' NE 'R').B9
.* register
         AIF   ('&LP1' EQ 'N').B8A    decimal
         ST    &P1,##FD+4
         LA    Ø,4
         LA    1,##FD+4
         AGO   .B6B
.*
.B8A     CVD   &P1,##FD
         LA    1,11*16
         AGO   .A2
.*
.B9      AIF   ('&TP' NE 'A').B1Ø
.* ADDRESS
         MVC   ##FD+4(4),&P1
         AGO   .B6C
.*
.B1Ø     AIF   ('&TP' NE 'U').E1
         AIF   ('&C' LT 'Ø').E1
.* explicit address
.* type hexadecimal (implicit)
         LA    15,&P1
&P       SETC  'Ø(15)'
         AGO   .B6D
.*
.E1      MNOTE 8,'*** INVALID DATA TYPE ***'
         MEXIT
.E2      MNOTE 8,'*** INVALID LENGTH ***'
         MEND
```

## SAMPLE CODE FRAGMENT

```
…
         LA      15,2Ø
         DISPLAY 15,,R      R15 (hex)
         LA      15,2Ø
         DISPLAY 15,N,R     R15 (decimal)
         DISPLAY 'tag'      literal
         DISPLAY PID
         DISPLAY FNO,1,X
         DISPLAY FDATA,8,C
         LA      2,TEXT     set base address
         DISPLAY 5(2),4,C
         DISPLAY CTR        packed decimal
         DISPLAY ZCTR       zoned decimal (with sign)
         LA      2,4        data length
         DISPLAY text,(2)   truncate
         DISPLAY text       complete field
…
PID      DC    CL4'1234'
FLD      DS    ØCL256
FNO      DC    AL1(8)
FLEN     DC    AL1(16)
FDATA    DC    CL254'alpha'
TEXT     DC    C'beta gamma'
CTR      DC    P'-79'
```

## ASSOCIATED OUTPUT

```
15        ØØØØØØ14
15       +ØØØØØØØØØØ2Ø
          tag
PID       1234
FNO       Ø8
FDATA     alpha
5(2)      gamm
CTR      -ØØ79
ZCTR      7H
text      beta
text      beta gamma
```

*A Rudd*
*Systems Programmer (Germany)*

# Researching CHPID problems

CHPID problems can point to serious I/O problems that can affect DASD, tape, or communication devices. There are many messages that can identify CHPID problems. This article was originally written for operations and shows how to determine whether a CHPID problem is a major or minor concern.

## WHAT IS A CHPID?

A CHPID is a Channel Path ID. MVS has always had the ability to use channels, control units, and devices to accomplish input/

output (I/O) operations. A device (like DASD, tape, printers, etc) is always represented in the operating system as a Unit Control Block (UCB). Devices are connected to control units and control units are attached to the mainframe with channels. The pre-MVS/XA naming convention for UCBs enforced a three-digit numbering scheme and was made up of the one-digit channel, plus a one-digit control unit, plus a one-digit device number (eg A26 – channel A, control unit 2, device number 6). The hardware and software architecture allowed for only 4,096 I/O devices per mainframe. When MVS evolved to MVS/XA (early 1980s), the I/O subsystem was enhanced to allow for more than 256 devices per channel and up to eight paths to each device. With MVS/XA, the I/O subsystem was significantly enhanced and the ability to use four-digit UCBs allowed the addition of over 65,000 I/O devices. The old naming conventions were abandoned and the introduction of a new logical mapping of a physical channel to a logical path was now necessary. Hence the creation of Channel Path IDs, or CHPIDs, to help us exploit the more powerful I/O subsystem. So a CHPID is a logical path from a device to a physical channel. With current control unit technology, each device can have up to eight physical paths to perform I/O.

## WHAT DOES A CHPID FAILURE MEAN?

A CHPID failure means a physical channel has failed or had a severe problem. Since most channels these days are ESCON or FICON, a failure is usually associated with a ‘loss of light’. If there are many devices on this channel, it may be a major problem. If there are only a few devices on the channel, or if all the devices on the channel have multiple alternative paths through unaffected channels, this may not be a major problem. Since each channel can support multiple devices and each device can ‘ride’ multiple channels, it is necessary to know what devices are on which channels.

## HOW DO WE KNOW WE HAVE A CHPID PROBLEM?

The most likely indication of a CHPID problem will be a message on the console or an automation alert. Occasionally, the CEC will ‘phone home’ with a CHPID problem and IBM will call. If the IBM Support Center calls to report a problem, we will usually have seen an alert for the CHPID error and problem determination should already be in progress. IBM will usually tell us which CEC reported the problem. The IBM Support Center does not know our CECs’ names; they will give us the IBM serial number for the box. Always match serial numbers to CECs to determine the affected LPARs. Armed with this information, always check to see whether any changes are in progress before escalating.

## HOW CAN WE DETERMINE WHAT IS ON A CHANNEL/CHPID?

We have several MVS commands to trace devices. We can trace from the device back to the channel or from the channel down to the device. The approach we will use depends on the type of message we receive and which direction we have to research.

## USING MVS COMMANDS TO RESEARCH DEVICES AND CHPIDS

Suppose we get a device error message like:

```
IOSØØØI 87D4,19,IOE,Ø2,Ø6ØØ,,**,,HSM
```

First, we would use the *Messages and Code* manual (or MVS/ Quickref) to determine what the message meant. This particular message will always contain the device number (also known as a UCB). 87D4 is the device number and 19 is the CHPID. Next, you might want to determine what kind of device this is by using the DISPLAY UNIT command:

```
D U,,,87D4,1

IEE457I Ø7.27.Ø5 UNIT STATUS 42Ø
UNIT TYPE STATUS        VOLSER     VOLSTATE
87D4 359L O    -M                      /REMOV
```

This device is a 359L (logical 3590 in a virtual tape server). We tend to keep the same types of device isolated on a CHPID. If one device is a tape, the others are probably tapes also. Although this is not 100% true, it is a good rule-of-thumb; but always check. The reason this is important is that it gives us a quick feel for what types of device will be affected. Depending on what type of device is on the channel, we may be more or less likely to sustain the hit.

If some other message presents a device number without the CHPID, you could also do a DISPLAY MATRIX command for the device (also called a DM DEV):

```
D M=DEV(87D4)

IEE174I Ø7.32.28 DISPLAY M 499
DEVICE 87D4   STATUS=ONLINE
CHP                   19
DEST LINK ADDRESS     64
DEST LOGICAL ADDRESS  ØØ
PATH ONLINE           Y
CHP PHYSICALLY ONLINE Y
PATH OPERATIONAL      Y
MANAGED               N
MAXIMUM MANAGED CHPID(S) ALLOWED:  Ø
ND         = ØØ359Ø.A5Ø.IBM.13.ØØØØØØØ44712
DEVICE NED = ØØ359Ø.E1A.IBM.13.ØØØØØØØ44712
```

We can see from the third line that this device is on CHPID 19 (with no alternative paths).

Most of our DASD will be configured with multiple CHPIDs for throughput and redundancy:

```
D U,,,A123,1

IEE457I Ø7.35.45 UNIT STATUS 6Ø2
UNIT TYPE STATUS        VOLSER     VOLSTATE
A123 339Ø O             1GA123     PRIV/RSDNT

D M=DEV(A123)

IEE174I Ø7.34.5Ø DISPLAY M 599
DEVICE A123   STATUS=ONLINE
CHP                   A2 D2 62 1F B6
DEST LINK ADDRESS     Ø6 Ø5 Ø4 Ø5 Ø5
DEST LOGICAL ADDRESS  Ø1 Ø1 Ø1 Ø1 Ø1
PATH ONLINE           Y  Y  Y  Y  Y
CHP PHYSICALLY ONLINE Y  Y  Y  Y  Y
PATH OPERATIONAL      Y  Y  Y  Y  Y
MANAGED               N  N  N  N  N
MAXIMUM MANAGED CHPID(S) ALLOWED:  Ø
ND         = ØØ21Ø5.   .HTC.12.ØØØØØØØ4Ø358
DEVICE NED =   21Ø5.   .HTC.12.ØØØØØØØ4Ø358
```

DASD A123 has five paths (CHPIDs A2, D2, 62, 1F, and B6). If one of these CHPIDs has a failure and all the devices on the failing CHPID are configured with the same five CHPIDs, this problem will have minimal impact. There is the potential for a 20% performance hit, but there should be no loss of functionality. This problem could most likely be deferred until after hours.

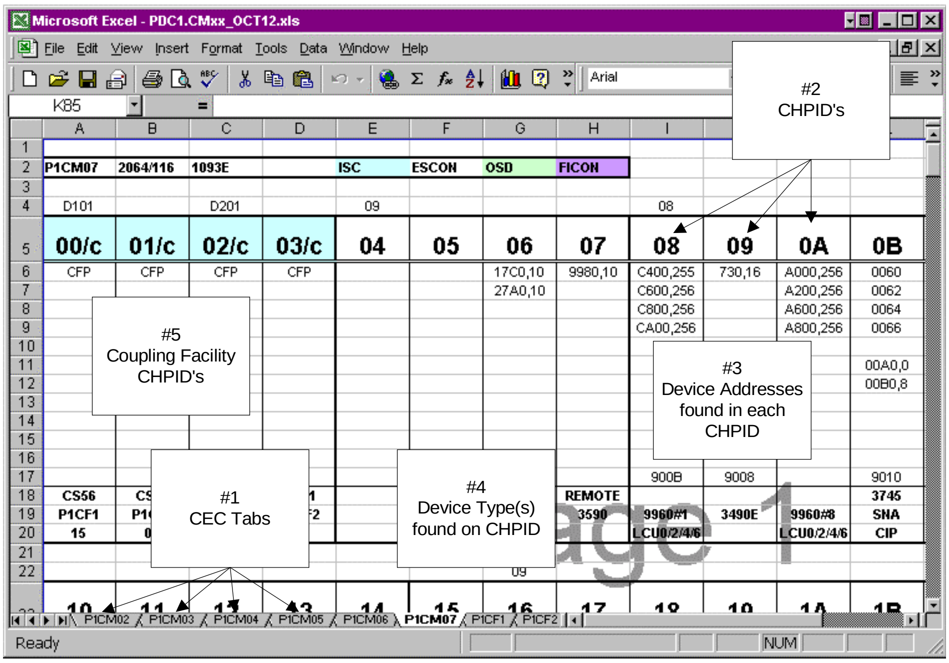
Suppose we get a message like this:

```
IOS581E LINK FAILED REPORTING CHPID=A2 INCIDENT UNIT TM=ØØ9Ø32/ØØ5
SER=IBMØ2-Ø41278 IF=ØØØ5 IC=Ø3 INCIDENT UNIT LIF=Ø9
```

This means we have detected a channel/CHPID failure. The quickest way to determine what is on the CHPID is to use the DISPLAY MATRIX command again for the CHPID.

```
D M=CHP(A2)

