192

MVS update

September 2002

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 £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

ASCII/EBCDIC translation

I often get asked to convert data written with ASCII code tables into EBCDIC and *vice versa*.

To this end, I wrote a routine in COBOL, which can be used in IBM Language Environment using the standard linkage section.

The routine can be easily called by COBOL or Assembler programs, or by CLIST and REXX commands.

The program, CONVEBAS, receives three parameters:

1 Input – a maximum of 500 alphanumeric characters.

2 Output – a maximum of 500 alphanumeric characters.

3 Function – 'EA' for EBCDIC to ASCII, and 'AE' for ASCII to EBCDIC.

The routine has been tested under OS/390 2.10.

CONVEBAS

```
       ID DIVISION.
       PROGRAM-ID.     CONVEBAS.
      ** ---------------------------------------------**
      ** this routine allows for converting a string of **
      ** characters in either ASCII/EBCDIC code tables. **
      ** ---------------------------------------------- **
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
       DATA DIVISION.
       FILE SECTION.
      /
       WORKING-STORAGE SECTION.
      ** ------------------------------------------------- **
       Ø1 W-TAB-ASCII-EBCDIC.
         Ø3 FILLER         PIC X(24) VALUE
ØØ-Ø7        " ØØ,Ø1,Ø2,Ø3,37,2D,2E,2F".
         Ø3 FILLER         PIC X(24) VALUE
Ø8-ØF        " 16,Ø5,25,ØB,ØC,ØD,ØE,ØF".
         Ø3 FILLER         PIC X(24) VALUE
1Ø-17        " 1Ø,11,12,13,3C,3D,32,26".
```

```
          Ø3 FILLER          PIC X(24) VALUE
18-1F         " 18,19,3F,27,1C,1D,1E,1F".
          Ø3 FILLER          PIC X(24) VALUE
2Ø-27         " 4Ø,5A,7F,7B,5B,6C,5Ø,7D".
          Ø3 FILLER          PIC X(24) VALUE
28-2F         " 4D,5D,5C,4E,6B,6Ø,4B,61".
          Ø3 FILLER          PIC X(24) VALUE
3Ø-37         " FØ,F1,F2,F3,F4,F5,F6,F7".
          Ø3 FILLER          PIC X(24) VALUE
38-3F         " F8,F9,7A,5E,4C,7E,6E,6F".
          Ø3 FILLER          PIC X(24) VALUE
4Ø-47         " 7C,C1,C2,C3,C4,C5,C6,C7".
          Ø3 FILLER          PIC X(24) VALUE
48-4F         " C8,C9,D1,D2,D3,D4,D5,D6".
          Ø3 FILLER          PIC X(24) VALUE
5Ø-57         " D7,D8,D9,E2,E3,E4,E5,E6".
          Ø3 FILLER          PIC X(24) VALUE
58-5F         " E7,E8,E9,AD,EØ,BD,5F,6D".
          Ø3 FILLER          PIC X(24) VALUE
6Ø-67         " 79,81,82,83,84,85,86,87".
          Ø3 FILLER          PIC X(24) VALUE
68-6F         " 88,89,91,92,93,94,95,96".
          Ø3 FILLER          PIC X(24) VALUE
7Ø-77         " 97,98,99,A2,A3,A4,A5,A6".
          Ø3 FILLER          PIC X(24) VALUE
78-7F         " A7,A8,A9,CØ,4F,DØ,A1,Ø7".
          Ø3 FILLER          PIC X(24) VALUE
8Ø-87         " ØØ,Ø1,Ø2,Ø3,37,2D,2E,2F".
          Ø3 FILLER          PIC X(24) VALUE
88-8F         " 16,Ø5,25,ØB,ØC,ØD,ØE,ØF".
          Ø3 FILLER          PIC X(24) VALUE
9Ø-97         " 1Ø,11,12,13,3C,3D,32,26".
          Ø3 FILLER          PIC X(24) VALUE
98-9F         " 18,19,3F,27,1C,1D,1E,1F".
          Ø3 FILLER          PIC X(24) VALUE
AØ-A7         " 4Ø,5A,7F,7B,5B,6C,5Ø,7D".
          Ø3 FILLER          PIC X(24) VALUE
A8-AF         " 4D,5D,5C,4E,6B,6Ø,4B,61".
          Ø3 FILLER          PIC X(24) VALUE
BØ-B7         " FØ,F1,F2,F3,F4,F5,F6,F7".
          Ø3 FILLER          PIC X(24) VALUE
B8-BF         " F8,F9,7A,5E,4C,7E,6E,6F".
          Ø3 FILLER          PIC X(24) VALUE
CØ-C7         " 7C,C1,C2,C3,C4,C5,C6,C7".
          Ø3 FILLER          PIC X(24) VALUE
C8-CF         " C8,C9,D1,D2,D3,D4,D5,D6".
          Ø3 FILLER          PIC X(24) VALUE
DØ-D7         " D7,D8,D9,E2,E3,E4,E5,E6".
          Ø3 FILLER          PIC X(24) VALUE
D8-DF         " E7,E8,E9,AD,EØ,BD,5F,6D".
          Ø3 FILLER          PIC X(24) VALUE
```

```
E0-E7         " 79,81,82,83,84,85,86,87".
          03 FILLER          PIC X(24) VALUE
E8-EF         " 88,89,91,92,93,94,95,96".
          03 FILLER          PIC X(24) VALUE
F0-F7         " 97,98,99,A2,A3,A4,A5,A6".
          03 FILLER          PIC X(24) VALUE
F8-FF         " A7,A8,A9,C0,4F,D0,A1,07".
       ** ------------------------------------------------ **
        01 W-TAB-ASEB REDEFINES W-TAB-ASCII-EBCDIC.
          03 W-ELEM-ASEB OCCURS 256 INDEXED BY A.
            05 FILLER             PIC X.
            05 W-ELEM-ASEB-VAL    PIC X(2).
       ** ------------------------------------------------ **
        01 W-TAB-EBCDIC-ASCII.
          03 FILLER          PIC X(24) VALUE
00-07         " 00,01,02,03,20,09,20,7F".
          03 FILLER          PIC X(24) VALUE
08-0F         " 20,20,20,0B,0C,0D,0E,0F".
          03 FILLER          PIC X(24) VALUE
10-17         " 10,11,12,13,20,20,08,20".
          03 FILLER          PIC X(24) VALUE
18-1F         " 18,19,20,20,1C,1D,1E,1F".
          03 FILLER          PIC X(24) VALUE
20-27         " 20,20,20,20,20,0A,17,1B".
          03 FILLER          PIC X(24) VALUE
28-2F         " 20,20,20,20,20,05,06,07".
          03 FILLER          PIC X(24) VALUE
30-37         " 20,20,16,20,20,20,20,04".
          03 FILLER          PIC X(24) VALUE
38-3F         " 20,20,20,20,14,15,20,1A".
          03 FILLER          PIC X(24) VALUE
40-47         " 20,20,20,20,20,20,20,20".
          03 FILLER          PIC X(24) VALUE
48-4F         " 20,20,20,2E,3C,28,2B,7C".
          03 FILLER          PIC X(24) VALUE
50-57         " 26,20,20,20,20,20,20,20".
          03 FILLER          PIC X(24) VALUE
58-5F         " 20,20,21,24,2A,29,3B,5E".
          03 FILLER          PIC X(24) VALUE
60-67         " 2D,2F,20,20,20,20,20,20".
          03 FILLER          PIC X(24) VALUE
68-6F         " 20,20,7C,2C,25,5F,3E,3F".
          03 FILLER          PIC X(24) VALUE
70-77         " 20,20,20,20,20,20,20,20".
          03 FILLER          PIC X(24) VALUE
78-7F         " 20,60,3A,23,40,27,3D,22".
          03 FILLER          PIC X(24) VALUE
80-87         " 20,61,62,63,64,65,66,67".
          03 FILLER          PIC X(24) VALUE
88-8F         " 68,69,20,20,20,20,20,20".
          03 FILLER          PIC X(24) VALUE
```

```
9Ø-97        " 2Ø,6A,6B,6C,6D,6E,6F,7Ø".
         Ø3 FILLER         PIC X(24) VALUE
98-9F        " 71,72,2Ø,7D,2Ø,29,2Ø,2Ø".
         Ø3 FILLER         PIC X(24) VALUE
AØ-A7        " 2D,7E,73,74,75,76,77,78".
         Ø3 FILLER         PIC X(24) VALUE
A8-AF        " 79,7A,2Ø,2Ø,2Ø,5B,2Ø,2Ø".
         Ø3 FILLER         PIC X(24) VALUE
BØ-B7        " 3Ø,31,32,33,34,35,36,37".
         Ø3 FILLER         PIC X(24) VALUE
B8-BF        " 38,39,2Ø,2Ø,2Ø,5D,23,2D".
         Ø3 FILLER         PIC X(24) VALUE
CØ-C7        " 7B,41,42,43,44,45,46,47".
         Ø3 FILLER         PIC X(24) VALUE
C8-CF        " 48,49,2Ø,2Ø,2Ø,2Ø,2Ø,2Ø".
         Ø3 FILLER         PIC X(24) VALUE
DØ-D7        " 7D,4A,4B,4C,4D,4E,4F,5Ø".
         Ø3 FILLER         PIC X(24) VALUE
D8-DF        " 51,52,2Ø,2Ø,2Ø,2Ø,2Ø,2Ø".
         Ø3 FILLER         PIC X(24) VALUE
EØ-E7        " 5C,2Ø,53,54,55,56,57,58".
         Ø3 FILLER         PIC X(24) VALUE
E8-EF        " 59,5A,2Ø,2Ø,2Ø,2Ø,2Ø,2Ø".
         Ø3 FILLER         PIC X(24) VALUE