IEE174I Ø7.42.38 DISPLAY M 65Ø
CHPID A2:  TYPE=Ø5, DESC=ESCON SWITCHED POINT TO POINT, ONLINE
DEVICE STATUS FOR CHANNEL PATH A2
    Ø  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
A1Ø +@ +@ +  +  +  +  +  +  +  +  +  +  +  +  +  +
A11 +  +  +@ +@ +  +  +  +  +  +  +  +  +  +  +  +
A12 +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
A13 +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  $@
A14 $@ $@ $@ $@ $@ $@ $@ $@ AL AL AL AL AL AL AL AL
A15 AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
A16 AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
A17 AL AL AL AL UL AL AL UL UL UL AL AL AL AL AL AL
.
. several lines removed from the command output
.
AB8 AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
AB9 AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
ABA AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
ABB AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
ABC AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
ABD AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
ABE AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
ABF AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
SWITCH DEVICE NUMBER = 9Ø12
************************ SYMBOL EXPLANATIONS ************************
+ ONLINE    @ PATH NOT VALIDATED   - OFFLINE    . DOES NOT EXIST
* PHYSICALLY ONLINE   $ PATH NOT OPERATIONAL
BX DEVICE IS BOXED          SN SUBCHANNEL NOT AVAILABLE
DN DEVICE NOT AVAILABLE      PE SUBCHANNEL IN PERMANENT ERROR
AL DEVICE IS AN ALIAS        UL DEVICE IS AN UNBOUND ALIAS
```


Microsoft Excel - PDC1.CMxx_OCT12.xls
File Edit View Insert Format Tools Data Window Help
Arial
K85
=
#2
CHPID's
#5
Coupling Facility
CHPID's
#3
Device Addresses
found in each
CHPID
#1
CEC Tabs
#4
Device Type(s)
found on CHPID
age 1
P1CM02 P1CM03 P1CM04 P1CM05 P1CM06 P1CM07 P1CF1 P1CF2
Ready
NUM

| | A | B | C | D | E | F | G | H | I | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | | | |
| 2 | P1CM07 | 2064/116 | 1093E | | ISC | ESCON | OSD | FICON | | | | |
| 3 | | | | | | | | | | | | |
| 4 | D101 | | D201 | | 09 | | | | 08 | | | |
| 5 | 00/c | 01/c | 02/c | 03/c | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B |
| 6 | CFP | CFP | CFP | CFP | | | 17C0,10 | 9980,10 | C400,255 | 730,16 | A000,256 | 0060 |
| 7 | | | | | | | 27A0,10 | | C600,256 | | A200,256 | 0062 |
| 8 | | | | | | | | | C800,256 | | A600,256 | 0064 |
| 9 | | | | | | | | | CA00,256 | | A800,256 | 0066 |
| 10 | | | | | | | | | | | | |
| 11 | | | | | | | | | | | | 00A0,0 |
| 12 | | | | | | | | | | | | 00B0,8 |
| 13 | | | | | | | | | | | | |
| 14 | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | |
| 17 | | | | | | | | | 900B | 9008 | | 9010 |
| 18 | CS56 | CS | | 1 | | | | REMOTE | | | | 3745 |
| 19 | P1CF1 | P1 | | F2 | | | | 3590 | 9960#1 | 3490E | 9960#8 | SNA |
| 20 | 15 | 0 | | | | | | | LCU0/2/4/6 | | LCU0/2/4/6 | CIP |
| 21 | | | | | | | | | | | | |
| 22 | | | | | | | 09 | | | | | |
| 23 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B |

*Figure 1: CHPID spreadsheet*

## USING THE CHPID SPREADSHEET TO RESEARCH CHPIDS

The D M=CHP(xx) shows the larger picture of what is on the CHPID. This is very complete, but it can be overwhelming, tedious, and time-consuming to research. Most shops maintain a set of spreadsheets to document each CHPID by CEC by data centre. These spreadsheets can help identify the use of a CHPID very quickly. Figure 1 shows what our spreadsheet looks like.

Legend:

#1 Excel tabs for each CEC.

#2 CHPID numbers.

#3 Device number found on that CHPID.

#4 The device types found on that CHPID.

#5 If this is a CF CHPID, the heading will have a blue background.

To find the CHPID in question:

1 Select the tab for the correct CEC.

2 Scroll to the correct CHPID.

3 Review the device types and device addresses for the CHPID.

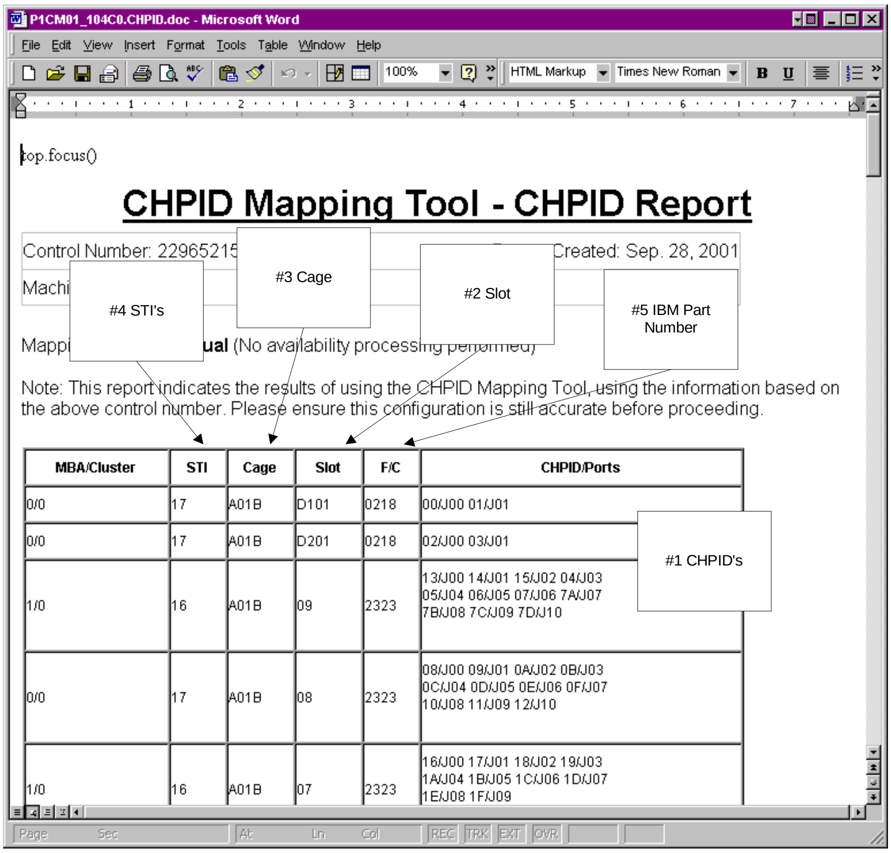
4 Determine whether known changes are in progress for this CHPID or device range.

5 Assess whether this is a problem that needs immediate attention.

## WHAT IF IBM CALLS AND SAYS WE NEED TO REPLACE AN I/O CARD?

Always match serial numbers to CECs to determine the affected LPARs before allowing IBM to do anything. The CHPID is the logical name for a channel. The channel is a fibre cable that is plugged into a port in the CEC. The CEC has ‘cages’ containing cards with ports. Each CHPID is actually a fibre-channel cable that plugs into an associated port in a card in a cage. A cage is just a frame in the CEC that holds cards. The type of port and the actual location of the port the cable plugs into is based on the IOCDS and the type of card that supports the desired channel/ device type. IBM provides different channel cards for the different types of device. For example, cards that support DASD are different from cards that support coupling facilities. Each type of card is also referred to as a Self-Timed Interface (STI). We have ordered all the appropriate STIs for our machines and ‘genned’ the system to use all those devices.

To keep all this straight, there is a set of Word documents that were provided during the IBM system assurance process when the CECs were installed (CHPID mapping tool). These are in a shared folder and show all the CHPIDs and which ports they plug into. This is important because each cage contains a different mix of cards. Some cards support multiple CHPIDs, so an error on one CHPID does not mean the STI is available for replacement. There must be research to determine whether the STI is shared


P1CM01_104C0.CHPID.doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help
100%
HTML Markup
Times New Roman
#4 STI's
#3 Cage
#2 Slot
#5 IBM Part Number
#1 CHPID's
Page Sec At Ln Col REC TRK EXT OVR

top.focus()

CHPID Mapping Tool - CHPID Report

Control Number: 22965215 Created: Sep. 28, 2001

Machi

Mappi **ual** (No availability processing performed)

Note: This report indicates the results of using the CHPID Mapping Tool, using the information based on the above control number. Please ensure this configuration is still accurate before proceeding.

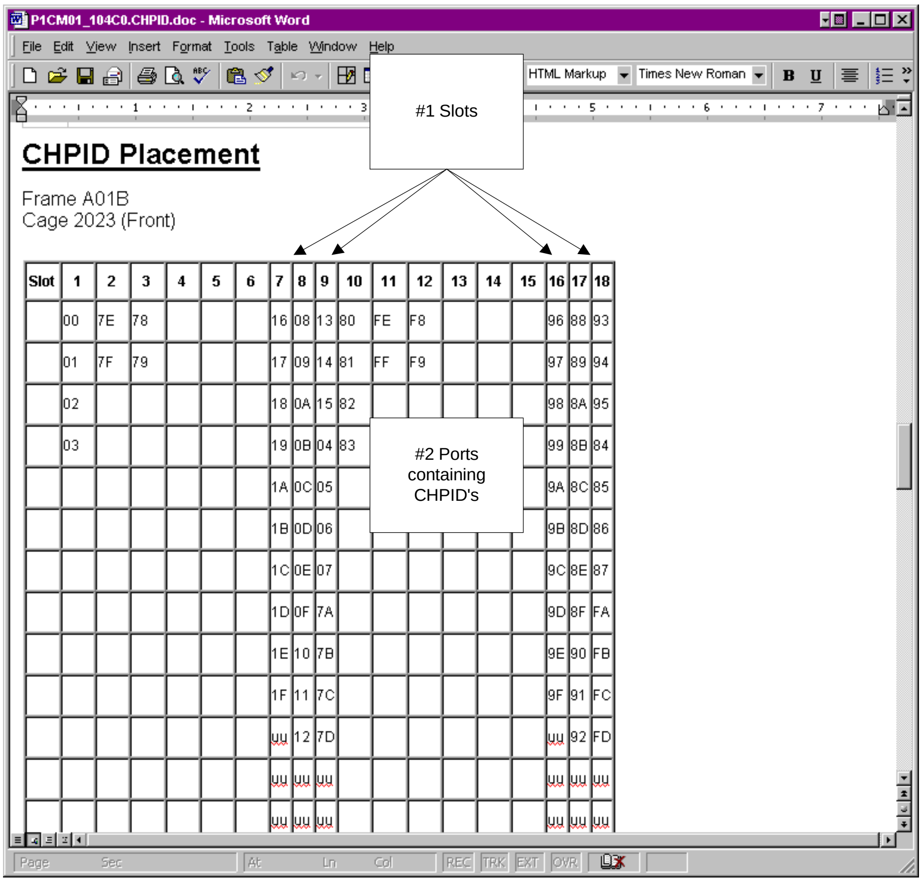
| MBA/Cluster | STI | Cage | Slot | F/C | CHPID/Ports |
|---|---|---|---|---|---|
| 0/0 | 17 | A01B | D101 | 0218 | 00/J00 01/J01 |
| 0/0 | 17 | A01B | D201 | 0218 | 02/J00 03/J01 |
| 1/0 | 16 | A01B | 09 | 2323 | 13/J00 14/J01 15/J02 04/J03 05/J04 06/J05 07/J06 7A/J07 7B/J08 7C/J09 7D/J10 |
| 0/0 | 17 | A01B | 08 | 2323 | 08/J00 09/J01 0A/J02 0B/J03 0C/J04 0D/J05 0E/J06 0F/J07 10/J08 11/J09 12/J10 |
| 1/0 | 16 | A01B | 07 | 2323 | 16/J00 17/J01 18/J02 19/J03 1A/J04 1B/J05 1C/J06 1D/J07 1E/J08 1F/J09 |

*Figure 2: Using the CHPID report*

by multiple CHPIDs. STIs are replaceable while the machine is up and running provided it is possible to VARY/CONFIG all the devices and associated CHPIDs OFFLINE. This may or may not be possible based on the devices on the CHPID. For instance if the paging packs are on a shared STI with a bad port it is not likely that this can be replaced without a maintenance window. If the STI is pulled out while other CHPIDs are active, we will have serious problems.

## FINDING AND READING THE CHPID MAPPING TOOL

The CHPID mapping tool allows you to research the location and STI for a CHPID. This should be used if IBM calls to determine whether a concurrent STI replacement can occur.



**CHPID Placement**

Frame A01B
Cage 2023 (Front)

| Slot | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 00 | 7E | 78 | | | | 16 | 08 | 13 | 80 | FE | F8 | | | | 96 | 88 | 93 |
| | 01 | 7F | 79 | | | | 17 | 09 | 14 | 81 | FF | F9 | | | | 97 | 89 | 94 |
| | 02 | | | | | | 18 | 0A | 15 | 82 | | | | | | 98 | 8A | 95 |
| | 03 | | | | | | 19 | 0B | 04 | 83 | | | | | | 99 | 8B | 84 |
| | | | | | | | 1A | 0C | 05 | | | | | | | 9A | 8C | 85 |
| | | | | | | | 1B | 0D | 06 | | | | | | | 9B | 8D | 86 |
| | | | | | | | 1C | 0E | 07 | | | | | | | 9C | 8E | 87 |
| | | | | | | | 1D | 0F | 7A | | | | | | | 9D | 8F | FA |
| | | | | | | | 1E | 10 | 7B | | | | | | | 9E | 90 | FB |
| | | | | | | | 1F | 11 | 7C | | | | | | | 9F | 91 | FC |
| | | | | | | | uu | 12 | 7D | | | | | | | uu | 92 | FD |
| | | | | | | | uu | uu | uu | | | | | | | uu | uu | uu |
| | | | | | | | uu | uu | uu | | | | | | | uu | uu | uu |

*Figure 3: CHPID  placement report*

Figure 2 shows to use the CHPID report:

#1 CHPID from the error message or CHPID being researched.

#2 The slot the cable is plugged into in the cage.

#3 The cage in the mainframe.

#4 The STI that supports the slot.

#5 The IBM part/card number.

How to determine which STI a CHPID is on:

1 Find the CHPID in question

2 Look up the slot, cage, and STI.

How to determine whether the STI is shared:

1 If more than one CHPID is listed for the STI/cage/slot, it is shared.

2 If more than one cage/slot is listed for the STI, it is shared.

Figure 3 shows how to use the CHPID placement report:

#1 The slot identified on the CHPID report.

#2 The port used by each CHPID.

How to determine whether a single port or an entire STI is bad:

1 Take the cage/slot from the CHPID report and scroll forward to the CHPID placement diagram.

2 Look up other CHPIDs on the slot (column).

3 If the others are working, this is a port problem.

How to prepare an STI for replacement without an IPL:

1 Under the direction of OSVS.

2 Research each CHPID on the STI and see if it can be VARY'd OFFLINE.

3 If so, VARY all the CHPIDs devices OFFLINE.

4 CONFIG the CHPID OFFLINE.

5 Repeat for every port/CHPID on the STI.

## HOW DOES THIS REALLY WORK?

Here is a complete example of determining whether it is possible to get all CHPIDs/devices on an STI off-line for concurrent maintenance or if a maintenance window is needed.

### Example

IBM calls and says serial number 104C0 had a hit on card number 2323 on STI 16 in cage A01B. What do you do?

1 Using the CEC to LPAR to serial number mapping report, find the CEC.

2 Using the CEC to LPAR configuration chart, determine which LPARs will be affected.

3 Find the CHPID mapping tool and open the right document for the CEC.

4 Using the CHPID report, find STI 16.

5 Still using the CHPID report, locate all CHPIDs using STI 16 (remember, an STI can span multiple slots and can contain several CHPIDs).

6 Using the CHPID spreadsheets, locate each CHPID and determine the device type.

7 Issue D M=CHP(xx) commands to determine the device statuses.

8 If DASD or TAPE and ONLINE, determine whether it is realistic to take the devices OFFLINE.

If the DASD can be taken OFFLINE:

1 VARY all the appropriate ranges OFFLINE to all LPARs on the CEC.

2 CONFIG all the CHPIDs OFFLINE to all LPARs on the CEC.

3 Turn over the CEC to IBM.

If the DASD cannot be taken OFFLINE, schedule a maintenance window.

## ARE COUPLING FACILITY CHPIDS ANY DIFFERENT?

Yes, a CF CHPID is used exclusively by a coupling facility. The CHPID on the LPAR side is called a sender path and the CHPID on the CF side is called a receiver path. You can see the sender paths from the LPAR only by using the D CF command or a D M=CHP(xx) on a CF CHPID. The only way to see the receiver path is to reference diagrams that show what is connected to what. Usually CF sender path CHPID problems can be fixed by CONFIGing the CHPID off-line and on-line. The CONFIG

CHP(xx),ONLINE can take a few minutes to complete. This should be done only if another CF sender path is available and ONLINE to the same CF. Otherwise, this should be done only after a CF is 'drained' of all structures and under the supervision of OSVS.

Occasionally, it is necessary to resolve this problem from the CF using CFCC commands to the CF from the HMC. Here is an example of 'fixing' a CF CHPID after an IPL.

## RESOLVING CF CHPID CONNECTIVITY PROBLEMS

First, attempt to resolve the problem from the LPAR side.

From the LPAR with the CF connectivity problem:

- Confirm the CHPIDs in use by the CF by using the D CF command.
- The CFNAME can be found by finding the NAMED keyword.
- The CFCHPIDs can be found by finding the SENDER keyword.

```
V PATH(CFNAME,CFCHPID),OFFLINE
```

- Wait until the path comes off-line (MVS message IXL101I):

```
CF CHP(CFCHPID),OFFLINE
```

- Wait until the CHPID comes off-line (MVS messages IEE503I and IEE712I), then attempt to bring it back on-line:

```
CF CHP(CFCHPID),ONLINE
```

- It may take a few minutes to complete (goes NOT OPERATIONAL first):

```
V PATH(CFNAME,CFCHPID),ONLINE
```

If the above commands do not fix the problem repeat the sequence and bounce the CF side while the LPAR CHPID is OFFLINE:

```
V PATH(CFNAME,CFCHPID),OFFLINE
CF CHP(CFCHPID),OFFLINE
```

Go to the HMC and bounce the CF RECEIVER PATH on the CF using CFCC commands:

1 Log on to the HMC.

2 Drill into the IPL work area for the correct data centre.

3 Highlight the CF.

4 Double-click on *Operating System Messages* in the *Daily* pane.

5 Click the *Send Command* button.

6 CONFIGURE cfchpid OFFLINE and press *Enter* (use CF CHPID).

7 Wait until it comes off-line (CF message CF0149I).

8 CONFIGURE cfchpid ONLINE and press *Enter*.

9 You may receive an error message (CF0264I Link Failed – CHPID cfchpid).

10 Confirm that the CHPID is on-line with the DISPLAY CHPID ALL command.

11 You should see the CHPID listed in the CF0106I message.

Then return to the LPAR and bring the CF CHPID back ONLINE:

```
CF CHP(CFCHPID),ONLINE
V PATH(CFNAME,CFCHPID),ONLINE
```

Contact hardware support if this does not fix the problem.

## ARE CTC CHPIDS ANY DIFFERENT?

Yes, CTCs are owned by VTAM for Channel Adapters (Cross Domain CTCs) and MPC+ channels (more common in APPN CP to CP connections). If, after attempting to resolve connectivity problems through all the normal VTAM commands, the CTCs still will not connect, try using a D M=DEV command and ESCON manager to confirm that everything is mapped correctly in the IOCDS and cabled correctly though the ESCON directors.

If everything is mapped correctly you will see the CECs, CHPIDs, and PORTs matching up in displays from each system. The PORT NAME is made up of the CEC name and the CHPID ID, and the PORTs on each side should point to each other and the TYPEs should be CTC_S on one side and CNC_S on the other.

In this example, LPAR1 (SYS1) on CEC06 has a CTC (0FAE) to LPAR4 (SYS4) on CEC03. The CHPID on the SYS1 side is EF and the CHPID on the SYS4 side is D9. The ESCON director port patched to SYS1 is AC and the port patched to SYS4 is 95.

```
ROUTE SYS1,D M=DEV(ØFAE)
IEE174I 16.56.5Ø DISPLAY M 877
DEVICE ØFAE   STATUS=ONLINE
CHP                   EF
DEST LINK ADDRESS     95
ENTRY LINK ADDRESS    AC
PATH ONLINE           Y
CHP PHYSICALLY ONLINE Y
PATH OPERATIONAL      Y
DESTINATION CU LOGICAL ADDRESS = Ø4
CU ND      = NOT AVAILABLE
DEVICE NED = ØØ2Ø64.CTC.IBM.Ø2.9542AØØ1Ø93D

ROUTE SYS1,F IHVPROC,D D ØFAE *

IHVC999I ESCON MANAGER DISPLAY 233
IHVC824I                                PORT
IHVC825I          CHP   SWCH           STATUS
IHVC826I DEVN CHP TYPE  DEVN LSN PORT H S C  P PORT NAME
IHVC827I ØFAE EF  CTC_S 9Ø1Ø 17   AC         P CECØ6.CHPEF.CTC/CNC
IHVC82AI CNTL UNIT DATA:9Ø1Ø 17   95         P CECØ3.CHPD9.CTC/CNC
IHVOØØØI I/O-OPS IS READY TO PROCESS OPERATOR COMMANDS

ROUTE SYS4,D M=DEV(ØFAE)
IEE174I 16.57.22 DISPLAY M 154
DEVICE ØFAE   STATUS=ONLINE
CHP                   D9
DEST LINK ADDRESS     AC
ENTRY LINK ADDRESS    95
PATH ONLINE           Y
CHP PHYSICALLY ONLINE Y
PATH OPERATIONAL      Y
DESTINATION CU LOGICAL ADDRESS = Ø2
CU ND      = NOT AVAILABLE
DEVICE NED = ØØ2Ø64.CTC.IBM.Ø2.9542BØØ1Ø93D

ROUTE SYS4,F IHVPROC,D D ØFAE *
```

```
IHVC999I ESCON MANAGER DISPLAY 746
IHVC824I                                PORT
IHVC825I          CHP   SWCH           STATUS
IHVC826I DEVN CHP TYPE  DEVN LSN PORT H S C  P PORT NAME
IHVC827I ØFAE D9   CNC_S 9Ø1Ø 17   95         P CECØ3.CHPD9.CTC/CNC
IHVC82AI CNTL UNIT DATA:9Ø1Ø 17   AC         P CECØ6.CHPEF.CTC/CNC
IHVOØØØI I/O-OPS IS READY TO PROCESS OPERATOR COMMANDS
```

If you can't match things up like this, there is a cabling or IOCDS problem.

*Robert Zenuk*
*Systems Programmer (USA)*

## Sending e-mail attachments from a mainframe

Included is a piece of JCL and control cards for distributing a report generated on a mainframe via e-mail. The report is contained as a text attachment in the e-mail. This ensures that the report does not clutter the mail by inline inclusions. The e-mail is sent out using an SMTP (Simple Mail Transfer Protocol) server running on the mainframe.

There are two inputs that are required to be sent to the mail server: first, a set of control cards containing the SMTP commands. This is identified by the DD name IFILE1 in the example. The second input is the report itself, which needs to be attached to the e-mail. This is identified by the DD name IFILE2 in the example.

The SMTP commands contain MIME (Multimedia Internet Mail Extension) extensions to build the report as a text attachment.

The MVS utility ICETOOL is used to copy the two input files to the mail server. This utility has the ability to take in two input files with different DCBs (record/block lengths) and write them to a common output file. The common output file is written to the input reader of the SMTP server.

The report can have a maximum record length of 240 characters; anything longer would need control cards.

In the example below, the angle brackets <> must be included as is. Replace *SMTPSERV* with the name of the MVS machine on which your SMTP server is running. Also, replace *MYREPORT* with the actual dataset containing the report to be attached.

This example can be extended to include multiple report files as well as binary files. It is just a matter of including the right MIME extension commands in the SMTP control cards. For more details on SMTP and MIME commands, refer to RFC 0821 (SMTP) and RFC 2045 (MIME). These RFCs (Request For Comments documents define the Internet standards) are available at various sites on the Internet.

```
//PSTYØØ1Ø EXEC PGM=ICETOOL
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//IFILE1   DD *
HELO SMTPSERV
MAIL FROM: <Source e-mail address>
RCPT TO: <Dest e-mail address 1>
RCPT TO: <Dest e-mail address 2>
DATA
From: <Source e-mail address>
To: <Dest e-mail address 1>,
      <Dest e-mail address 2>
Subject: Test Mail
MIME-Version: 1.Ø
Content-Type: multipart/mixed; boundary="simple boundary"

—simple boundary
Content-Type: Text/Plain

Attached is a test report.