FØ-F7        " 3Ø,31,32,33,34,35,36,37".
         Ø3 FILLER         PIC X(24) VALUE
F8-FF        " 38,39,7C,2Ø,2Ø,2Ø,2Ø,2Ø".
      ** ------------------------------------------------- **
       Ø1 W-TAB-EBAS REDEFINES W-TAB-EBCDIC-ASCII.
         Ø3 W-ELEM-EBAS OCCURS 256 INDEXED BY E.
           Ø5 FILLER             PIC X.
           Ø5 W-ELEM-EBAS-VAL    PIC X(2).
       Ø1   W-VAL-DISPLAY.
         Ø3 W-VAL-S              PIC X.
            Ø3 W-VAL-S-N REDEFINES W-VAL-S PIC 9.
         Ø3 W-VAL-D              PIC X.
            Ø3 W-VAL-D-N REDEFINES W-VAL-D PIC 9.
       Ø1   W-PLUTO.
         Ø3 FILLER                    PIC X VALUE LOW-VALUE.
         Ø3 W-PIPPO                   PIC X.
       Ø1   W-CONV-DEC REDEFINES W-PLUTO PIC 9(4) COMP.
       Ø1   W-RIS                     PIC 9(4) COMP.
       Ø1   W-RIS-A REDEFINES W-RIS.
         Ø3 FILLER                    PIC X.
         Ø3 W-RIS-HEX                 PIC X.
      /
       LINKAGE SECTION.
       Ø1 P-TAB-INPUT.
         Ø3 P-ELEM-INPUT OCCURS 5ØØ INDEXED BY I PIC X.
       Ø1 P-TAB-OUTPUT.
         Ø3 P-ELEM-OUTPUT OCCURS 5ØØ INDEXED BY O PIC X.
```

```
 Ø1   P-CONV                      PIC XX.
   88 RIC-CONV           VALUE "EA" "AE".
/
 PROCEDURE DIVISION USING P-TAB-INPUT P-TAB-OUTPUT P-CONV.
 LBLØØØ.
      MOVE ALL SPACES TO P-TAB-OUTPUT.
      IF   P-TAB-INPUT = ALL SPACES
         OR P-TAB-INPUT = ALL LOW-VALUE
           OR P-TAB-INPUT = ALL HIGH-VALUE
            OR NOT RIC-CONV
             MOVE 99 TO RETURN-CODE
              GO TO LBL999
      END-IF.
      SET  I TO 1.
      SET  O TO 1.
 LBL1ØØ.
      MOVE P-ELEM-INPUT (I) TO W-PIPPO.
      IF   P-CONV = "AE"
          AND W-CONV-DEC > 127
           MOVE 98 TO RETURN-CODE
            GO TO LBL999
      END-IF.
      IF   P-ELEM-INPUT (I) NOT = LOW-VALUE
          SET  E TO W-CONV-DEC
          SET  E UP BY 1
          SET  A TO E
      END-IF.
      IF   P-ELEM-INPUT (I) =  LOW-VALUE
          SET  E TO 1
          SET  A TO 1
      END-IF.
      IF     P-CONV = "EA"
      MOVE   W-ELEM-EBAS-VAL (E) TO W-VAL-DISPLAY
      END-IF.
      IF     P-CONV = "AE"
      MOVE   W-ELEM-ASEB-VAL (A) TO W-VAL-DISPLAY
      END-IF.
      MOVE   ALL ZEROES TO W-RIS.
      EVALUATE TRUE
        WHEN   W-VAL-D = "A" ADD 1Ø TO W-RIS
        WHEN   W-VAL-D = "B" ADD 11 TO W-RIS
        WHEN   W-VAL-D = "C" ADD 12 TO W-RIS
        WHEN   W-VAL-D = "D" ADD 13 TO W-RIS
        WHEN   W-VAL-D = "E" ADD 14 TO W-RIS
        WHEN   W-VAL-D = "F" ADD 15 TO W-RIS
        WHEN   W-VAL-D IS NUMERIC
                  ADD W-VAL-D-N TO W-RIS
      END-EVALUATE.
      EVALUATE TRUE
        WHEN   W-VAL-S = "A" ADD 16Ø TO W-RIS
        WHEN   W-VAL-S = "B" ADD 176 TO W-RIS
```

```
            WHEN   W-VAL-S = "C" ADD 192 TO W-RIS
            WHEN   W-VAL-S = "D" ADD 208 TO W-RIS
            WHEN   W-VAL-S = "E" ADD 224 TO W-RIS
            WHEN   W-VAL-S = "F" ADD 240 TO W-RIS
            WHEN   W-VAL-S IS NUMERIC
                      COMPUTE W-RIS = W-RIS + (W-VAL-S-N * 16)
          END-EVALUATE.
          MOVE W-RIS-HEX TO  P-ELEM-OUTPUT (O).
          IF I > 499
                   THEN GO TO LBL999
          END-IF.
          SET I UP BY 1.
          SET O TO I.
          GO TO LBL100.
        LBL999.
          IF RETURN-CODE NOT = 0
            DISPLAY "RETURN-CODE     : " RETURN-CODE
            DISPLAY "FUNCTION/PARM   : " P-CONV
            DISPLAY "INPUT/PARM      : " P-TAB-INPUT
            DISPLAY "OUTPUT/PARM     : " P-TAB-OUTPUT
          END-IF.
          GOBACK.
```

Massimo Ambrosini
Systems Programmer (Italy)

IPL information

The following utility was created to show when the last IPL took place, along with the parameters used and related information. It comprises a REXX EXEC and an ISPF panel. The information is taken from the system memory. The variables in the EXEC have the same names as the Assembler macros that map those areas. Those macros can be found in SYS1.MACLIB, along with a description of each field. The pointers and data areas used are outlined below; the macro names for each area are indicated in parentheses:

```
Storage (208) ----->              (ppca)
                  CPUID

Storage (10) ----> (cvt)
                  CVTSNAME
                  CVTRLSTG
```

```
CVTEXT2 ---->  (cvt - cvtxtnt2)
               CVTNUCLS
               CVTIOCID

CVTECVT ---->  (ihaecvt)
               ECVTSPLX ---->    (ipa)
               ECVTMLPR          IPAIODFU
               ECVTIPA           IPALOADS
                                 IPAHWNAM
CVTSMCA ---->  (ieesmca)         IPALPNAM
               SMCAIDTE          IPALPDSN
               SMCAITME          IPALPDDV
               SMCASID           IPAIOSUF
                                 IPASPSUF
CVTFMCTL ----> (erbstgst)        IPASCVOL
               STGSPRTN          IPASCDSN
               STGSTNM           IPASXNAM
               STGSTSF           IPAPLDSN
               STGSTTM           IPAPLVOL
               STGSTY1
               STGSTDT

CVTSYSAD ----> ucb chain
               Sysaddev
               sysadvol
```

The output produced looks as follows:

```
+--------------------   IPL information  ---------------------+
| System name....: SN15         MVS level..: SP6.1.5         |
| Sysplex name...: SXP2         Fmid.......: HBB9945         |
| Hardware name..: RZSTAS21     CPU model..: 9672            |
| LPAR name......: SYSDEV       Version....: 7A              |
| Lpar number....: 2            Serial.....: 249870          |
|                                                            |
| Last ipl date.........: 2002/01/15   22:44:59              |
| Iplparm unit and dsn.: 065D  SYS1.IPLPARM                  |
| Load parameter........: 065D09 1                           |
|                                                            |
| IODF unit and dsn....: 065D  HCD.IODF09                    |
| Dsn creation date....: 2001/09/14   13.21.37               |
|                                                            |
| Master catalog dsn...: CATALOG.MASTER.SYSDEV1              |
| Master catalog volume: VSY321                              |
+------------------------------------------------------------+
```