—simple boundary
Content-Type: Text/Plain

/*
//IFILE2   DD DSN=MYREPORT,,DISP=SHR
//OFILE1   DD SYSOUT=(B,SMTP),DEST=SMTPSERV
//DFSMSG   DD SYSOUT=*
//TOOLMSG  DD SYSOUT=*
//TOOLIN   DD *
 COPY FROM(IFILE1) TO(OFILE1) USING(MAIL)
 COPY FROM(IFILE2) TO(OFILE1)
/*
//MAILCNTL DD *
 OUTREC FIELDS=(1:1,8Ø,81:16ØX)
/*
```

*S Prasad Ganti*
*Consultant*
*Citibank (USA)*

## Disaster recovery procedure

Recently we had to review our disaster recovery procedure. Previously all the back-ups were done with DFDSS on 3490 cartridges. Now that we have 3590 Magstar devices, we can save more DASD volumes on the same cartridge. The goal of this procedure is to check that all our DASD have a back-up, except some which are used for test data or volumes without data or volumes with page datasets or JES spool.

For this we do an IDCAMS DCOLLECT, which we sort into two files. The first is sorted by volume name and the second by device number. Afterwards we run a REXX procedure that shows us our configuration from the two dcollect reports.

First we produce a report with the DASD volume names and their device number.

Then we use the catalog search interface to get all the files having a dataset name mask BKUP.*.G* because we do our full dump DASD on GDGs. The first qualifier is BKUP, the second is the DASD's name, and the third is the generation number.

We print the list of files found using our search criteria, with the cartridge volume name and creation date. We print a report of back-ups that seem to be too old, older than a number of days specified as a parameter – 21 days is the default.

We create IDCAMS define commands to catalog the non-VSAM back-up dataset names, so it’s easier to retrieve them on our restore system. Then we do a matching between the DASD volume and the corresponding back-up dataset name – the second qualifier of the dataset must match the volume.

We can do a match on the last version or on a previous version.

Now we may get a list of DASD volumes without back-up. We exclude some, based on our standards, such as some starting with TEST** TT**** RV****.

Now we sort our list of back-ups that we’ve selected and matched. We do this on the cartridge name and file number (file sequence number on cartridge).

We also create some IEBUPDTE statements to add this report later in a PDS as documentation.

We could also create DFDSS dump commands to back-up the volumes without back-up, but this function is described later.

Next we produce JCL to restore the DASD volumes, using 3590 cartridge and file sequence number. We create a member by 3590 cartridge in a PDS, and there is also an alias for each member. This alias is the DASD name of the first dataset name on the 3590 cartridge.

In the case of a DASD volume on two 3590 cartridges, it goes with the first volume and the next DASD goes to a second cartridge member.

We produce a report with the DASD volumes by device number and a report with the gap between device numbers for which no volumes have been found.

Then we produce a JCL with ICKDSF to initialize the DASD volumes as required by our DRP supplier. We also have some volumes that we do not restore, but we need them to do our work.

To restore our production system, we’re using a mini OS/390 system. All the JCL produced here is saved in a library on this mini OS/390. This is one volume that we back-up each week.

We have written a small procedure to eject this last back-up out of the library.

All this has been done without a tape manager.

After saving our mini-system we do a logical full dump of our master, user, and OAM catalogs on the same cartridge as our mini system.

Our JCL:

```
//JOBDRPØØ JOB  ,CLASS=T,MSGCLASS=X,MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID
//STEPØØØ  EXEC PGM=IEFBR14,REGION=2M
//DRPCNTL  DD   DSN=SYS1.DRPOSXX.CNTL,
//         SPACE=(TRK,Ø),DISP=(MOD,DELETE)
//*
//STEPØ1Ø  EXEC PGM=IEFBR14,REGION=4M
//DRPCNTL  DD   DSN=SYS1.DRPOSXX.CNTL,DISP=(NEW,CATLG),
//         DSORG=PO,DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=Ø),
//         SPACE=(TRK,(15,5,15)),UNIT=339Ø,VOL=SER=SOSXXX
//*
//STEPØ2Ø  EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD   SYSOUT=*
//OUTDS    DD   DSN=&&DCOLLECT,DISP=(NEW,PASS),
//         DSORG=PS,DCB=(RECFM=VB,LRECL=644,BLKSIZE=Ø),
//         SPACE=(TRK,(15,5),RLSE),UNIT=VIO
//SYSIN    DD   *
  DCOLLECT  OFILE(OUTDS)  VOLUME(*)  NODATAINFO
/*
//STEPØ3Ø  EXEC PGM=SORT,COND=(Ø,LT)
//SYSOUT   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SORTIN   DD   DSN=&&DCOLLECT,DISP=(OLD,PASS)
//SORTOUT  DD   DSN=&&DCOLSORT,DISP=(NEW,PASS),
//         SPACE=(TRK,(15,5),RLSE),UNIT=VIO
//SYSIN   DD *
  RECORD TYPE=V,LENGTH=644
  SORT FIELDS=(29,Ø6,CH,A)
  SUM FIELDS=NONE
//*
//STEPØ4Ø  EXEC PGM=SORT,COND=(Ø,LT)
//SYSOUT   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SORTIN   DD   DSN=&&DCOLLECT,DISP=(OLD,PASS)
//SORTOUT  DD   DSN=&&DCOLSORD,DISP=(NEW,PASS),
//         SPACE=(TRK,(15,5),RLSE),UNIT=VIO
//SYSIN   DD *
  RECORD TYPE=V,LENGTH=644
  SORT FIELDS=(81,Ø2,CH,A)
  SUM FIELDS=NONE
//*
//STEPØ5Ø  EXEC PGM=IKJEFT1B,DYNAMNBR=2Ø,REGION=6M
//SYSEXEC  DD   DISP=SHR,DSN=your.rexx.exec
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//DASDVD   DD   SYSOUT=*
//DASDDV   DD   DSN=&&DASDDV,DISP=(,PASS),
//         DCB=(LRECL=8Ø,RECFM=FB,DSORG=PS),
//         UNIT=VIO,SPACE=(TRK,(5,5))
//DASDBK   DD   DSN=&&DASDBK,DISP=(,PASS),
//         DCB=(LRECL=8Ø,RECFM=FB,DSORG=PS),
//         UNIT=VIO,SPACE=(TRK,(5,5))
//DASDDB   DD   DSN=&&DEFNVSAM,DISP=(,PASS),
//         DCB=(LRECL=8Ø,RECFM=FB,DSORG=PS),
//         UNIT=VIO,SPACE=(TRK,(5,5))
//DASDIN   DD   DSN=&&DASDINIT,DISP=(,PASS),
//         DCB=(LRECL=8Ø,RECFM=FB,DSORG=PS),
//         UNIT=VIO,SPACE=(TRK,(5,5))
//DASDFD   DD   SYSOUT=*
//DASDRS   DD   DSN=&&DASDREST,DISP=(,PASS),
//         DCB=(LRECL=8Ø,RECFM=FB,DSORG=PS),
//         UNIT=VIO,SPACE=(TRK,(5,5))
//DCOLIN   DD   DSN=&&DCOLSORT,DISP=(OLD,PASS)
//DCOLDN   DD   DSN=&&DCOLSORD,DISP=(OLD,PASS)
//SYSTSIN  DD   *
  DRPXVOLØ BKUP.*.G*   7 21 99
//*
//STEPØ6Ø  EXEC PGM=IEBUPDTE,REGION=4M,PARM='MOD'
//SYSUT1   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSUT2   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSPRINT DD   DUMMY    SYSOUT=*
//SYSIN    DD   DSN=&&DASDREST,DISP=(OLD,PASS)
//*
//STEPØ7Ø  EXEC PGM=IEBUPDTE,REGION=4M,PARM='MOD'
//SYSUT1   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSUT2   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSPRINT DD   DUMMY    SYSOUT=*
//SYSIN    DD   DSN=&&DASDINIT,DISP=(OLD,PASS)
//*
//STEPØ8Ø  EXEC PGM=IEBUPDTE,REGION=4M,PARM='MOD'
//SYSUT1   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSUT2   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSPRINT DD   DUMMY    SYSOUT=*
//SYSIN    DD   DSN=&&DEFNVSAM,DISP=(OLD,PASS)
//*
//STEPØ9Ø  EXEC PGM=IEBUPDTE,REGION=4M,PARM='MOD'
//SYSUT1   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSUT2   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
```

```
//SYSPRINT DD   DUMMY    SYSOUT=*
//SYSIN    DD   DSN=&&DASDDV,DISP=(OLD,PASS)
//*
//STEP1ØØ  EXEC PGM=IEBUPDTE,REGION=4M,PARM='MOD'
//SYSUT1   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSUT2   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSPRINT DD   DUMMY    SYSOUT=*
//SYSIN    DD   DSN=&&DASDBK,DISP=(OLD,PASS)
//*
//STEP2ØØ  EXEC PGM=IKJEFT1B,DYNAMNBR=2Ø,REGION=6M
//SYSEXEC  DD   DISP=SHR,DSN=your.rexx.exec
//SYSTSPRT DD   SYSOUT=*
//DUMPCAT  DD   SYSOUT=*
//DUMPCAT  DD   DSN=&&DUMPCT,DISP=(,PASS),
//         DCB=(LRECL=8Ø,RECFM=FB,DSORG=PS),
//         UNIT=VIO,SPACE=(TRK,(5,5))
//SYSTSIN  DD   *
  DRPXLCAT
//*
//STEP3ØØ  EXEC PGM=ADRDSSU,REGION=6M
//SYSPRINT DD   SYSOUT=*
//DASD     DD   UNIT=339Ø,VOL=SER=SOSXXX,DISP=SHR
//TAPE     DD   DSN=BKUP.SOSXXX(+1),DISP=(,CATLG,DELETE),
//         UNIT=MAG,VOL=(,,,3Ø),
//         DCB=(DSCB,LRECL=32756,BLKSIZE=3276Ø,RECFM=VB,TRTCH=COMP),
//         LABEL=EXPDT=99ØØØ
//SYSIN    DD   *
  DUMP  FULL  INDDNAME(DASD)  OUTDDNAME(TAPE)  ADMIN CANCELERROR
//*
//STEP31Ø  EXEC PGM=ADRDSSU,REGION=6M
//SYSPRINT DD   SYSOUT=*
//TAPE     DD DSN=BKUP.CAT$$$(+1),DISP=(,CATLG,DELETE),
//         UNIT=(MAG,,DEFER),VOL=(,RETAIN,,99,REF=*.STEP3ØØ.TAPE),
//         DCB=(DSCB,LRECL=32756,BLKSIZE=3276Ø,RECFM=VB,TRTCH=COMP),
//         LABEL=(2,SL),EXPDT=99ØØØ
//SYSIN    DD   DSN=&&DUMPCT,DISP=(OLD,PASS)
//*
//STEP39Ø  EXEC PGM=IKJEFT1B,DYNAMNBR=2Ø,REGION=6M
//SYSEXEC  DD   DISP=SHR,DSN=your.rexx.exec
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  DRPXEJEØ BKUP.SOSXXX.G*
//
```

### REXX PROC to create our restore JCL:

```
/* REXX                                                               */
/*--------------------------------------------------------------------*/
/* Proc drpxvolØ                                                      */
/*   Input : BKPA   -> generic mask for backup datasets               */
/*           NBRD   -> number of days for last backup           7     */
/*           NBRL   -> number of days for oldest backup        21     */
/*           BLM    -> backup limit number Ø1 oldest 99 last   99     */
/*   Output : report with Volume name & device number.                */
/*            report with backup informations.                        */
/*            report with dasd volume without backup.                 */
/*-----------------------------------------------------------------*/
 ARG BKPA NBRD NBRL BLM
 TRACE o;
 CALL PROC_PARM;
 CALL PROC_DCOL;
 CALL PROC_NONVSFL;
 CALL PROC_PRTBKPL;
 CALL PROC_DFNVBKP;
 CALL PROC_CHECKBKP;
 CALL PROC_SORTBKP ;
 CALL PROC_DUMPDASD;
 CALL PROC_RESTDASD;
 CALL PROC_SORTDVNO;
 CALL PROC_INITDASD;
 "EXECIO Ø DISKW DASDBK (FINIS";
 "EXECIO Ø DISKW DASDDV (FINIS";
 "EXECIO Ø DISKW DASDVD (FINIS";
 "EXECIO Ø DISKW DASDIN (FINIS";
RETURN;
 /*******************************************************/
 /* Proc parm                                           */
 /*******************************************************/
PROC_parm:
 IF BKPA = "" THEN HQNVS = "BKUP.*.G*";
              ELSE HQNVS = BKPA;
 IF NBRD = "" THEN NBRD = 7;
 IF NBRL = "" THEN NBRL = 21;
 IF BLM  = "" THEN BLM  = 99;
 DATE_WKJ = DATE('J');
 DATE_WKS = DATE('S');
 YEAR_BKUP = SUBSTR(DATE_WKS,1,4);
 DAY_BKUP = SUBSTR(DATE_WKJ,3,3) - NBRD;
 IF DAY_BKUP < Ø
 THEN DO;
   DAY_BKUP = 365 - DAY_BKUP;
   YEAR_BKUP = YEAR_BKUP - 1;
 END;
 DAY_BKUP = RIGHT(DAY_BKUP,3,"Ø");
 BKUPD = YEAR_BKUP || DAY_BKUP;
 YEAR_BKUP = SUBSTR(DATE_WKS,1,4);
 DLM_BKUP = SUBSTR(DATE_WKJ,3,3) - NBRL;
 IF DLM_BKUP < Ø
 THEN DO;
   DLM_BKUP = 365 - DLM_BKUP;
   YEAR_BKUP = YEAR_BKUP - 1;
 END;
 Dlm_BKUP = RIGHT(Dlm_BKUP,3,"Ø");
 BKUPl = YEAR_BKUP || Dlm_BKUP;
 say " ********************************************* ";
 SAY "   nbrd : " nbrd ;
 SAY "   date : " date_wks;
 SAY "   date : " date_wkj;
 SAY "   MinD : " DAY_BKUP;
 SAY "   MaxD : " DLM_BKUP;
 SAY "   last : " BKUPD;
 SAY "   old  : " BKUPl;
 SAY "   BkVer: " BLM;
 say " ********************************************* ";
RETURN;
 /*********************************************************/
 /* Proc read Dcollect print report volume device number */
 /*********************************************************/
PROC_DCOL:
 "EXECIO * DISKR DCOLIN (STEM DCOLV.";
 VI = Ø;
 DO WHILE VI < DCOLV.Ø;
   VI = VI + 1;
   TV.VI = SUBSTR(DCOLV.VI,25,6);
   TD.VI = C2X(SUBSTR(DCOLV.VI,77,2));
   TS.VI = SUBSTR(DCOLV.VI,83,8);
   tb.vi = "?";
 END;
 K = 1;
 R.1 = "  " COPIES("*",71);
 R.2 = "   ** DASD VOLUME WITH DEVICE NUMBER" COPIES(" ",34) "**";
 R.3 = "  " COPIES("*",71);
 "EXECIO 3 DISKW DASDVD (STEM R.";
 DO WHILE K <= VI;
   DASD_O = "     ";
   DO J = 1 TO 5 WHILE K <= VI;
     DASD_O = DASD_O || TV.K || " " || TD.K || "   ";
     K = K + 1;
   END;
   RECO.1 = DASD_O ;
   "EXECIO 1 DISKW DASDVD (STEM RECO.";
 END;
 R.1 = "  " COPIES("*",71);
 "EXECIO 1 DISKW DASDVD (STEM R.";
RETURN;
 /*********************************************************/
 /* Proc NonVS_fl                                         */
 /*********************************************************/
PROC_NONVSFL:
 KEY = HQNVS || '.**';
```

```
COUNT = Ø                           /* TOTAL ENTIRES FOUND           */
MODRSNRC = SUBSTR(' ',1,4)          /*   CLEAR MODULE/RETURN/REASON  */
CSIFILTK = SUBSTR(KEY,1,44)         /*   MOVE FILTER KEY INTO LIST   */
CSICATNM = SUBSTR(' ',1,44)         /*   CLEAR CATALOG NAME          */
CSIRESNM = SUBSTR(' ',1,44)         /*   CLEAR RESUME NAME           */
CSIDTYPS = SUBSTR('ABH',1,16)       /*   CLEAR ENTRY TYPES           */
CSICLDI  = SUBSTR('Y',1,1)          /*   INDICATE DATA AND INDEX     */
CSIRESUM = SUBSTR(' ',1,1)          /*   CLEAR RESUME FLAG           */
CSIS1CAT = SUBSTR(' ',1,1)          /* SEARCH > 1 CATALOGS           */
CSIRESRV = SUBSTR(' ',1,1)          /*   CLEAR RESERVE CHARACTER     */
CSINUMEN = 'ØØØ5'X                  /*   INIT NUMBER OF FIELDS       */
CSIFLD1    = SUBSTR('VOLSER',1,8)   /*   INIT FIELD 1 FOR VOLSERS    */
CSIFLD2    = SUBSTR('DEVTYP',1,8)   /*   INIT FIELD 2 FOR DEVTYP     */
CSIFLD3    = SUBSTR('FILESEQ',1,8)  /*   INIT FIELD 5 FOR DS EX DT   */
CSIFLD4    = SUBSTR('DSCRDT2',1,8)  /*   INIT FIELD 3 FOR DS CR DT   */
CSIFLD5    = SUBSTR('DSEXDT2',1,8)  /*   INIT FIELD 4 FOR DS EX DT   */
 /********************************************************************/
 /*  BUILD THE SELECTION CRITERIA FIELDS PART OF PARAMETER LIST      */
 /********************************************************************/
CSIOPTS  = CSICLDI || CSIRESUM || CSIS1CAT || CSIRESRV
CSIFIELD = CSIFILTK || CSICATNM || CSIRESNM || CSIDTYPS || CSIOPTS
CSIFIELD = CSIFIELD || CSINUMEN || CSIFLD1 || CSIFLD2 || CSIFLD3
CSIFIELD = CSIFIELD || CSIFLD4 || CSIFLD5;
 /********************************************************************/
 /*  INITIALIZE AND BUILD WORK ARE OUTPUT PART OF PARAMETER LIST     */
 /********************************************************************/
WORKLEN = 131Ø72  /* 128K */
DWORK = 'ØØØ2ØØØØ'X || COPIES('ØØ'X,WORKLEN-4)
 /********************************************************************/
 /*  INITIALIZE WORK VARIABLES                                       */
 /********************************************************************/
RESUME = 'Y'
CATNAMET = SUBSTR(' ',1,44)
DNAMET = SUBSTR(' ',1,44)
IC = Ø;
 /********************************************************************/
 /*  SET UP LOOP FOR RESUME (IF A RESUME IS NCESSARY)                */
 /********************************************************************/
DO WHILE RESUME = 'Y'
 /********************************************************************/
 /*  ISSUE LINK TO CATALOG GENERIC FILTER INTERFACE                  */
 /********************************************************************/
  ADDRESS LINKPGM 'IGGCSIØØ  MODRSNRC  CSIFIELD  DWORK'
  RESUME = SUBSTR(CSIFIELD,15Ø,1);
  USEDLEN = C2D(SUBSTR(DWORK,9,4));
  POS1=15;
 /********************************************************************/
 /*  PROCESS DATA RETURNED IN WORK AREA                              */
 /********************************************************************/
  DO WHILE POS1 < USEDLEN
```