IPL REXX SOURCE CODE

```
/* REXX MVS *=================================================*/
```

```
/*  IPL  -  Display information about last IPL.  Information is */
/*          gathered directly from memory.  Variable names used */
/*  in this EXEC are, whenever possible, identical to the names */
/*  of the Assembler macros mapping that memory area.           */
/*  Those macros can be found is SYS1.MACLIB, and their name is */
/*  in the comment that precedes each section of this EXEC.     */
/*                                                              */
/*==============================================================*/
/*  CPU information                                             */
/*==============================================================*/
pcca     = storage(208,4)
cpuid    = storage(d2x(c2d(pcca)+4),22)
cpuversi = substr(cpuid,1,2)
cpuseria = substr(cpuid,3,6)
cpumodel = substr(cpuid,9,4)
/*==============================================================*/
/* CVT, ECVT - Communications vector table (Macros CVT IHAECVT) */
/*==============================================================*/
cvt      = storage(10,4)
prodn    = storage(d2x(c2d(cvt)-40),8)
prodi    = storage(d2x(c2d(cvt)-32),8)
cvtsname = storage(d2x(c2d(cvt)+340),8)
cvtrlstg = storage(d2x(c2d(cvt)+856),4)
cvtrlstg = c2d(cvtrlstg)%1024
cvtext2  = storage(d2x(c2d(cvt)+328),4)
cvtnucls = storage(d2x(c2d(cvtext2)+4),1)
cvtiocid = storage(d2x(c2d(cvtext2)+6),2)
ecvt     = storage(d2x(c2d(cvt)+140),4)
ecvtsplx = storage(d2x(c2d(ecvt)+8),8)
ecvtmlpr = storage(d2x(c2d(ecvt)+168),8)
/*==============================================================*/
/*  IPA - Initialization parameter area (Macro IPA)             */
/*==============================================================*/
ecvtipa  = storage(d2x(c2d(ecvt)+392),4)
ipaiodfu = storage(d2x(c2d(ecvtipa)+016),4)
iodfu    = ipaiodfu
ipaloads = storage(d2x(c2d(ecvtipa)+020),2)
ipahwnam = storage(d2x(c2d(ecvtipa)+024),8)
ipalpnam = storage(d2x(c2d(ecvtipa)+032),8)
ipalpdsn = storage(d2x(c2d(ecvtipa)+048),44)
ipalpddv = storage(d2x(c2d(ecvtipa)+092),4)
lpddv    = ipalpddv
ipaiosuf = storage(d2x(c2d(ecvtipa)+096),2)
ipaspsuf = storage(d2x(c2d(ecvtipa)+160),2)
ipascvol = storage(d2x(c2d(ecvtipa)+224),6)
ipascdsn = storage(d2x(c2d(ecvtipa)+234),44)
ipasxnam = storage(d2x(c2d(ecvtipa)+352),8)
ipapldsn = storage(d2x(c2d(ecvtipa)+416),44)
ipaplvol = storage(d2x(c2d(ecvtipa)+461),6)
/*==============================================================*/
/*  SMCA - SMF control table (Macro IEESMCA)                    */
```

```
/*================================================================*/
cvtsmca  = storage(d2x(c2d(cvt)+197),3)
smcaidte = storage(d2x(c2d(cvtsmca)+340),4)
smcaidte = c2d(smcaidte)%16
smcaidte = d2x(smcaidte)
smcaidte = convert_date(smcaidte)
smcaitme = storage(d2x(c2d(cvtsmca)+336),4)
smcaitme = c2d(smcaitme)%100
smcaitme = convert_time(smcaitme)
smcasid  = storage(d2x(c2d(cvtsmca)+16),4)
/*================================================================*/
/*  cvtsysad - pointer to the UCB chain entry of sysres volume  */
/*================================================================*/
cvtsysad = storage(d2x(c2d(cvt)+48),4)
sysadvol = storage(d2x(c2d(cvtsysad)+28),6)
sysaddev = storage(d2x(c2d(cvtsysad)+4),2)
sysaddev = c2x(sysaddev)
/*================================================================*/
/*  STGS - Global supervisor table (macro ERBSTGST)             */
/*================================================================*/
cvtfmctl = storage(d2x(c2d(cvt)+796),4)
stgsprtn = storage(d2x(c2d(cvtfmctl)+174),1)
stgsprtn = c2d(stgsprtn)
stgstnm  = storage(d2x(c2d(cvtfmctl)+258),44)
stgstsf  = storage(d2x(c2d(cvtfmctl)+302),2)
stgsttm  = storage(d2x(c2d(cvtfmctl)+314),8)
stgsty1  = storage(d2x(c2d(cvtfmctl)+322),2)
stgstdt  = storage(d2x(c2d(cvtfmctl)+306),8)
stgstdt  = stgsty1||right(stgstdt,2)"/"left(stgstdt,3)||,
           substr(stgstdt,4,2)
call display_panel
exit
/*================================================================*/
/*  Subroutines                                                   */
/*================================================================*/
convert_time: procedure
 arg seconds
 hh = seconds%3600
 mm = (seconds//3600)%60
 ss = (seconds//60)
return right(hh,2,"0")":"right(mm,2,"0")":"right(ss,2,"0")
convert_date: procedure
 arg julian
 cent = left(julian,1)
 year = substr(julian,2,2)
 day  = right(julian,3)
 if year//4 = 0 then extra = 1
 else extra = 0
 do mnt = 1 to 12
   dayprv = day
   mntprv = mnt
```

```
   select
     when mnt=4 | mnt=6 | mnt=9 | mnt=11 then ndays = 30
     when mnt=2 & extra=1 then ndays = 29
     when mnt=2 & extra=0 then ndays = 28
     otherwise ndays = 31
   end
   day = day - ndays
   if day < 0 | day = 0 then leave mnt
 end
return "20"year"/"right(mntprv,2,"0")"/"right(dayprv,2,"0")
display_panel:
 address ispexec
 'addpop row(1) column(1)'
 'display panel(ipl)'
 address tso
return
```

IPL PANEL SOURCE CODE

```
)ATTR
  % TYPE(TEXT)   INTENS(HIGH) SKIP(ON) COLOR(YELLOW)
  + TYPE(TEXT)   INTENS(HIGH) SKIP(ON) COLOR(TURQUOISE)
  ? TYPE(TEXT)   INTENS(HIGH) SKIP(ON) COLOR(RED)
  $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON) COLOR(GREEN) CAPS(OFF)
  * TYPE(OUTPUT) INTENS(HIGH) SKIP(ON) COLOR(WHITE) CAPS(OFF)
  _ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON) COLOR(BLUE)  CAPS(OFF)
)BODY WINDOW(74,18)
+
+System name....:$cvtsname    +MVS level..:$prodn
+Sysplex name...:$ecvtsplx    +Fmid.......:$prodi
+Hardware name..:$ipahwnam    +CPU model..:$cpumodel
+LPAR name......:$ipalpnam    +Version....:$cpuversi
+Lpar number....:$stgsprtn    +Serial.....:$cpuseria
+
%Last ipl date........:*smcaidte    *smcaitme
%Iplparm unit and dsn.:*lpddv*ipalpdsn
%Load parameter.......:*ecvtmlpr
+
%IODF unit and dsn....:_iodfu_stgstnm
%Dsn creation date....:_stgstdt     _stgsttm
+
%Master catalog dsn...:_ipascdsn
%Master catalog volume:_ipascvol
+
)INIT
&zwinttl = ' IPL information '
)END
```

OS/390 hints and tips

More and more organizations are making use of a corporate intranet, and technical and support departments are often invited to contribute information for Web pages aimed at the user community. The text of this article provides a collection of hints and tips designed for technical users, primarily in support, development, and testing, of an OS/390 environment. It could form the basis of the technical Web pages for end users.

The key to maintaining a successful Web site is to have useful information. The key to holding a regular audience and maintaining interest is to present that information in an accessible way and to provide frequent updates. The information here can easily be expanded to cover more subjects and could be extended to include site-specific information. The subjects covered in this article are ISPF, ISPF edit, SDSF, JCL and batch, and Bookmanager.

There is little that needs be changed before publishing as a Web page but note the following:

1 The JCL for ISPF in batch should be updated with the correct dataset names for the IBM ISPF libraries for your site.

2 There is a section on Bookmanager. You may wish to include details of how to access this at your site. If appropriate you could use hyperlinks to the books and bookshelves.

The hints and tips presented here apply to OS/390 2.10. Some features are not available at all in earlier releases and some exist but have a different syntax. You are advised to verify the functioning of all features as they operate on your system.

ISPF

Further information about any of the commands referenced in this section can be found online in ISPF HELP, accessed via PF1 or the HELP command, and in the OS/390 ISPF manuals, specifically the *OS/390 ISPF User's Guide*.

Getting help

The first source of information in ISPF is the **HELP** command, accessed by PF1. For software products this often takes you into a tutorial about how to use the product. When a new version is installed the tutorial is a valuable resource for identifying changes and new features.