```
    IF SUBSTR(DWORK,POS1+1,1) = 'Ø'
    THEN DO
      CATNAME=SUBSTR(DWORK,POS1+2,44)
      POS1 = POS1 + 5Ø
    END
    IF POS1 < USEDLEN     /* IF STILL MORE DATA       */
    then DO               /* CONTINUE WITH NEXT ENTRY */
      DNAME = SUBSTR(DWORK,POS1+2,44)  /* GET ENTRY NAME   */
```

*Editor's note: this article will be concluded next month.*

*Alain Piraux*
*System Engineer (Belgium)*

# New Z990 channel subsystem

Announced on 13 May 2003, the new Z990 IBM server, code named T-Rex, provides new levels of scalability, including:

- Up to 9,000 MIPS on a 32-processor configuration.
- Up to 30 logical partitions (LPARs) will be supported.
- Up to 256GB of memory.
- Up to 512 channels using the new Logical Channel SubSystems (LCSS) concept.

The Z990 machine type is 2084.

The Z990 can be purchased in four models – A08, B16, C24, and D32:

- Models A08 (up to 8 processors) and B16 (up to 16 processors) have been available since 16 June 2003 (GA1).
- Models C24 (up to 24 processors) and D32 model (up to 32 processors) have been available since 31 October 2003 (GA2).

The Z990 processor is especially designed to allow customer consolidation of workloads and OS images. This consolidation objective has been until now limited by architecture constraints:

- A maximum of 15 LPARs per processor.
- A maximum of 256 channels.

Architectural enhancements of the Z990 server require a new approach to I/O configuration management. This article will focus on the channel subsystem changes introduced by the Z990 to support up to 30 LPARs and up to 512 channels.

## Z990 CHANNEL SUBSYSTEM CONCEPTS

### History

Every IBM system since 370/XA has been limited to a maximum of 256 channels by the architecture. Without FICON (and in a few cases even with it – a z900 has a maximum of 96 FICON channels) this was a serious constraint for very large installations.

The new Z990 breaks this 256-channel limit.

### Logical Channel SubSystem (LCSS)

The Z990 I/O infrastructure has been redesigned to handle a large increase in I/O system performance.

The Z990 provides the ability to define more than 256 CHPIDs because of the introduction of the Logical Channel SubSystem concept.

An LCSS is a logical replication of the channel subsystem used on older S/390 systems. Two hundred and fifty six CHPIDs can be defined within an LCSS with a range of from 00 to FF.

Up to two LCSSs are supported on the Z990 – LCSS 0 and LCSS 1.

### Logical partitions are now defined to one LCSS

Logical partitions are now defined to an LCSS, and not to a processor any more. An LCSS can be configured with 1 to 15 logical partitions.

So, a Z990 configured with two LCSSs can handle up to 512 CHPIDs and up to 30 logical partitions (function available since 31 October 2003).

Multiple Image Facility (MIF) enables resource sharing across logical partitions within a single LCSS or across the LCSSs.

The MIF Image ID (MIF ID) is a number in the range 1 to F that identifies a logical partition within an LCSS.

The logical partition identifier (LPAR ID) is a number in the range from 00 to 3F. It is assigned by the user on the image profile through the Hardware Management Console (HMC). The LPAR ID is unique across the Z990.

MIF ID is not unique within the Z990 processor: logical partitions in different LCSSs can have the same MIF ID.

Partition names must be unique within the Z990 complex.

## Physical Channel ID (PCHID) concept

On a Z990, a CHPID does not directly correspond to a hardware channel port.

A Physical Channel ID, or PCHID, reflects the physical identifier of a channel-type interface.

A PCHID number is based on the I/O cage location, the channel feature slot number, and the port number of the channel feature – see Figure 1.

CHPIDs are not pre-assigned to a PCHID: it is the responsibility of the user to assign the CHPID numbers through the use of the CHPID Mapping Tool (CMT) or HCD/IOCP.

Assigning CHPIDs means that the CHPID number is associated with a physical channel port location (PCHID) and an LCSS.

The CHPID number range is still from 00 to FF and must be unique within an LCSS.

Any CHPID not connected to a PCHID will fail validation when an attempt is made to build a production IODF or an IOCDS.

### Channel spanning concept

Channel spanning extends the MIF concept of sharing channels across logical partitions to sharing channels across logical partitions and LCSSs. Spanning is the ability of the channel to be configured to multiple LCSSs.

| *Cage* | *Front PCHID numbers* | *Rear PCHID numbers* |
|---|---|---|
| CEC cage | 000-0FF | - |
| 1st I/O cage | 100-1FF | 200-2FF |
| 2nd I/O cage | 300-3FF | 400-4FF |
| 3rd I/O cage | 500-5FF | 600-6FF |

*Figure 1: PCHID number*

When defined that way, the channels can be transparently shared by any or all of the configured logical partitions, regardless of the LCSS to which the logical partition is configured.

A channel is considered a spanned channel if the same CHPID number in different LCSSs is assigned to the same PCHID in the IOCP, or is defined as ‘spanned’ in the HCD.

CHPIDs that span LCSSs reduce the total number of channels available on the Z990. This total is reduced since no LCSS can have more than 256 CHPIDs.

For a Z990 with two LCSSs, a total of 512 CHPIDs are supported. If all CHPIDs are spanned across the two LCSSs, then only 256 channels can be supported.

Spanning was introduced on 31 October 2003 for IC links and HiperSockets.

## Z990 SOFTWARE SUPPORT

Software support for the Z990 comes in two stages:

- Compatibility support – provides no additional functionality over and above a z900 or z800. Compatibility support provides only PTFs that allow the operating system to run on a Z990.
- Exploitation support – provides the operating systems with the ability to take advantage of greater than 15 logical partitions and multiple LCSSs.

### Compatibility support

Compatibility support has been available since 16 June 2003 (it can be downloaded from http://www-1.ibm.com/servers/eserver/zseries/zos/downloads/).

The following functions are available:

- Models A08, B16
- 128GB memory
- Two LCSSs
- Fifteen defined partitions
- Two-digit LPAR ID.

### Exploitation support

Exploitation support  has been available since 31 October 2003 and it delivers the following functions:

- Models C24, D32
- 256GB memory
- Spanned internal channel
- Dynamic I/O support for LCSS 1
- Thirty defined partitions.

### IBM ‘statement of direction’

As a ‘statement of direction’ (SOD), IBM announced that:

- “Up to four Logical Channel Subsystems, with up to 1024 CHPIDs and up to 60 logical partitions” will be supported in the future.
- “More than 16 processors will be supported in a single LPAR image” with Z/OS 1.6.

## HCD DEFINITIONS

I/O configuration definition support for the Z990 is one of the most important activities that a Z990 customer will face when preparing for a Z990 install.

In order to define and configure a Z990 processor, you should first install HCD 1.4, which is included in the compatibility support package downloaded from the Internet.

I will describe, step-by-step, the operations required to define the following Z990 LCSSs.

### Defining the Z990 processor

The first thing to do is to define the new Z990 processor using Option 3 (Processors).

The processor type of a Z990 is 2084 and in our case study we will define a B16 model with two LCSSs:

```
  Goto  Filter  Backup  Query  Help
.-------------------------- Add Processor --------------------------.
|                                                                   |
|                                                                   |
| Specify or revise the following values.                           |
|                                                                   |
| Processor ID . . . . . . . . . Z99Ø                               |
|                                                                   |
| Processor type . . . . . . . . 2Ø84      +                        |
| Processor model  . . . . . . . B16       +                        |
| Configuration mode . . . . . . LPAR      +                        |
| Number of channel subsystems . . 2       +     $ LCSSØ and LCSS1  |
|                                                                   |
| Serial number  . . . . . . . . __________                         |
| Description  . . . . . . . . . Sample Z99Ø configuration______    |
|                                                                   |
| Specify SNA address only if part of an S/39Ø microprocessor cluster:|
|                                                                     |
| Network name . . . . . . . . . . ________  +                        |
| CPC name . . . . . . . . . . . . ________  +                        |
|                                                                     |
|  F1=Help   F2=Split   F3=Exit     F4=Prompt   F5=Reset   F9=Swap    |
| F12=Cancel                                                          |
'---------------------------------------------------------------------'
```

## Defining an LCSS

Once the processor is defined, the next step is to configure the two LCSSs.

When you select the new processor:

```
                          Processor List        Row 1 of 1 More:       >
Command ===> __________________________________________ Scroll ===> HALF

Select one or more processors, then press Enter. To add, use F11.


/ Proc. ID Type +    Model +   Mode+ Serial-# + Description
S Z99Ø     2Ø84      B16       LPAR  __________ Sample Z99Ø configuration
*************************** Bottom of data *****************************
```

you get the new *Channel Subsystem List* panel:

```
                         Channel Subsystem List                Row 1 of 2
 Command ===> _________________________________________ Scroll ===> HALF

Select one or more channel subsystems, then press Enter. To add, use F11

 Processor ID . . . : Z99Ø         Sample Z99Ø configuration

   CSS Max number
 / ID  of devices +  Description
 _ Ø   64512         ________________________________
 _ 1   64512         ________________________________
 *************************** Bottom of data ****************************
```

On this panel, you can specify the maximum number of devices that can be defined for this channel subsystem.

The MAXDEV parameter replaces the *Dynamic I/O expansion* setting of the HMC RESET profile. It has a direct impact on the HSA size and its maximum value is 64,512.

There is no HSA expansion support for dynamic I/O on the Z990 Support Element.

The HSA allocation is controlled by the *maximum number of devices* field on the HCD *Channel Subsystem List* panel. This value can be changed only by a power-on reset.

## Defining logical partitions

When LCSSs are defined, you can now define logical partitions.

From the *Channel Subsystem List* panel, you can work with partitions using option P:

```
                        Channel Subsystem List                 Row 1 of 2
Command ===> __________________________________________ Scroll ===> HALF

Select one or more channel subsystems, then press Enter. To add, use F11

Processor ID . . . : Z99Ø         Sample Z99Ø configuration

  CSS Max number
/ ID  of devices +  Description
p Ø   64512         ___________________________  working with partitions
_ 1   64512         ___________________________
************************** Bottom of data ******************************
```

Then you get the *Partition List* panel:

```
.--------------------------- Partition List --------------------------.
|   Goto  Backup  Query  Help                                         |
| ------------------------------------------------------------------- |
|                                                                     |
| Command ===> ____________________________________ Scroll ===> HALF  |
|                                                                     |
| Select one or more partitions, then press Enter. To add, use F11.   |
|                                                                     |
| Processor ID  . . . . : Z99Ø      Sample Z99Ø configuration         |
| Configuration mode  . : LPAR                                        |
| Channel Subsystem ID  : Ø                              $ LCSS       |
|                                                                     |
| / Partition Name   Number Usage + Description                       |
| ************************* Bottom of data *************************** |
|                                                                     |
|  F1=Help       F2=Split      F3=Exit       F4=Prompt     F5=Reset   |
|  F7=Backward   F8=Forward    F9=Swap      F1Ø=Actions   F11=Add     |
| F12=Cancel    F13=Instruct  F22=Command                             |
'---------------------------------------------------------------------'
```

where you can hit PF11 to add partition LP1, whose MIF ID is 1:

```
.--------------------- Add Partition ---------------------.
```

```
|                                                           |
|                                                           |
| Specify the following values.                             |
|                                                           |
| Partition name . . . LP1_____                             |
| Partition number . . 1      (same as MIF image ID)   $ MIF ID (1 to F)
| Partition usage  . . OS     +                             |
|                                                           |
| Description  . . . . ________________________________     |
|                                                           |
|                                                           |
|  F1=Help    F2=Split   F3=Exit    F4=Prompt  F5=Reset     |
|  F9=Swap   F12=Cancel                                     |
'----------------------------------------------------------'
```

You can do the same thing to define LP2 and LP3 on LCSS 0:

```
.--------------------------- Partition List ---------------------------.
|   Goto  Backup  Query  Help                                          |
| ------------------------------------------------------------------  |
|                                                         Row 1 of 3 |
| Command ===> ____________________________________ Scroll ===> HALF   |
|                                                                      |
| Select one or more partitions, then press Enter. To add, use F11.   |
|                                                                      |
| Processor ID  . . . . : Z99Ø      Sample Z99Ø configuration          |
| Configuration mode  . : LPAR                                         |
| Channel Subsystem ID  : Ø                                            |
|                                                                      |
| / Partition Name   Number Usage + Description                        |
| _ LP1              1      OS      ________________________________   |
| _ LP2              3      OS      ________________________________   |
| _ LP3              5      OS      ________________________________   |
| ************************** Bottom of data ************************** |
```

And to define LP14, LP15, and LP16 on LCSS 1:

```
.--------------------------- Partition List ---------------------------.
|   Goto  Backup  Query  Help                                          |
| ------------------------------------------------------------------  |
|                                                         Row 1 of 3 |
| Command ===> ____________________________________ Scroll ===> HALF   |
|                                                                      |
| Select one or more partitions, then press Enter. To add, use F11.   |
|                                                                      |
| Processor ID  . . . . : Z99Ø      Sample Z99Ø configuration          |
| Configuration mode  . : LPAR                                         |
| Channel Subsystem ID  : 1                                            |
|                                                                      |
| / Partition Name   Number Usage + Description                        |
| _ LP14               2      OS       ________________________________   |
| _ LP15               3      OS       ________________________________   |
| _ LP16               5      OS       ________________________________   |
| ************************** Bottom of data ************************** |
```

## Defining CHPIDs

In order to define CHPIDs to LCSS, you have to select the LCSS from the *Channel Subsystem List* with option S:

```
                          Channel Subsystem List                Row 1 of 2
Command ===> __________________________________________ Scroll ===> HALF

Select one or more channel subsystems, then press Enter. To add, use F11

Processor ID . . . : Z99Ø         Sample Z99Ø configuration

  CSS Max number
/ ID  of devices +  Description
s Ø   64512         ________________________________      $ select LCSS Ø
_ 1   64512         ________________________________
**************************** Bottom of data ****************************
```

Then you get the *Channel Path List* panel:

```
                               Channel Path List
 Command ===> _________________________________________ Scroll ===> HALF

 Select one or more channel paths, then press Enter. To add use F11.

 Processor ID . . . . : Z99Ø         Sample Z99Ø configuration
 Configuration mode . : LPAR
 Channel Subsystem ID : Ø

                     DynEntry Entry +
 / CHPID Type+ Mode+ Switch + Sw Port Con Mngd Description
 *************************** Bottom of data ****************************
```

where you can hit PF11 to add CHPID 80, whose PCHID is 140:

```
.-------------------------- Add Channel Path --------------------------.
|                                                                       |
|                                                                       |
| Specify or revise the following values.                               |
|                                                                       |
| Processor ID . . . . : Z99Ø         Sample Z99Ø configuration         |
| Configuration mode . : LPAR                                           |
| Channel Subsystem ID : Ø                                              |
|                                                                       |
| Channel path ID  . . . . 8Ø    +            PCHID . . . 14Ø           |
ç specify PCHID
| Number of CHPIDs . . . . 1                                            |
| Channel path type  . . . CNC   +                                      |
| Operation mode . . . . . SHR   +                                      |
| Managed  . . . . . . . . No    (Yes or No)   I/O Cluster ________  +  |
| Description  . . . . . . ________________________________            |
|                                                                       |
| Specify the following values only if connected to a switch:          |
| Dynamic entry switch ID  __  + (ØØ - FF)                              |
| Entry switch ID  . . . . __  +                                        |
| Entry port . . . . . . . __  +                                        |
|                                                                       |
|                                                                       |
'-----------------------------------------------------------------------'
```

This channel is shared between all LCSS 0 partitions:

```
.------------------------ Define Access List -------------------------.
|                                                           Row 1 of 3 |
| Command ===> ____________________________________ Scroll ===> HALF   |
|                                                                      |
| Select one or more partitions for inclusion in the access list.      |
|                                                                      |
| Channel subsystem ID : Ø                                             |
| Channel path ID  . . : 8Ø     Channel path type  . : CNC             |
| Operation mode . . . : SHR    Number of CHPIDs . . : 1               |
|                                                                      |
| / CSS ID Partition Name   Number Usage Description                   |
| / Ø      LP1              1      OS                                  |
| / Ø      LP2              3      OS                                  |
| / Ø      LP3              5      OS                                  |
| ************************* Bottom of data ************************** |
```

You can do the same thing to define CHPIDs 81, 90, and 91 on LCSS 0:

```
                        Channel Path List       Row 1 of 4 More:       >
Command ===> __________________________________________ Scroll ===> HALF

Select one or more channel paths, then press Enter. To add use F11.

Processor ID . . . . : Z99Ø         Sample Z99Ø configuration
Configuration mode . : LPAR
Channel Subsystem ID : Ø
$ LCSS Ø

                     DynEntry Entry +
/ CHPID Type+ Mode+ Switch + Sw Port Con Mngd Description
_ 8Ø    CNC   SHR   __       __ __       No
________________________________
_ 81    CNC   SHR   __       __ __       No
```

```
________________________________
_ 9Ø     CNC    SHR    __        __ __       No
________________________________
_ 91     CNC    SHR    __        __ __       No
________________________________
************************** Bottom of data *****************************
```

And to define CHPIDS 80, 81, 90, and 91 on LCSS 1:

```
                        Channel Path List       Row 1 of 4 More:       >
Command ===> __________________________________________ Scroll ===> HALF

Select one or more channel paths, then press Enter. To add use F11.