ISPF manuals can be found in the OS/390 ISPF Bookshelf, in the OS/390 Collection in BookManager.

ISRDDN

The command **TSO ISRDDN** displays a panel showing the current dataset allocations for the TSO session, along with the active DDname. From this screen you can browse/edit/view an individual dataset or the first four in a concatenation. Scroll right (PF11) to view dataset DCB information.

You can use the **MEMBER** subcommand to search all allocated datasets, or a concatenation of datasets for a particular member name. The syntax is **MEMBER *mem-name DDname***, where *mem-name* is the name of the member and *DDname* is the optional concatentation to limit the search. For example, **MEMBER ISR@PRIM ISPPLIB** will search all datasets in the ISPPLIB concatenation and highlight those containing the member ISR@PRIM.

Retrieving commands – RETRIEVE, RETP, and RETF

Use the RETRIEVE command to recall previously entered commands to the command line. The commands are displayed one at a time in last-in-first-out (LIFO) sequence. You can retrieve a command and resubmit it for processing. You can also edit a command before resubmitting it. This is most useful when assigned to a PF key.

RETP: instead of recalling commands one at a time, the RETP command causes a pop-up panel to be displayed with a selectable list of the last 25 commands in the retrieve stack. With this panel you can see at a glance the previous commands entered. This command is also useful when in SDSF because the SDSF RETRIEVE command will only retrieve its own commands.

RETF: another variation is the RETF command, which is the same as RETRIEVE except that it retrieves commands from the command stack moving in the direction from the oldest command in the command stack toward the most recent commands in the command stack (FIFO).

Manipulating pop-up windows

ISPF pop-up windows can be moved to view the screen below. Using the mouse or cursor keys position the cursor on the window edge. Press *Enter*. Move the cursor to the new position and press *Enter* again. The window will be redisplayed in the new relative location.

Pop-up windows can also be resized to fit the whole screen. If the window includes a command line, type **RESIZE**. The window will be redisplayed and maximized. To return to the original pop-up size enter the **RESIZE** command again. If the window does not include a command line you can still maximize but you must have assigned the RESIZE command to a PF key.

ISPF command shell – TSOCMD

You may know the ISPF command shell as the TSO command processor, or option 6. It can now be accessed by entering the command **TSOCMD** on the ISPF command line. You can then enter the commands you want, or send and receive files via the terminal emulator, then press PF3 (END) to return to your previous screen. It is very useful because you don't need to interrupt an edit session, for example.

A similar function is provided by the CMDE command.

CMDE command

If you need more space than the current ISPF command line allows, the **CMDE** command provides the ability to enter up to 234 characters using an extended entry field provided. This looks a bit like the PDF Option 6 panel and it will process TSO commands, REXX EXECs, and CLISTs, but not EDIT commands.

CMDE will accept any parameters and these will appear in the new command line for you to complete the command. Unlike TSOCMD, when the command has been executed you are returned to the previous screen.

This panel is processed much like the PDF Option 6 panel. Data passed to this panel will normally be translated to uppercase. Data passed from here will remain as it appears on the panel.

Improving CLIST/EXEC performance

Commands entered on the ISPF and TSO command lines can be programs or CLISTs/EXECs. If you know that the command is a CLIST/EXEC you can speed its execution by prefixing the command name with a % sign, eg **TSO %WHO**.

This bypasses program search and goes straight to the libraries allocated to SYSPROC, finding the command faster and saving the processing involved in the search.

Sorting a member list

The **SORT** command sorts a member list by any two fields displayed on the member list, except the line command field and RENAME field. The field names are the column headings. For example, **SORT ID** will sort the member list by the userid in the ID field. If you type **SORT** and omit the field name you can select the order by moving the cursor to the column heading and pressing *Enter*.

The sort sequence is determined by the field – the default field, NAME, sorts in ascending order, but CHANGED sorts in descending order showing the most recently changed first.

Refreshing a dataset list

When changes are made to the datasets listed in the ISPF 3.4 option, for example renames or deletes, the list can be refreshed by entering the command **REFRESH** on the ISPF command line. There is no need to end the display and reselect.

Member list refresh

An ISPF member list display can be refreshed at any time using the **REFRESH** command. This is similar to the REFRESH command on dataset lists. For example, if you are in Option 3.1 and rename a member in a PDS, you can enter **refresh** on the command line, and the new

member name will be displayed straight away.