Processor ID . . . . : Z99Ø          Sample Z99Ø configuration
Configuration mode . : LPAR
Channel Subsystem ID : 1
$ LCSS 1

                    DynEntry Entry +
/ CHPID Type+ Mode+ Switch + Sw Port Con Mngd Description
_ 8Ø     CNC    SHR    __        __ __       No
________________________________
_ 81     CNC    SHR    __        __ __       No
________________________________
_ 9Ø     CNC    SHR    __        __ __       No
________________________________
_ 91     CNC    SHR    __        __ __       No
________________________________
*************************** Bottom of data *****************************
```

## Defining the DASD control units

At this point, you can select HCD Option 4 (control units) and hit PF11 to define the DASD control units:

```
.-------------------------- Add Control Unit --------------------------.
|                                                                      |
|                                                                      |
| Specify or revise the following values.                              |
|                                                                      |
| Control unit number  . . . . ØØ1_  +                                 |
| Control unit type  . . . . . 399Ø_________   +                       |
|                                                                      |
| Serial number  . . . . . . . __________                              |
| Description  . . . . . . . . ________________________________        |
|                                                                      |
| Connected to switches  . . . __  __  __  __  __  __  __  __  +       |
| Ports  . . . . . . . . . . . __  __  __  __  __  __  __  __  +       |
|                                                                      |
| If connected to a switch:                                         |
|                                                                   |
| Define more than eight ports . . 2   1.  Yes                      |
|                                      2.  No                       |
| Propose CHPID/link addresses and                                  |
| unit addresses . . . . . . . . . 2   1.  Yes                      |
|                                      2.  No                       |
|                                                                   |
|                                                                   |
'-------------------------------------------------------------------'
```

Then you have to specify the ESCD connections on the *Processor/ CU* panel:

```
                     Select Processor / CU     Row 1 of 2 More:       >
 Command ===> _________________________________________ Scroll ===> HALF

 Select processors to change CU/processor parameters, then press Enter.

 Control unit number . . : ØØØ1     Control unit type . . . : 399Ø

          --------------Channel Path ID . Link Address + -------------
/ Proc.CSSID 1------ 2------ 3------ 4------ 5------ 6------ 7------ 8--
_ Z99Ø.Ø     8Ø.Ø1__ 81.Ø2__ _______ _______ _______ _______ _______ ___
_ Z99Ø.1     8Ø.11__ 81.12__ _______ _______ _______ _______ _______ ___
 ******************************* Bottom of data
********************************
```

You should notice that you get one *processor* line for each LCSS.

For the other control unit, you have to enter:

```
                     Select Processor / CU     Row 1 of 2 More:       >
 Command ===> _________________________________________ Scroll ===> HALF

 Select processors to change CU/processor parameters, then press Enter.

 Control unit number . . : ØØØ2     Control unit type . . . : 399Ø

          --------------Channel Path ID . Link Address + -------------
/ Proc.CSSID 1------ 2------ 3------ 4------ 5------ 6------ 7------ 8--
_ Z99Ø.Ø     9Ø.Ø1   91.Ø2   _______ _______ _______ _______ _______ ___
_ Z99Ø.1     9Ø.11   91.12   _______ _______ _______ _______ _______ ___
 *************************** Bottom of data ****************************
```

# An IPCS VERBEXIT routine for displaying NAME/TOKEN lists

Sharing data between address spaces has been a long-time requirement on MVS systems. Over the course of time, many techniques have been employed:

- Using the CVTUSER field as an anchor point for shared data.
- Using a subsystem SSCVT control block for sharing data.
- Using cross-memory services.
- Using data spaces.

This is by no means a complete list, it just offers a representative example of some of the methods that have been used through the years. In each case, the data sharers need to agree on the anchor point for access to the shared data and the format layout of shared data components.

## USING NAME/TOKEN PAIRS

Since as far back as MVS/ESA 4 (and possibly even earlier), IBM has offered NAME/TOKEN pairs as a method for sharing data between address spaces. There are some very appealing aspects of using NAME/TOKEN services:

- IBM provides a suite of service routines to manage the NAME/TOKEN pairs including services for creation, deletion, and retrieval.
- The anchor points for NAME/TOKEN tables are pre-determined.

Depending on your requirements, you can make use of three different levels of NAME/TOKEN pairs. The system-level NAME/TOKEN is useful for sharing information between many different address spaces. An address space-level NAME/TOKEN comes in two flavours – home address space and primary address space – and is useful for sharing information between programs running in either the same home or primary address space. A task-level NAME/TOKEN is useful for sharing information between different programs running in the same task.

In each case, the format of the internal table created is the same and the real advantage is that the operating system manages the table entries and any associated searches.

## THE NAMETOKN IPCS SUBCOMMAND

A good application interface also requires a diagnostic tool. For NAME/TOKEN pairs, IBM has provided the NAMETOKN subcommand for IPCS. For a given dump dataset, this subcommand can be used to provide information about a specified NAME/TOKEN. A restrictive drawback to using the subcommand is that you must know the name component of a NAME/TOKEN pair as well as the level at which it was defined (ie system, address space, or task) before you can make practical use of the NAMETOKN subcommand. In many cases, the name component of a NAME/TOKEN pair is unknown or variable in nature, which makes using the NAMETOKN subcommand unsatisfactory. Also, if you simply wanted to obtain a list of all the NAME/TOKEN pairs at a given level, the NAMETOKN subcommand is not capable of providing that information either.

## THE NMTKLST IPCS VERBEXIT ROUTINE

This article discusses an IPCS VERBEXIT routine that can be used to overcome the deficiencies of the NAMETOKN subcommand outlined above. The NMTKLST IPCS VERBEXIT provided with this article allows you to list all the NAME/TOKEN pairs at any or all of the NAME/TOKEN levels, depending on the content of the dump dataset and the keyword parameters supplied to the NMTKLST routine. In its simplest format, from IPCS Option 6 you can specify:

```
    VERBX NMTKLST
```

This invocation will list all the system-level NAME/TOKEN pairs active in the current default dump dataset.

Optional keywords that can be supplied to the NMTKLST routine include:

- ASCBADDR(ascbaddr) – indicates a specific ASCB.
- TCBADDR(tcbaddr) – indicates a specific TCB in either the dump's default address space or the address space specified in the ASCBADDR keyword.
- NOSYSLVL – disables a system-level NAME/TOKEN list display unless no address space-level or task-level NAME/TOKEN list is requested, in which case this keyword is ignored.
- NOASLVL – disables an address space-level NAME/TOKEN list display unless no task-level NAME/TOKEN list is requested, in which case this keyword is ignored.

The ASCBADDR and TCBADDR keywords are used to target specific address spaces and tasks. If you specify both the ASCBADDR keyword and the TCBADDR keyword, the task-level NAME/TOKEN list will be displayed for the TCB requested (if the TCB address is valid and a task-level NAME/TOKEN table exists). The address space-level NAME/TOKEN list will also be displayed (if one exists) unless the NOASLVL keyword is also used.

Using the TCBADDR keyword without a corresponding ASCBADDR keyword will cause the NAME/TOKEN list for the indicated TCB for the dump dataset's default address space to be listed.

Below are some example invocation formats:

```
VERBX NMTKLST 'ASCB(ascbaddr)'
VERBX NMTKLST 'ASCB(ascbaddr) NOSYSLVL'
VERBX NMTKLST 'ASCB(ascbaddr) TCB(tcbaddr)'
VERBX NMTKLST 'ASCB(ascbaddr) TCB(tcbaddr) NOSYSLVL'
VERBX NMTKLST 'ASCB(ascbaddr) TCB(tcbaddr) NOSYSLVL NOASLVL'
```

where *ascbaddr* and *tcbaddr* represent ASCB addresses and TCB addresses respectively.

The NMTKLST routine will issue appropriate messages if storage areas cannot be located in the dump or if data in the control block search chains is inconsistent with expected data (ie control block eye-catcher data is incorrect). This minimizes the chance that invalid NAME/TOKEN data will be listed.

Example output from issuing the NMTKLST VERBEXIT routine would look similar to the following:

```
NMTKN090I - Processing system-level NAME/TOKEN table
    System level
    TOKEN.... Ø21B3DF8  Ø2ØØØØ48  ØØØØØØØØ  ØØØØØØØØ
    NAME..... DSNLOGREC
    ASID..... ØØØ1
    Persistent
    Created by authorized program

    System level
    TOKEN.... Ø7912Ø38  ØØØØØØØØ  ØØØØØØØØ  ØØØØØØØØ
    NAME..... IBMJESXCFAS
    ASID..... ØØ1Ø
    Created by authorized program

    System level
    TOKEN.... Ø7BA45A8  ØØFB7EØØ  ØØØØØØØØ  ØØØØØØØØ
    NAME..... JES2_AUXECB_JES2
    ASID..... ØØ17
    Created by authorized program

    System level
    TOKEN.... ØØØØ25ØØ  ØØØØØØØØ  ØØØØØØØØ  ØØØØØØØØ
    NAME..... JES2_LX_NUM_JES2
    ASID..... ØØ17
    Persistent
    Created by authorized program

    System level
    TOKEN.... Ø6D91Ø4Ø  Ø2ØAØØØØ  ØØØØØØØØ  ØØØØØØØØ
    NAME..... ISFHSVT.SDSF
    ASID..... ØØ39
    Persistent
    Created by authorized program

    System level
    TOKEN.... ØØØØ2DØØ  ØØØØØØØØ  ØØØØØØØØ  ØØØØØØØØ
```

```
      NAME..... ISFHLX.SDSF
      ASID..... ØØ39
      Persistent
      Created by authorized program

      System level
      TOKEN.... ØØØØ2DØØ  ØØØØØØØØ  ØØØØØØØØ  ØØØØØØØØ
      NAME..... ISFQSRV.SDSF
      ASID..... ØØ39
      Created by authorized program
      System level
      TOKEN.... 738763F4  7F122ØØØ  ØØØØØ9FC  ØØØØØØØØ
      NAME..... C9E2C64B  E2C4E2C6  4Ø4Ø4Ø4Ø  738763F4
      ASID..... ØØ39
      Created by authorized program

  NMTKNØ91I - Processing address space-level NAME/TOKEN table

      Address space level
      TOKEN.... ØØØØØØØØ  ØØØ171A2  ØØØØØØØØ  ØØØ173Ø8
      NAME..... C9E2D7C6  E2E5C3E7  ØØØØØØEØ  ØØØØØ121
      ASID..... ØØ38

  NMTKNØ92I - Processing task-level NAME/TOKEN table

      Task level
      TOKEN.... ØØØØ3B4F  ØØØØØØØØ  ØØØØØØØØ  ØØØØØØØØ
      NAME..... C9D9E7E3  D6D2C5D5  ØØ8CC5AØ  Ø7C3BC9Ø
      ASID..... ØØ38  TCB@..... ØØ8CC5AØ
      Created by authorized program

      Task level
      TOKEN.... ØØØØ3B51  ØØØØØØØØ  ØØØØØØØØ  ØØØØØØØØ
      NAME..... C9D9E7E3  D6D2C5D5  ØØ8CC5AØ  Ø7C3B8AØ
      ASID..... ØØ38  TCB@..... ØØ8CC5AØ
      Created by authorized program
```

## ACTIVATING THE NMTKLST VERBEXIT EXIT

In order to make the NMTKLST VERBEXIT exit available to your IPCS session, linkedit NMTKLST into a load library that resides somewhere in the search order for your active session – the linklist or STEPLIB are two options.

Depending on which NAME/TOKEN lists you want to display, the source dump data will need to contain CSA and RGN. The system-level NAME/TOKEN table is maintained in CSA and the address space-level and task-level NAME/TOKEN tables are maintained in private area storage (RGN).

## POINTS TO NOTE

The *ascbaddr* and *tcbaddr* values mentioned earlier can, for the most part, be specified as a standard IPCS data-descriptor. There are two known exceptions. When IPCS creates ASCB symbols (ASCBnnnnn) or stack pointer entry symbols (Znnnnn), the IPCS LISTSYM subcommand lists the symbols with the trailing numeric component of the symbol name showing only the significant numbers (ie the leading zeros are stripped out) – for example, the stack pointer entry symbol for which the real symbol name would be Z00001 shows up in the LISTSYM display as Z1. These symbols have been defined within IPCS with the fully-qualified, five-digit numeric suffix. If these symbols are used for either the *ascbaddr* value or the *tcbaddr* value in the NMTKLST parameters, they must be used in their fully-qualified format, otherwise NMTKLST will terminate with a message indicating that the specified symbol could not be found.

## CONCLUSION

If you use NAME/TOKEN services, or are interested in examining NAME/TOKEN table information, the NMTKLST VERBEXIT routine should be added to your IPCS toolkit.

## NMTKLST ASSEMBLER

```
NMTKLST  CSECT
NMTKLST  AMODE 31
NMTKLST  RMODE ANY
*-------------------------------------------------------------------*
*   NMTKLST is designed to be used as an IPCS VERBX exit routine      *
*   that can be used to display the information regarding the         *
*   various NAME/TOKEN lists.                                         *
*                                                                     *
*   There are three possible NAME/TOKEN lists maintained in an        *
*   OS/39Ø or z/OS system.  These are:                                *
*   - the system-level NAME/TOKEN list                                *
*   - the address space-level NAME/TOKEN list                         *
*   - the task-level NAME/TOKEN list                                *
*                                                                   *
*   To be able to display information on the system-level           *
*   NAME/TOKEN list, be sure the dump contains CSA.  To be able to  *
*   display information for the address space-level or task-level  *
*   NAME/TOKEN list, be sure the dump contains RGN.                 *
*                                                                   *
*   The simplest format for exit invocation is as follows:          *
*     VERBX NMTKLST                                                 *
*                                                                   *
*   This will list the system-level NAME/TOKEN list.                *
*                                                                   *
*   Alternatively, you can invoke the NMTKLST verbexit routine with *
*   parameters that allow you to display the address space-level    *
*   NAME/TOKEN list and/or a task-level NAME/TOKEN list.  Here are  *
*   some example invocations:                                       *
*     VERBX NMTKLST 'ASCB(ascbaddr)'                                *
*     VERBX NMTKLST 'ASCB(ascbaddr) NOSYSLVL'                       *
*     VERBX NMTKLST 'ASCB(ascbaddr) TCB(tcbaddr)'                   *
*     VERBX NMTKLST 'ASCB(ascbaddr) TCB(tcbaddr) NOSYSLVL'          *
*     VERBX NMTKLST 'ASCB(ascbaddr) TCB(tcbaddr) NOSYSLVL NOASLVL'  *
*                                                                   *
*   where 'ascbaddr' is the address of the appropriate ASCB and     *
*   'tcbaddr' is the address of the TCB of interest.  The 'ascbaddr' *
*   and 'tcbaddr' can be any valid IPCS data descriptor.            *
*                                                                   *
*   By default, the NMTKLST routine will display higher level       *
*   NAME/TOKEN lists unless the NOSYSLVL and/or NOASLVL keywords    *
*   are detected in the optional parameters.  For example, the      *
*   first command above would display both the system-level         *
*   NAME/TOKEN list and the address space NAME/TOKEN list for the   *
*   specified address space.  The second command above would display *
*   only the address space-level NAME/TOKEN list.                   *
*                                                                   *
*   Use the NOASLVL keyword to disable the address space-level      *
*   NAME/TOKEN list display.  Use of the NOASLVL keyword has no     *
*   effect if the TCB(tcbaddr) keyword has not been specified.      *
*                                                                   *
*   Use the NOSYSLVL keyword to disable the system-level NAME/TOKEN  *
*   list display.  Use of the NOSYSLVL keyword has no effect if     *
*   neither the ASCB(ascbaddr) or TCB(tcbaddr) keyword have been    *
*   specified.                                                      *
*                                                                   *
*   If the TCB(tcbaddr) keyword is used in the absence of the       *
*   ASCB(ascbaddr) keyword, the current ASCB will be used as the    *
*   default source address space.                                   *
*                                                                   *
*   The parameters are all keyword parameters.  The order in which  *
*   they are specified is of no significance.                       *
*                                                                   *
*   IPCS symbols will be created for each NAME/TOKEN list detected.   *
*   For example, the symbol NMTKSYS will be created when a system     *
*  -level NAME/TOKEN list is requested and detected.  A symbol        *
*   NMTKASasid will be created when an address space-level            *
*   NAME/TOKEN list is requested and detected.  A symbol              *
*   NMTKTASKasidtcbaddr will be created when a task-level NAME/TOKEN  *
*   list is requested and detected.                                   *
*                                                                     *
*   The following IPCS exit services are demonstrated in this        *
*   program:                                                          *
*     Storage Access              (IPCS service code ADPLSACC)        *
*     Name Token Lookup           (IPCS service code ADPLSNTK)        *
*     Expanded Print              (IPCS service code ADPLSPR2)        *
*     Equate Symbol               (IPCS service code ADPLSEQS)        *
*     Table of Contents           (IPCS service code ADPLSNDX)        *
*     Get Symbol                  (IPCS service code ADPLSGTS)        *
*                                                                     *
*   Chapter 1Ø. in the OS/39Ø MVS IPCS Customization manual discusses *
*   the various IPCS exit services in detail.  This exit example      *
*   offers usage demonstration for only a handful of the available    *
*   services.                                                         *
*                                                                     *
*   In order to use the NMTKLST VERBX exit ensure that it is          *
*   linkedited somewhere into the load module search order for your   *
*   active IPCS session.  Linkedit JCL similar to the following can   *
*   be used:                                                          *
*      //IEWL     EXEC  PGM=HEWLHØ96,PARM='XREF,LIST,MAP,RENT'        *
*      //SYSPRINT DD    SYSOUT=*                                      *
*      //SYSUT1   DD    UNIT=SYSDA,SPACE=(CYL,(2,1))                  *
*      //OBJECT   DD    DSN=object.code.pds,DISP=SHR                  *
*      //SYSLMOD  DD    DSN=laod.library,DISP=SHR                     *
*      //SYSLIN   DD    *                                             *
*         INCLUDE OBJECT(NMTKLST)                                     *
*         ENTRY   NMTKLST                                             *
*         NAME    NMTKLST(R)                                          *
*                                                                     *
*   Register Usage Conventions:                                       *
*   RØ        :  work register, but generally available for use by    *
*                calls to system functions                            *
*   R1        :  contains the parameter address on entry; work        *
*                register, but generally available for use by calls   *
*                to system functions                                  *
*   R2 - R7   :  work registers                                       *
*   R8        :  ABDPL base register                                  *
*   R9        :  function specific parameter list address             *
*   R1Ø       :  future base register expansion                       *
*   R11       :  second base register                                 *
*   R12       :  first base register                                  *
*   R13       :  savearea/temporary storage address                   *
*   R14 - R15 :  work registers; return address and return code; but  *
*                generally available for use by calls to system       *
*                functions                                            *
*--------------------------------------------------------------------*
         STM   R14,R12,12(R13)       Save incoming registers
         LR    R12,R15               Copy module address
         LA    R11,4Ø95(,R12)        Set up second ...
         LA    R11,1(,R11)              base register
         USING NMTKLST,R12,R11       Set module addressability
         LR    R2,R1                 Copy parameter address
         LR    R3,R13                Copy savearea address
         STORAGE OBTAIN,LENGTH=WORKLEN,LOC=ANY
         LR    RØ,R1                 Copy working storage address
         LR    R14,R1                Again
         LR    R13,R1                Again
         L     R1,=A(WORKLEN)        Get length
         XR    R15,R15               Set fill byte
         MVCL  RØ,R14                Clear the storage
         USING WORKAREA,R13          Set addressability
         ST    R3,SAVEAREA+4         Save incoming savearea address
         LA    R9,WORKPACC           Get ADPLPACC address
         USING ADPLPACC,R9           Set addressability
         LR    R8,R2                 Get ABDPL address
         USING ABDPL,R8              Set addressability
         MVC   ASID(2),ADPLASID      Save the ASID
         MVC   CVTADDR(4),ADPLCVT    Save the CVT address
*---------------------------------------------------------------------*
*   The ADPLEXT contains the address of the extension pointer.  If    *
*   you want to process any input parameters passed to the VERBX      *
*   program they can be captured at this point and processed.         *
*                                                                     *
*   +Ø from the ADPLEXT address contains the parameter address.       *
*   +4 from the ADPLEXT address contains the CPPL address.            *
*                                                                     *
*   See comments earlier for the format of valid parameters.          *
*---------------------------------------------------------------------*
         L     R7,ADPLEXT            Get extension address
         LTR   R7,R7                 An extension?
         BZ    NOPARM                No - unusual, but nothing to do
         USING ADPLEXTN,R7           Set addressability
         L     R15,ADPLOPTR          Get parm buffer address
         LTR   R15,R15               A parameter?
         BZ    NOPARM                No - nothing to do
         LR    R5,R15                Copy parm buffer address
         S     R15,=F'4'             Point to length
         XR    R14,R14               Clear R14
         ICM   R14,B'ØØ11',Ø(R15)    Save the length
         LR    R6,R14                Copy to R6
         S     R6,=F'4'              Reduce by length word length
PARMLP   DS    ØH
         C     R6,=F'7'              Enough data for a keyword?
         BNL   CHKKYWDS              Yes - go through keyword check
         BCTR  R6,Ø                  Reduce by one for EX
         EX    R6,BLNKCLC            Blanks?
         BNE   BADPARM1              No - that's an error
         B     NOPARM                Done the parm check
CHKKYWDS DS    ØH
         C     R6,=F'16'             Enough for an ASCBADDR check?
         BL    CHKKYWD2              No - check second keyword
         CLC   Ø(9,R5),=C'ASCBADDR(' ASCBADDR keyword prefix?
         BNE   CHKKYWD2              No - check second keyword
         LA    R5,9(,R5)             Point past prefix
         S     R6,=F'9'              Reduce length
         LR    RØ,R6                 Save the length
         LR    R1,R5                 Save current buffer loc addr
         BAL   R14,ADDREXTR          Extract the address value
         C     R15,=F'8'             Parm was bad?
         BE    BADPARM1              Yes - issue a message
         C     R15,=F'4'             Symbol bad?
         BE    BADPARM2              Yes - issue a message
         LR    R5,R1                 Reload buffer address
         LR    R6,RØ                 Reload buffer length
         MVC   ASCBADDR(4),DBL1      Save the ASCB address value
         OI    KYWDFLAG,KYWDASCB     Set the ASCB keyword flag
         LA    R5,1(,R5)             Point to next data byte
         BCTR  R6,Ø                  Reduce length by one
         B     NEXTPARM              Prepare for next parm
CHKKYWD2 DS    ØH
         C     R6,=F'15'             Enough for a TCBADDR check?
         BL    CHKKYWD3              No - check third keyword
         CLC   Ø(8,R5),=C'TCBADDR('  TCBADDR keyword prefix?
         BNE   CHKKYWD3              No - check third keyword
         LA    R5,8(,R5)             Point past prefix
         S     R6,=F'8'              Reduce length
         LR    RØ,R6                 Save the length
         LR    R1,R5                 Save current buffer loc addr
         BAL   R14,ADDREXTR          Extract the address value
         C     R15,=F'8'             Parm was bad?
         BE    BADPARM1              Yes - issue a message
         C     R15,=F'4'             Symbol bad?
         BE    BADPARM2              Yes - issue a message
         LR    R5,R1                 Reload buffer address
         LR    R6,RØ                 Reload buffer length
         MVC   TCBADDR(4),DBL1       Save the TCB address value
         OI    KYWDFLAG,KYWDTCB      Set the TCB keyword flag
         LA    R5,1(,R5)             Point to next data byte
         BCTR  R6,Ø                  Reduce length by one
         B     NEXTPARM              Prepare for next parm
CHKKYWD3 DS    ØH
         C     R6,=F'8'              Enough for a NOSYSLVL check?
         BL    CHKKYWD4              No - check fourth keyword
         CLC   Ø(8,R5),=C'NOSYSLVL'  NOSYSLVL keyword?
         BNE   CHKKYWD4              No - check fourth keyword
         LA    R5,8(,R5)             Point past keyword
         S     R6,=F'8'              Reduce length
         OI    KYWDFLAG,KYWDNSYS     Set the NOSYSLVL keyword flag
         B     NEXTPARM              Prepare for next parm
CHKKYWD4 DS    ØH
         C     R6,=F'7'              Enough for a NOASLVL check?
         BL    CHKKYWD5              No - check fifth keyword
         CLC   Ø(7,R5),=C'NOASLVL'   NOASLVL keyword?
         BNE   CHKKYWD5              No - check fifth keyword
         LA    R5,7(,R5)             Point past keyword
         S     R6,=F'7'              Reduce length
         OI    KYWDFLAG,KYWDNAS      Set the NOASLVL keyword flag
         B     NEXTPARM              Prepare for next parm
CHKKYWD5 DS    ØH
         B     NEXTPARM              Prepare for next parm
NEXTPARM DS    ØH
         LTR   R6,R6                 End of parameter buffer?
         BZ    NOPARM                Yes - that's fine
         CLI   Ø(R5),C' '            A blank?
         BNE   BADPARM1              No - indicate invalid parm
         LA    R5,1(,R5)             Point to next data byte
         BCTR  R6,Ø                  Reduce length by one
         B     PARMLP                Check next keyword
*--------------------------------------------------------------------*
BADPARM1 DS    ØH
         LA    RØ,PRMMSG1L           Get message length
         LA    R1,PARMMSG1           Get message address
         BAL   R14,PRINTLN           Go print the line
         LA    RØ,1                  Set message length
         LA    R1,=C' '              Get message address
         BAL   R14,PRINTLN           Go print a blank line
         B     RETURN                Exit when parms are bad
*---------------------------------------------------------------------*
BADPARM2 DS    ØH
         LA    RØ,PRMMSG2L           Get message length
         LA    R1,PARMMSG2           Get message address
         BAL   R14,PRINTLN           Go print the line
         LA    RØ,1                  Set message length
         LA    R1,=C' '              Get message address
         BAL   R14,PRINTLN           Go print a blank line
         B     RETURN                Exit when parms are bad
         DROP  R7
*--------------------------------------------------------------------*
NOPARM   DS    ØH
         TM    KYWDFLAG,KYWDNSYS     NOSYSLVL specified?
         BNO   DONMTKL               No - no cross-reference required
         TM    KYWDFLAG,KYWDASCB     An ASCB address specified?
         BO    DONMTKL               Yes - settings are fine
         TM    KYWDFLAG,KYWDTCB      A TCB address specified?
         BO    DONMTKL               Yes - settings are fine
         NI    KYWDFLAG,255-KYWDNSYS Reset NOSYSLVL flag
DONMTKL  DS    ØH
*---------------------------------------------------------------------*
*   The keywords have been validated.  If the system-level NAME/TOKEN *
*   list will be displayed, we'll need to start with the CVT.         *
*---------------------------------------------------------------------*
*   Obtain the CVT.                                                   *
*---------------------------------------------------------------------*
         TM    KYWDFLAG,KYWDNSYS     System-level NAME/TOKEN display?
         BO    CHKASLVL              No - check address space-level
         MVC   ADPLPAAD(4),CVTADDR   Set address to the CVT
         MVC   ADPLDLEN(2),=AL2(CVTOSLVF+1-CVT) Set get length
         OI    ADPLPRDP,ADPLVIRT+ADPLSAMK Indicate virtual 24-bit addr
         L     R15,ADPLSERV          Get service routine address
         CALL  (15),                                                   X
               ((R8),                                                  X
               CODEACC,                                                X
               (R9)),MF=(E,CALLLST)
         MVC   CBNAME(4),=C'CVT '    Indicate control block acronym
         LTR   R15,R15               Were things ok?
         BNZ   NOSTORE               No - issue storage not found msg
*---------------------------------------------------------------------*
*   Obtain the ECVT.                                                  *
*---------------------------------------------------------------------*
         L     R1,ADPLPART           Get buffer location address
         USING CVT,R1
         MVC   CBNAME(4),=C'CVT '    Copy control block name
         MVC   CBADDR(4),ADPLPAAD    Copy control block address
         CLC   CBNAME(3),CVTCVT+1    Correct control block?
         BNE   CBERROR               No - no sense going on
         MVC   ADPLPAAD(4),CVTECVT   Get ECVT address
         MVC   ADPLDLEN(2),=AL2(ECVTEND-ECVT) Set get length
         NI    ADPLPRDP,255-ADPLSAMK Indicate virtual 31-bit addr
         DROP  R1
         L     R15,ADPLSERV          Get service routine address
         CALL  (15),                                                   X
               ((R8),                                                  X
               CODEACC,                                                X
               (R9)),MF=(E,CALLLST)
         MVC   CBNAME(4),=C'ECVT'    Indicate control block acronym
         LTR   R15,R15               Were things ok?
         BNZ   NOSTORE               No - issue storage not found msg
*---------------------------------------------------------------------*
*   Obtain the NTTP.                                                  *
*---------------------------------------------------------------------*
         L     R1,ADPLPART           Get buffer location address
         USING ECVT,R1
         MVC   CBNAME(4),=C'ECVT'    Copy control block name
         MVC   CBADDR(4),ADPLPAAD    Copy control block address
         CLC   CBNAME(4),ECVTECVT    Correct control block?
         BNE   CBERROR               No - no sense going on
         MVC   ADPLPAAD(4),ECVTNTTP  Get NTTP address
         DROP  R1
         CLC   ADPLPAAD(4),=F'Ø'     An NTTP address?
         BNE   SYSLVL                Yes - process system-level NM/TKN
         LA    RØ,L'MSG1ØØ           Set message length
         LA    R1,MSG1ØØ             Get message address
         BAL   R14,PRINTLN           Go print the line
         LA    RØ,1                  Set message length
         LA    R1,=C' '              Get message address
         BAL   R14,PRINTLN           Go print the line
         B     CHKASLVL              Go check address space level
SYSLVL   DS    ØH
         MVC   SYMNAME(32),=CL32'NMTKSYS' Set symbol name
         MVC   SYMLEN(4),=F'7'       Set symbol length
         MVC   SYMREMRK(4Ø),SYMREM1  Set symbol remark
         BAL   R14,SYMDEF            Define the symbol
         MVC   LINEBUF(L'TOCMSG1),TOCMSG1 Copy the TOC message value
         MVC   LINELEN(4),=AL4(L'TOCMSG1) Set the length
         BAL   R14,TOCENTRY          Create a TOC entry
         LA    RØ,L'MSGØ9Ø           Set message length
         LA    R1,MSGØ9Ø             Get message address
         BAL   R14,PRINTLN           Go print the line
         LA    RØ,1                  Set message length
         LA    R1,=C' '              Get message address
         BAL   R14,PRINTLN           Go print the line
         BAL   R14,NTTPPROC          Go process NAME/TOKEN list
         B     CHKASLVL              Go check address space level
*--------------------------------------------------------------------*
CHKASLVL DS    ØH
         TM    KYWDFLAG,KYWDASCB     ASCBADDR keyword specified?
         BNO   CHKTLVL               No - bypass a/s level check
*---------------------------------------------------------------------*
*   If the ASCBADDR keyword has been specified, we will need to       *
*   start off with the ASCB.                                          *
*---------------------------------------------------------------------*
*   Obtain the ASCB.                                                  *
*---------------------------------------------------------------------*
         L     R1,ADPLPART           Get buffer location address
         MVC   ADPLPAAD(4),ASCBADDR  Get ASCB address
         MVC   ADPLDLEN(2),=AL2(384) Set get length
         OI    ADPLPRDP,ADPLVIRT+ADPLSAMK Indicate virtual 24-bit addr
         L     R15,ADPLSERV          Get service routine address
         CALL  (15),                                                   X
               ((R8),                                                  X
               CODEACC,                                                X
               (R9)),MF=(E,CALLLST)
         MVC   CBNAME(4),=C'ASCB'    Indicate control block acronym
         LTR   R15,R15               Were things ok?
         BNZ   NOSTORE               No - issue storage not found msg
         USING ASCB,R1
         L     R1,ADPLPART           Get buffer location address
         MVC   CBNAME(4),=C'ASCB'    Copy control block name
         MVC   CBADDR(4),ADPLPAAD    Copy control block address
         CLC   CBNAME(4),ASCBASCB    Correct control block?
         BNE   CBERROR               No - no sense going on
         MVC   SAVEASID(2),ASCBASID  Save the ASID
         TM    KYWDFLAG,KYWDNAS+KYWDTCB NOASLVL & KYWDTCB flag set?
         BO    CHKTLVL               Yes - just do task level
*----------------------------------------------------------------------*
*   Obtain the ASSB.                                                   *
*----------------------------------------------------------------------*
         MVC   ADPLPAAD(4),ASCBASSB  Get ASSB address
         MVC   ADPLDLEN(2),=AL2(ASSBEND-ASSB) Set get length
         NI    ADPLPRDP,255-ADPLSAMK Indicate virtual 31-bit addr
         DROP  R1
         L     R15,ADPLSERV          Get service routine address
         CALL  (15),                                                   X
               ((R8),                                                  X
               CODEACC,                                                X
               (R9)),MF=(E,CALLLST)
         MVC   CBNAME(4),=C'ASSB'    Indicate control block acronym
         LTR   R15,R15               Were things ok?
         BNZ   NOSTORE               No - issue storage not found msg
*----------------------------------------------------------------------*
*   Obtain the NTTP.                                                   *
*----------------------------------------------------------------------*
         L     R1,ADPLPART           Get buffer location address
         USING ASSB,R1
         MVC   CBNAME(4),=C'ASSB'    Copy control block name
         MVC   CBADDR(4),ADPLPAAD    Copy control block address
         CLC   CBNAME(4),ASSBASSB    Correct control block?
         BNE   CBERROR               No - no sense going on
         MVC   ADPLPAAD(4),ASSBNTTP  Get NTTP address
         DROP  R1
         CLC   ADPLPAAD(4),=F'Ø'     An NTTP address?
         BNE   ASLVL                 Yes - process a/s level NM/TKN
         LA    RØ,L'MSG1Ø1           Set message length
         LA    R1,MSG1Ø1             Get message address
         BAL   R14,PRINTLN           Go print the line
         LA    RØ,1                  Set message length
         LA    R1,=C' '              Get message address
         BAL   R14,PRINTLN           Go print the line
         B     CHKTLVL               Go check task level
ASLVL    DS    ØH
         MVC   SYMNAME(32),=CL32'NMTKAS' Set symbol name prefix
         MVC   SYMLEN(4),=F'1Ø'      Set symbol length
         MVC   SYMREMRK(4Ø),SYMREM2  Set symbol remark
         XR    R15,R15               Clear R15
         ICM   R15,B'ØØ11',SAVEASID  Copy the ASID
         BAL   R14,HEXCNVT           Make it readable
         MVC   SYMNAME+6(4),DBL1+4   Copy ASID
         BAL   R14,SYMDEF            Define the symbol
         MVC   LINEBUF(L'TOCMSG2),TOCMSG2 Copy the TOC message value
         MVC   LINELEN(4),=AL4(L'TOCMSG2) Set the length
         BAL   R14,TOCENTRY          Create a TOC entry
         LA    RØ,L'MSGØ91           Set message length
         LA    R1,MSGØ91             Get message address
         BAL   R14,PRINTLN           Go print the line
         LA    RØ,1                  Set message length
         LA    R1,=C' '              Get message address
         BAL   R14,PRINTLN           Go print the line
         BAL   R14,NTTPPROC          Go process NAME/TOKEN list
         B     CHKTLVL               Go check task level
*--------------------------------------------------------------------*
CHKTLVL  DS    ØH
         TM    KYWDFLAG,KYWDTCB      TCBADDR keyword specified?
         BNO   RETURN                No - bypass task-level check
         CLC   SAVEASID(2),=2X'ØØ'   An ASID?
         BNE   NOASID1               Yes - bypass
         MVC   SAVEASID(2),ASID      Copy the default ASID
NOASID1  DS    ØH
         OI    FLAG1,TCBNTTP         Set task-level flag
*---------------------------------------------------------------------*
*   If the TCBADDR keyword has been specified, we will need to        *
*   start off with the TCB.                                           *
*---------------------------------------------------------------------*
*   Obtain the TCB.                                                   *
*---------------------------------------------------------------------*
         L     R1,ADPLPART           Get buffer location address
         MVC   ADPLPAAD(4),TCBADDR   Get TCB address
         MVC   ADPLDLEN(2),=AL2(TCBMNLEN) Set get length
         OI    ADPLPRDP,ADPLVIRT+ADPLSAMK Indicate virtual 24-bit addr
         MVC   ADPLASID(2),ASID      Set to default ASID
         CLC   SAVEASID(2),=2X'ØØ'   An ASID?
         BE    NOASID2               No - bypass
         MVC   ADPLASID(2),SAVEASID  Copy the ASID
NOASID2  DS    ØH
         L     R15,ADPLSERV          Get service routine address
         CALL  (15),                                                   X
               ((R8),                                                  X
               CODEACC,                                                X
               (R9)),MF=(E,CALLLST)
         MVC   CBNAME(4),=C'TCB '    Indicate control block acronym
         LTR   R15,R15               Were things ok?
         BNZ   NOSTORE               No - issue storage not found msg
*---------------------------------------------------------------------*
*   Obtain the STCB.                                                  *
```

```
*---------------------------------------------------------------------*
         L     R1,ADPLPART           Get buffer location address
         USING TCB,R1
         MVC   CBNAME(4),=C'TCB '    Copy control block name
         MVC   CBADDR(4),ADPLPAAD    Copy control block address
         CLC   CBNAME(4),TCBTCBID    Correct control block?
         BNE   CBERROR               No - no sense going on
         MVC   ADPLPAAD(4),TCBSTCB   Get STCB address
         MVC   ADPLDLEN(2),=AL2(TCBMNLEN) Set get length
         NI    ADPLPRDP,255-ADPLSAMK Indicate virtual 31-bit addr
         DROP  R1
         MVC   ADPLASID(2),ASID      Set to default ASID
         CLC   SAVEASID(2),=2X'ØØ'   An ASID?
         BE    NOASID3               No - bypass
         MVC   ADPLASID(2),SAVEASID  Copy the ASID
NOASID3  DS    ØH
         L     R15,ADPLSERV          Get service routine address
         CALL  (15),                                                   X
               ((R8),                                                  X
               CODEACC,                                                X
               (R9)),MF=(E,CALLLST)
         MVC   CBNAME(4),=C'STCB'    Indicate control block acronym
         LTR   R15,R15               Were things ok?
         BNZ   NOSTORE               No - issue storage not found msg
*---------------------------------------------------------------------*
*   Obtain the NTTP.                                                  *
*---------------------------------------------------------------------*
         L     R1,ADPLPART           Get buffer location address
         USING STCB,R1
         MVC   CBNAME(4),=C'STCB'    Copy control block name
         MVC   CBADDR(4),ADPLPAAD    Copy control block address
         CLC   CBNAME(4),STCBSTCB    Correct control block?
         BNE   CBERROR               No - no sense going on
         MVC   ADPLPAAD(4),STCBNTTP  Get NTTP address
         DROP  R1
         CLC   ADPLPAAD(4),=F'Ø'     An NTTP address?
         BNE   TASKLVL               Yes - process task-level NM/TKN
         LA    RØ,L'MSG1Ø2           Set message length
         LA    R1,MSG1Ø2             Get message address
         BAL   R14,PRINTLN           Go print the line
         LA    RØ,1                  Set message length
         LA    R1,=C' '              Get message address
         BAL   R14,PRINTLN           Go print the line
         B     RETURN                Go check task level
TASKLVL  DS    ØH
         MVC   SYMNAME(32),=CL32'NMTKTASK' Set symbol name prefix
         MVC   SYMLEN(4),=F'2Ø'      Set symbol length
         MVC   SYMREMRK(4Ø),SYMREM3  Set symbol remark
         XR    R15,R15               Clear R15
         ICM   R15,B'ØØ11',ADPLASID  Copy the ASID
         BAL   R14,HEXCNVT           Make it readable
         MVC   SYMNAME+8(4),DBL1+4   Copy ASID
         L     R15,TCBADDR           Copy TCB address
         BAL   R14,HEXCNVT           Make it readable
         MVC   SYMNAME+12(8),DBL1    Copy TCB address
         BAL   R14,SYMDEF            Define the symbol
         MVC   LINEBUF(L'TOCMSG3),TOCMSG3 Copy the TOC message value
         MVC   LINELEN(4),=AL4(L'TOCMSG3) Set the length
         BAL   R14,TOCENTRY          Create a TOC entry
         LA    RØ,L'MSGØ92           Set message length
         LA    R1,MSGØ92             Get message address
         BAL   R14,PRINTLN           Go print the line
         LA    RØ,1                  Set message length
         LA    R1,=C' '              Get message address
         BAL   R14,PRINTLN           Go print the line
         BAL   R14,NTTPPROC          Go process NAME/TOKEN list
         B     RETURN                Go check task level
*---------------------------------------------------------------------*
NTTPPROC DS    ØH
         ST    R14,REGSAVE2          Save the return address
NEXTNTTP DS    ØH
         MVC   ADPLDLEN(2),=AL2(72)  Set get length
         NI    ADPLPRDP,255-ADPLSAMK Indicate virtual 31-bit addr
         MVC   ADPLASID(2),ASID      Set to default ASID
         CLC   SAVEASID(2),=2X'ØØ'   An ASID?
         BE    NOASID4               No - bypass
         MVC   ADPLASID(2),SAVEASID  Copy the ASID
NOASID4  DS    ØH
         L     R15,ADPLSERV          Get service routine address
         CALL  (15),                                                  X
               ((R8),                                                 X
               CODEACC,                                               X
               (R9)),MF=(E,CALLLST)
         MVC   CBNAME(4),=C'NTTP'    Indicate control block acronym
         LTR   R15,R15               Were things ok?
         BNZ   NOSTORE               No - issue storage not found msg
*--------------------------------------------------------------------*
         L     R3,ADPLPART           Get buffer location address
         CLC   Ø(4,R3),=C'NTTH'      The header?
         BE    NTTH                  Yes - process header
         CLC   Ø(4,R3),=C'NTTE'      A NAME/TOKEN entry?
         BE    NTTE                  Yes - process the entry
         MVC   CBNAME(4),=C'NTT*'    Copy control block name
         MVC   CBADDR(4),ADPLPAAD    Copy control block address
         B     CBERROR               No sense going on
CHKNTTP  DS    ØH
         MVC   ADPLPAAD(4),64(R3)    Get NTTP address
         CLC   ADPLPAAD(4),=F'Ø'     End of the list?
         BNE   NEXTNTTP              No - process next entry
         L     R14,REGSAVE2          Restore return address
         BR    R14                   Return
NTTH     DS    ØH
         B     CHKNTTP               Process next entry
NTTE     DS    ØH
         MVC   WORKNTKP(132),NTKP    Copy NTKP model
         LA    R9,WORKNTKP           Get workarea address
         MVC   NTKPMODN-NTKP(8,R9),=C'NMTKLST ' Copy module name
         MVC   NTKPNAME-NTKP(16,R9),8(R3) Copy NAME
         MVC   NTKPASID-NTKP(2,R9),=2X'ØØ' Clear the ASID
         CLC   SAVEASID(2),=2X'ØØ'   An ASID?
         BE    NOASID5               No - bypass
         MVC   NTKPASID-NTKP(2,R9),SAVEASID Copy the ASID
NOASID5  DS    ØH
         TM    FLAG1,TCBNTTP         Task level?
         BNO   NOTLVL                No - not a task-level NAME/TOKEN
         MVC   NTKPTCBP-NTKP(4,R9),TCBADDR Copy the TCB address
NOTLVL   DS    ØH
         L     R15,ADPLSERV          Get service routine address
         CALL  (15),                                                   X
               ((R8),                                                  X
               CODENTK,                                                X
               (R9)),MF=(E,CALLLST)
         LTR   R15,R15               Were things ok?
         BNZ   NONMTK                No - issue error
         LA    RØ,1                  Set message length
         LA    R1,=C' '              Get message address
         BAL   R14,PRINTLN           Go print a blank line
*--------------------------------------------------------------------*
         B     CHKNTTP               Process next entry
*--------------------------------------------------------------------*
RETURN   DS    ØH
         L     R3,SAVEAREA+4         Load incoming savearea address
         LR    R1,R13                Get working storage address
         STORAGE RELEASE,LENGTH=WORKLEN,ADDR=(R1)
         LR    R13,R3                Restore incoming savearea address
         LM    R14,R12,12(R13)       Restore incoming registers
         XR    R15,R15               Set return code
         BR    R14                   Return
*---------------------------------------------------------------------*
*   Termination message routines.                                     *
*---------------------------------------------------------------------*
NOSTORE  DS    ØH
         MVI   LINEBUF,C' '          Set fill byte
         MVC   LINEBUF+1(131),LINEBUF Clear the area
         MVC   LINEBUF(STMSGL),STORMSG Copy the message
         BAL   R14,HEXCNVT           Make the rc readable
         MVC   LINEBUF+51(2),DBL1+6  Copy rc into message
         ICM   R15,B'1111',ADPLPAAD  Get control block address
         BAL   R14,HEXCNVT           Make it readable
         MVC   LINEBUF+37(8),DBL1    Copy c/b address into message
         MVC   LINEBUF+29(4),CBNAME  Copy c/b name into message
         LA    RØ,STMSGL             Get message length
         LA    R1,LINEBUF            Get message address
         BAL   R14,PRINTLN           Go print the line
TERM     DS    ØH
         LA    RØ,1                  Set message length
         LA    R1,=C' '              Get message address
         BAL   R14,PRINTLN           Go print a blank line
         LA    RØ,TRMMSG1L           Get message length
         LA    R1,TERMMSG1           Get message address
         BAL   R14,PRINTLN           Go print the line
         B     RETURN                We're done
*-------------------------------------------------------------------*
NONMTK   DS    ØH
         MVI   LINEBUF,C' '          Set fill byte
         MVC   LINEBUF+1(131),LINEBUF Clear the area
         MVC   LINEBUF(NMTMSG1L),NMTKMSG1 Copy the message
         BAL   R14,HEXCNVT           Make the rc readable
         MVC   LINEBUF+49(2),DBL1+6  Copy rc into message
         LA    RØ,NMTMSG1L           Get message length
         LA    R1,LINEBUF            Get message address
         BAL   R14,PRINTLN           Go print the line
         B     TERM                  All done
*-------------------------------------------------------------------*
CBERROR  DS    ØH
         MVI   LINEBUF,C' '          Set fill byte
         MVC   LINEBUF+1(131),LINEBUF Clear the area
         MVC   LINEBUF(CBEMSG1L),CBEMSG1 Copy the message
         L     R15,CBADDR            Get control block address
         BAL   R14,HEXCNVT           Make the rc readable
         MVC   LINEBUF+51(8),DBL1    Copy c/b address into message
         MVC   LINEBUF+2Ø(4),CBNAME  Copy c/b name into message
         LA    RØ,CBEMSG1L           Get message length
         LA    R1,LINEBUF            Get message address
         BAL   R14,PRINTLN           Go print the line
         B     TERM                  All done
*--------------------------------------------------------------------*
*   Subroutines                                                      *
*--------------------------------------------------------------------*
ADDREXTR DS    ØH
*--------------------------------------------------------------------*
*   The ADDREXTR subroutine isolates the address value for a given   *
*   keyword.                                                         *
*                                                                    *
*   On entry:  RØ - contains the remaining length of the parameter   *
*                   buffer area                                      *
*              R1 - contains the current parameter buffer location   *
*                   pointer                                          *
*              R8 - contains the address of the ABDPL                *
*                                                                    *
*   On exit:   RØ - contains the updated remaining length of the      *
*                   parameter buffer area                             *
*              R1 - contains the updated parameter buffer location    *
*                   pointer                                           *
*             R15 - Ø:  address value successfully captured           *
*                   4:  specified symbol value not located            *
*                   8:  address value not a valid hexadecimal address *
*---------------------------------------------------------------------*
         STM   RØ,R15,REGSAVE          Save the registers
         LR    R5,R1                   Save current buffer address
         LR    R6,RØ                   Save current buffer length
         LR    R3,R5                   Save start address for later
         LR    R2,R6                   Save buffer length for later
         LA    R4,Ø(R6,R5)             Set ending address
         CLI   Ø(R5),C'Ø'              First character < Ø?
         BL    SYMCHKØ                 Yes - not an obvious addr value
         CLI   Ø(R5),C'9'              First character > 9?
         BH    SYMCHKØ                 Yes - not an obvious addr value
ADDRCHKØ DS    ØH
         XR    R15,R15                 Clear counter
         XR    R14,R14                 Clear counter
         NI    FLAG1,255-ENDZERO       Reset leading zero flag
ADDRCHK1 DS    ØH
         CR    R5,R4                   End of buffer?
         BNL   ADDRRET8                Yes - indicate parm error
         CLI   Ø(R5),C'.'              Delimiter?
         BE    ADDRDEL1                Yes - go process
         CLI   Ø(R5),C')'              Delimiter?
         BE    ADDRDEL2                Yes - go process
         CLI   Ø(R5),C'A'              A?
         BL    ADDRRET8                Low - indicate parm error
         CLI   Ø(R5),C'F'              F?
         BNH   XOK1                    Not high - good hex character
         CLI   Ø(R5),C'Ø'              Zero?
         BL    ADDRRET8                Low - indicate parm error
         BE    ZEROCHK1                Yes - check for leading zero
         CLI   Ø(R5),C'9'              Nine?
         BH    ADDRRET8                High - indicate parm error
XOK1     DS    ØH
         LA    R15,1(,R15)             Add one to address length count
         LA    R5,1(,R5)               Point to next parm byte
         BCTR  R6,Ø                    Reduce buffer length by one
         OI    FLAG1,ENDZERO           Set end of leading zeros flag
         B     ADDRCHK1                Go check next byte
ZEROCHK1 DS    ØH
         TM    FLAG1,ENDZERO           Done with leading zeros?
         BO    XOK1                    Yes - just another hex character
         LA    R15,1(,R15)             Add one to address length count
         LA    R14,1(,R14)             Add one to leading zero count
         LA    R5,1(,R5)               Point to next parm byte
         BCTR  R6,Ø                  Reduce buffer length by one
         B     ADDRCHK1              Go check next byte
ADDRDEL1 DS    ØH
         LR    R7,R5                 Save current buffer address
         LA    R5,1(,R5)             Point to next parm byte
         BCTR  R6,Ø                  Reduce buffer length by one
         CR    R5,R4                 End of buffer?
         BNL   ADDRRET8              Yes - indicate parm error
         CLI   Ø(R5),C')'            End delimiter?
         BNE   ADDRRET8              No - indicate parm error
         B     ADDRCHK2              Check for good address value
ADDRDEL2 DS    ØH
         LR    R7,R5                 Save current buffer address
         B     ADDRCHK2              Check for good address value
ADDRCHK2 DS    ØH
         TM    FLAG1,ENDZERO         All leading zeros?
         BO    ADDRCHK3              No - not a special condition
         LR    R3,R7                 Point to current buffer location
         BCTR  R3,Ø                  Back up one byte
         LA    R15,1                 Set length to one
ADDRCHK3 DS    ØH
         SR    R15,R14               Reduce length by leading zero #
         C     R15,=F'8'             Too long?
         BH    ADDRRET8              Yes - indicate parm error
         C     R15,=F'Ø'             Too short?
         BE    ADDRRET8              Yes - indicate parm error
         LA    R3,Ø(R14,R3)          Point past leading zeros
         MVC   DBL2(8),=8C'Ø'        Set fill value
         L     R14,=F'8'             Set maximum length
         SR    R14,R15               Reduce by length of value
         LA    R14,DBL2(R14)         Point to target area
         BCTR  R15,Ø                 Reduce by one for EX
         EX    R15,ADDRMVC           Copy the address value
         TR    DBL2(8),TRTABLE       Change from EBCDIC to hex
         PACK  DBL1(5),DBL2(9)       Pack the address value
         B     ADDRRETØ              Return success
*--------------------------------------------------------------------*
SYMCHKØ  DS    ØH
         NI    FLAG1,255-SYMVAL      Reset flag
         NI    FLAG1,255-FIRSTCHR    Reset flag
         LA    R7,WORKESSY           Get ESSY area address
         MVC   Ø(ESSYLRL,R7),ESSY    Initialize the area
         LA    R9,ESSYSYM-ESSY(,R7)  Get address of symbol name area
         MVC   Ø(31,R9),=31C' '      Initialize the symbol name area
         XR    R15,R15               Clear counter
SYMCHK1  DS    ØH
         CLI   Ø(R5),C'$'            Valid start character?
         BE    SYMCHK2               Yes - keep going
         CLI   Ø(R5),C'#'            Valid start character?
         BE    SYMCHK2               Yes - keep going
         CLI   Ø(R5),C'@'            Valid start character?
         BE    SYMCHK2               Yes - keep going
         CLI   Ø(R5),C'.'            A delimiter character?
         BE    SYMDEL1               Yes - process delimiter
         CLI   Ø(R5),C')'            A delimiter character?
         BE    SYMGET                Yes - locate symbol
         CLI   Ø(R5),C'A'            Valid start character?
         BL    ADDRRET8              No - indicate parm error
         CLI   Ø(R5),C'I'            Valid start character?
         BNH   SYMCHK2               Yes - keep going
         CLI   Ø(R5),C'J'            Valid start character?
         BL    ADDRRET8              No - indicate parm error
         CLI   Ø(R5),C'R'            Valid start character?
         BNH   SYMCHK2               Yes - keep going
         CLI   Ø(R5),C'S'            Valid start character?
         BL    ADDRRET8              No - indicate parm error
         CLI   Ø(R5),C'Z'            Valid start character?
         BNH   SYMCHK2               Yes - keep going
         TM    FLAG1,FIRSTCHR        First character validated?
         BNO   ADDRRET8              No - indicate parm error
         CLI   Ø(R5),C'Ø'            Valid symbol name character?
         BL    ADDRRET8              No - indicate parm error
         CLI   Ø(R5),C'9'            Valid symbol name character?
         BH    ADDRRET8              No - indicate parm error
SYMCHK2  DS    ØH
         OI    FLAG1,FIRSTCHR        Set flag on
         MVC   Ø(1,R9),Ø(R5)         Copy to symbol name area
         CLI   Ø(R5),C'A'            Hex character?
         BL    NOTHEX                No - must treat as a symbol
         CLI   Ø(R5),C'F'            Hex character?
         BNH   SYMHEX                Yes - just go on for now
         CLI   Ø(R5),C'Ø'            Hex character?
         BL    NOTHEX                No - must treat as a symbol
         CLI   Ø(R5),C'9'            Hex character?
         BNH   SYMHEX                Yes - just go on for now
NOTHEX   DS    ØH
         OI    FLAG1,SYMVAL          Set symbol value flag
SYMHEX   DS    ØH
         LA    R9,1(,R9)             Point to next target byte
         LA    R5,1(,R5)             Point to next source byte
         LA    R15,1(,R15)           Add one to count
         BCTR  R6,Ø                  Reduce buffer length by one
         CR    R5,R4                 End of buffer?
         BNL   ADDRRET8              Yes - indicate parm error
         B     SYMCHK1               Check for more symbol name chars
SYMDEL1  DS    ØH
         TM    FLAG1,SYMVAL          Symbol name character detected?
         BO    ADDRRET8              Yes - indicate parm error
         LR    R5,R3                 Reset buffer start address
         LR    R6,R2                 Reset buffer length
         B     ADDRCHKØ              Check for valid address value
SYMGET   DS    ØH
         C     R15,=F'31'            Symbol length ok?
         BH    ADDRRET8              No - indicate parm error
         L     R15,ADPLSERV          Load addr of exit services router
         CALL  (15),                                                   X
               ((R8),                                                  X
               CODEGTS,                                                X
               (R7)),MF=(E,CALLLST)
         LTR   R15,R15               Got symbol information?
         BNZ   ADDRRET4              No - set failure return indicator
         MVC   DBL1(4),ESSYLAD-ESSY(R7) Copy logical address
ADDRRETØ DS    ØH
         LR    RØ,R6                 Save current length
         LR    R1,R5                 Save current buffer address
         LM    R2,R14,REGSAVE+8      Restore registers
         XR    R15,R15               Set return code to Ø
         BR    R14                   Return
ADDRRET4 DS    ØH
         LR    RØ,R6                 Save current length
         LR    R1,R5                 Save current buffer address
         LM    R2,R14,REGSAVE+8      Restore registers
         LA    R15,4                 Set return code to 4
         BR    R14                   Return
ADDRRET8 DS    ØH
         LR    RØ,R6                 Save current length
         LR    R1,R5                 Save current buffer address
         LM    R2,R14,REGSAVE+8      Restore registers
         LA    R15,8                 Set return code to 8
         BR    R14                   Return
*---------------------------------------------------------------------*
PRINTLN  DS    ØH
*---------------------------------------------------------------------*
*   The PRINTLN subroutine generates a line of output using the       *
*   IPCS print service.                                               *
*                                                                     *
*   On entry:  RØ - contains the length of the output line            *
*              R1 - contains the address of the output line           *
*              R8 - contains the address of the ABDPL                 *
*                                                                     *
*   On exit:  R15 - contains the return code from the IPCS print      *
*                   service                                           *
*---------------------------------------------------------------------*
         STM   RØ,R15,REGSAVE        Save the registers
         LA    R7,WORKPPR2           Get BLSUPPR2 address
         MVC   Ø(PPR2999-PPR2ØØØ,R7),PPR2 Copy the PPR2 model
         MVC   PPR2BUF-PPR2(4,R7),ADPLBUF Copy print buffer address
         ST    RØ,PPR2BUFL-PPR2(,R7) Save the message length
         L     R3,PPR2BUFL-PPR2(,R7) Copy the message length
         L     R15,ADPLBUF           Get message buffer address
         MVI   Ø(R15),C' '           Set fill byte
         MVC   1(131,R15),Ø(R15)     Clear message buffer area
         L     R15,ADPLBUF           Get message buffer address
         BCTR  R3,Ø                  Reduce length by one for EX
         EX    R3,MSGMVC             Copy the message
         MVI   PPR2PFL1-PPR2(R7),PPR2MSG Indicate buffer contains a msg
         L     R15,ADPLSERV          Get service routine address
         CALL  (15),                                                   X
               ((R8),                                                  X
               CODEPR2,                                                X
               (R7)),MF=(E,CALLLST)
PRINTLNE DS    ØH
         LM    RØ,R14,REGSAVE        Restore required registers
         BR    R14                   Return
*---------------------------------------------------------------------*
HEXCNVT  DS    ØH
*---------------------------------------------------------------------*
*   The HEXCNVT subroutine converts the hex contents of R15 to        *
*   a human readable format in variable DBL1.                         *
*---------------------------------------------------------------------*
         ST    R15,DBL2              Save the value
         UNPK  DBL1(9),DBL2(5)       Unpack it
         NC    DBL1(8),=8X'ØF'       Turn off high nibble
         TR    DBL1(8),=C'Ø123456789ABCDEF' Make it readable
         BR    R14                   Return
*---------------------------------------------------------------------*
TOCENTRY DS    ØH
*---------------------------------------------------------------------*
*   The TOCENTRY subroutine adds an entry to the IPCS table of        *
*   contents.                                                         *
*                                                                     *
*   On entry, LINELEN contains the length of the TOC message          *
*   (greater than Ø, less than 41).  LINEBUF contains the value       *
*   of the TOC message                                                *
*---------------------------------------------------------------------*
         STM   RØ,R15,REGSAVE        Save the registers
         L     R15,ADPLBUF           Get message buffer address
         L     R3,LINELEN            Get TOC message length
         LA    R3,4(,R3)             Add in length of length word
         BCTR  R3,Ø                  Reduce by one for EX
         EX    R3,TOCMVC             Copy the TOC message
         L     R15,ADPLSERV          Get service routine address
         CALL  (15),                                                   X
               ((R8),                                                  X
               CODENDX),                                               X
               MF=(E,CALLLST)
         LM    RØ,R14,REGSAVE        Restore required registers
         BR    R14                   Return
*--------------------------------------------------------------------*
SYMDEF   DS    ØH
```

```
*----------------------------------------------------------------------*
*   Create an IPCS symbol for the specified symbol name.  On entry     *
*   to this routine:                                                   *
*     SYMNAME  - contains the name of the symbol to be defined         *
*     SYMLEN   - contains the length of the symbol name                *
*     SYMREMRK - contains the remark to be associated with this symbol*
*----------------------------------------------------------------------*
         STM   RØ,R15,REGSAVE2       Save the registers
         NI    FLAG1,255-SYMTRY      Reset the SYMTRY flag
SYM1     DS    ØH
         LA    R7,WORKESSY           Get ESSY area address
         MVC   Ø(ESSYLRL,R7),ESSY    Initialize the area
         MVC   ESSYSYM-ESSY(32,R7),SYMNAME Copy symbol name
         MVC   ESSYAST-ESSY(2,R7),=C'CV' Move in address space type
         MVC   ESSYLAD-ESSY(4,R7),ADPLPAAD Move in header address
         MVC   ESSYDLE-ESSY(4,R7),=A(72) Move in header length
         MVC   ESSYDTY-ESSY(1,R7),=C'U' Indicate type as AREA
         MVC   ESSYDTD-ESSY(32,R7),ESSYSYM-ESSY(R7) Move in data name
         MVC   ESSYRL-ESSY(2,R7),=AL2(4Ø) Move in remark length
         MVC   ESSYRT-ESSY(4Ø,R7),SYMREMRK Copy symbol remark
*----------------------------------------------------------------------*
*   Determine proper ASID                                              *
*----------------------------------------------------------------------*
         MVC   ESSYAS2-ESSY+2(2,R7),=AL2(1) Move in default ASID
         CLC   SYMNAME(8),=C'NMTKSYS' System level?
         BE    SYSSYM                Yes - default ASID is good
         MVC   ESSYAS2-ESSY+2(2,R7),SAVEASID Copy in ASID
SYSSYM   DS    ØH
         OI    ESSYFC-ESSY(R7),ESSYFCD Set NODROP attribute on symbol
         L     R15,ADPLSERV          Load addr of exit services router
         CALL  (15),                                                   X
               ((R8),                                                  X
               CODEEQS,                                                X
               (R7)),MF=(E,CALLLST)
         C     R15,=F'12'            Symbol equate ok?
         BL    SYM1E                 Yes - go on
         TM    FLAG1,SYMTRY          Is this the second try?
         BO    NOSYM1                Yes - issue message
         OI    FLAG1,SYMTRY          Set flag
         B     SYM1                  Try a second time
SYM1E    DS    ØH
         LM    RØ,R14,REGSAVE2       Restore the registers
         BR    R14                   Return
NOSYM1   DS    ØH
         MVI   LINEBUF,C' '          Set fill byte
         MVC   LINEBUF+1(131),LINEBUF Clear the area
         MVC   LINEBUF(L'SYMDMSG1),SYMDMSG1 Copy the message
         LA    R1,LINEBUF+L'SYMDMSG1 Point to target area
         L     R14,SYMLEN            Get symbol length
         MVC   Ø(32,R1),SYMNAME      Copy symbol name
         LA    R1,Ø(R14,R1)          Point to target area
         MVC   Ø(9,R1),=C' - RC(  )' Move in remainder of message
         BAL   R14,HEXCNVT           Make the rc readable
         MVC   6(2,R1),DBL1+6        Copy rc into message
         L     R14,SYMLEN            Get symbol length
         LA    RØ,L'SYMDMSG1+9(,R14) Get message length
         LA    R1,LINEBUF            Get message address
         BAL   R14,PRINTLN           Go print the line
         LA    RØ,1                  Set message length
         LA    R1,=C' '              Get message address
         BAL   R14,PRINTLN           Go print a blank line
         B     SYM1E                 Go back for more
*--------------------------------------------------------------------*
*   Executed instructions                                             *
*--------------------------------------------------------------------*
BLNKCLC  CLC   Ø(*-*,R5),=6C' '      Remaining parm data blanks?
ADDRMVC  MVC   Ø(*-*,R14),Ø(R3)      Copy the address value
MSGMVC   MVC   Ø(*-*,R15),Ø(R1)      Copy the message
TOCMVC   MVC   Ø(*-*,R15),LINELEN    Copy the TOC message
*--------------------------------------------------------------------*
*   Constants                                                         *
*--------------------------------------------------------------------*
CODEACC  DC    A(ADPLSACC)           Storage access service code
CODEPR2  DC    A(ADPLSPR2)           Expanded print service code
CODEEQS  DC    A(ADPLSEQS)           Equate symbol service code
CODENDX  DC    A(ADPLSNDX)           Table of contents service code
CODENTK  DC    A(ADPLSNTK)           Name/token lookup service code
CODEGTS  DC    A(ADPLSGTS)           Get symbol service code
*---------------------------------------------------------------------*
ESSY     BLSRESSY DSECT=NO           IPCS equate symbol parm list
*---------------------------------------------------------------------*
PPR2     BLSUPPR2 DSECT=NO           IPCS expanded print parm list
*--------------------------------------------------------------------*
NTKP     BLSQNTKP DSECT=NO           IPCS NAME/TOKEN lookup parm list
*--------------------------------------------------------------------*
TOCMSG1  DC    C'System-level NAME/TOKEN list'
TOCMSG2  DC    C'Address space-level NAME/TOKEN list'
TOCMSG3  DC    C'Task-level NAME/TOKEN list'
SYMREM1  DC    CL4Ø'System-level NAME/TOKEN header'
SYMREM2  DC    CL4Ø'Address space-level NAME/TOKEN header'
SYMREM3  DC    CL4Ø'Task-level NAME/TOKEN header'
MSGØ9Ø   DC    C'NMTKNØ9ØI - Processing system-level NAME/TOKEN table'
MSGØ91   DC    C'NMTKNØ91I - Processing address space-level NAME/TOKEN x
               table'
MSGØ92   DC    C'NMTKNØ92I - Processing task-level NAME/TOKEN table'
MSG1ØØ   DC    C'NMTKN1ØØI - No system-level NAME/TOKEN table'
MSG1Ø1   DC    C'NMTKN1Ø1I - No address space-level NAME/TOKEN table'
MSG1Ø2   DC    C'NMTKN1Ø2I - No task-level NAME/TOKEN table'
PARMMSG1 DC    C'NMTKN11ØI - Invalid parm detected.  Valid parms are: '
         DC    C' ''NOSYSLVL'' ''NOASLVL'' ''ASCBADDR(hexaddr)'' and '
         DC    C'''TCBADDR(hexaddr)'''
PRMMSG1L EQU   *-PARMMSG1
PARMMSG2 DC    C'NMTKN111I - Invalid symbol value or symbol not '
         DC    C'defined to IPCS.'
PRMMSG2L EQU   *-PARMMSG2
STORMSG  DC    C'NMTKN112I - Unable to locate xxxx at xxxxxxxx - '
         DC    C'RC(xx)'
STMSGL   EQU   *-STORMSG
NMTKMSG1 DC    C'NMTKN121I - Error locating NAME/TOKEN entry - '
         DC    C'RC(xx)'
NMTMSG1L EQU   *-NMTKMSG1
SYMDMSG1 DC    C'NMTKN131I - Error detected defining symbol '
         DC    C'- RC(xx)'
SYDMSG1L EQU   *-SYMDMSG1
CBEMSG1  DC    C'NMTKN132I - Invalid xxxx control block detected at '
         DC    C'xxxxxxxx'
CBEMSG1L EQU   *-CBEMSG1
TERMMSG1 DC    C'NMTKN189I - NAME/TOKEN list display terminated.'
TRMMSG1L EQU   *-TERMMSG1
*--------------------------------------------------------------------*
TRTABLE  DC    256X'8Ø'
         ORG   TRTABLE+Ø
         DC    C'Ø123456789ABCDEF'
         ORG   TRTABLE+193
         DC    X'ØAØBØCØDØEØF'
         ORG   TRTABLE+24Ø
         DC    X'ØØØ1Ø2Ø3Ø4Ø5Ø6Ø7Ø8Ø9'
         ORG   ,
*--------------------------------------------------------------------*
         LTORG ,
*--------------------------------------------------------------------*
WORKAREA DSECT
SAVEAREA DS    18F
CALLLST  CALL  ,(,,,,,,),MF=L
REGSAVE  DS    16F
REGSAVE2 DS    16F
ASID     DS    XL2
KYWDFLAG DS    XL1
KYWDNSYS EQU   X'8Ø'                 NOSYSLVL specified
KYWDNAS  EQU   X'4Ø'                 NOASLVL specified
KYWDASCB EQU   X'2Ø'                 ASCB specified
KYWDTCB  EQU   X'1Ø'                 TCB specified
ASCBADDR DS    F                     ASCB address from ASCB keyword
TCBADDR  DS    F                     TCB address from TCB keyword
SAVEASID DS    XL2                   ASID for specified ASCB
CBNAME   DS    CL8                   Control block name save area
CBADDR   DS    ØD,CL(8)              Control block address save area
SYMNAME  DS    CL(32)                Symbol name
SYMREMRK DS    CL(4Ø)                Symbol remark
SYMLEN   DS    F                     Symbol length
```

```
FLAG1    DS    XL1
SYMTRY   EQU   X'8Ø'
TCBNTTP  EQU   X'4Ø'
ENDZERO  EQU   X'2Ø'
SYMVAL   EQU   X'1Ø'
FIRSTCHR EQU   X'Ø8'
CVTADDR  DS    F
WORKPACC DS    ØD,CL(ADPLLACC)
WORKESSY DS    ØD,CL(ESSYHRL)
WORKPPR2 DS    ØD,CL(PPR2999-PPR2ØØØ)
WORKNTKP DS    ØD,CL(132)
LINELEN  DS    F
LINEBUF  DS    CL(132)
DBL1     DS    2D
DBL2     DS    2D
WORKLEN  EQU   *-WORKAREA
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*--------------------------------------------------------------------*
         BLSABDPL DSECT=YES,                                           X
               AMDEXIT=YES,                                            X
               AMDOSEL=NO,                                             X
               AMDPACC=YES,                                            X
               AMDPFMT=YES,                                            X
               AMDPECT=NO,                                             X
               AMDPSEL=NO
         PRINT NOGEN
         CVT   DSECT=YES
         IHAECVT ,
         IHAASCB ,
         IHAASSB ,
         IKJTCB  ,
         IHASTCB ,
         END
```

*Systems Programmer (Canada)* © Xephon 2004

# MVS news

UFD Solutions AG has announced PDSXref, a kind of 'search-engine' for z/OS.

PDSXref quickly locates any member in all partitioned datasets for an entire system, displays all PDS libraries containing a specified member, and allows users to perform the usual system management functions on the retrieved member (such as browse, edit, copy, print, compare, etc).

Additionally, PDSXref allows users to detect duplicate or redundant members across all the different PDS libraries of an entire system.

For further information contact:
UFD AG Schweiz, Arnold Böcklin-Strasse 29, CH-4011 Basel, Switzerland
Tel: (61) 271 65 50.
URL: http://www.ubs.com/e/keytools/tools_for_z_os/mbrxref.html.

* * *

Compuware has announced File-AID/CS 3.0, its product for managing data on multiple databases and platforms.

File-AID/CS 3.0 now includes a data conversion utility that converts and transforms multiple data types through a graphical interface – including direct access to data residing in OS/390 and zSeries environments.

The product has an improved related-extract and load that allows users to subset related data from one database and load the data into another with minimal database expertise. File-AID/CS 3.0 also contains an enhanced Compare facility enabling users to validate results across heterogeneous data environments from a single point-of-control.

File-AID/CS supports Oracle, DB2 UDB, Microsoft SQL Server, Sybase, XML, Access, Excel, VSAM, IMS, DB2 UDB for z/OS, QSAM and ASCII and runs on Windows, Unix, and Linux platforms.

For further information contact:
Compuware, One Campus Martius, Detroit, MI 48226, USA.
Tel: (313) 227 7300.
URL: http://www.compuware.com/products/fileaid/default.htm.

* * *

Cybermation has announced enhancements to its Enterprise Systems Platform (ESP) job scheduling solution.

Espresso Release 4 has enhancements to Cybermation's Unix- and Windows-based ESP job scheduling engine. Workload Manager v5.4 has enhancements that increase performance and business agility for Cybermation's z/OS-based ESP job scheduling engine. OneView is a single graphical display for users who are scheduling jobs between ESP Workload Manager and ESP Espresso. System Agent for z/OS allows ESP Espresso users to define, manage, and monitor workload for z/OS environments from Unix and Windows environments. Web Services Interface for z/OS environments provides integration between job scheduling environments and Web service-enabled applications.

For further information contact:
Cybermation, 125 Commerce Valley Drive West, 8th Floor, Markham, ON, Canada L3T 7W4.
Tel: (905) 707 4400.
URL: http://www.cybermation.com/solutions/jobscheduling/zos/