Option 3.3 Move/Copy

The target dataset for the Move/Copy utility (Option 3.3) can now be allocated automatically if it does not exist. When ISPF finds that the TO dataset does not exist, a pop-up window is displayed allowing you to allocate the dataset and choose to use the allocation attributes of the original dataset or specify new attributes. If you don't want to allocate the dataset for any reason you can CANCEL out of the allocation.

This feature was introduced with OS/390 2.10.

EPDF command

Through the **EPDF** command, Edit, View, and Browse functions are available from any command line. From the command line, all you need do is type: **EPDF *'dataset name' options*** where *dataset name* is the dataset you wish to edit (including member name if appropriate) and *options* are parameters such as **BROWSE,** which lets you browse the dataset. For example:

```
EPDF 'TSG00I.ISPF.SRC(IEFUJV)' VIEW
```

For further information, enter **EPDF** on the command line and you will be shown a display of all the valid options.

SuperC: FMSTOP

The ISPF Compare program, SuperC, supports a performance option, **FMSTOP**, which stops processing on the first mismatch for file compare. FMSTOP is also supported for string searches.

ISPF EDIT

Further information about ISPF edit commands and features can be found in the *OS/390 ISPF Edit and Edit Macros* manual.

COMPARE command

One of the most useful commands in ISPF EDIT is the **COMPARE** command. It allows you to display the differences between the data being edited, and another file. Lines which exist only in the current file

are shown with a label on the line. Lines which exist only in the other file are displayed as information lines and are shown in white. You can use the Make Data (MD) line command to add those lines to your edit session.

The syntax is **COMPARE *member*** to compare the data to a member in the current dataset or concatenation. **COMPARE *dataset*** compares the data to a sequential dataset or to a member of the same name in the specified dataset, and **COMPARE *dataset*(*member*)** will compare the data to the specified member of the specified dataset. **COMPARE *** (or **COMPARE SESSION**) will compare your current edit session against the data saved on disk.

COMPARE (with no parameters) presents the Edit/Compare Settings panel, which allows you to modify the type of comparison and display.

As from OS/390 2.8 it is no longer a requirement to SAVE the data before performing a compare. Note that the above syntax applies to OS/390 2.8 ISPF and different syntax applies to earlier releases.

MODEL command

The **MODEL** command is invaluable when writing ISPF dialogs. A MODEL is a group of source statements and optional notes that provide sample data for creating and editing ISPF dialogs. The models contain prototype lines indicating the syntax of the requested dialog element, and showing parts which you may overtype with application-specific data. Notes and comments in these models explain the syntax, parameters, and possible return codes, as necessary.

For example, entering **MODEL LIBDEF** and the A(after)/B(before) line command will insert sample code into the current member, showing the full syntax of the ISPF LIBDEF service and notes on its use. The inserted code is in the language of the dataset being edited, eg CLIST, COBOL, REXX, etc.

MODEL (with no parameters) brings up a selection list of all the models available. The edit MODEL command is also used to identify the class from which to obtain subsequent models, for example MODEL CLASS COBOL, for COBOL source. The model class-name defaults to the name of your current EDIT profile so this command need be used only to specify the class-name if the EDIT profile name does not correspond

to the desired language, for example, COBOL in an SRC library, or REXX EXECs in a CLIST library. MODEL CLASS on its own will display a selection list of all valid classes.

Highlighting syntax and logic

The HILITE command and dialog allow you to set the colouring options for language-sensitive colouring in the ISPF editor. Type **HILITE ON** to automatically detect the language, including JCL, and colour accordingly. Unclosed quotes and brackets are automatically detected and highlighted. HILITE can be abbreviated to HI.

The editor can also highlight logical do/end blocks and match if/then/else constructs in several languages. **HILITE LOGIC** will highlight both do/end and if/then. **HILITE DOLOGIC** will highlight only do/end blocks and **HILITE IFLOGIC** only if/then constructs.

Unmatched ENDs or ELSEs are highlighted in reverse video pink to make them stand out. To find the first mismatched END or ELSE in the file, enable logic highlighting, scroll to the bottom of the file, and type **HI SEARCH** on the command line.

HI (with no keywords) will display a panel for you to select the type of highlighting required. **HILITE OFF** turns it all off.

MakeData command (MD)

The **MD** command is an edit line command. It is used to convert a non-data line, eg NOTE, MSG, or new line, from the compare command, into a data line. It can be used in a block pair as **MMD .. MMD** to convert a block of data instead of a single line.

Line commands as PF keys

Most users are familiar with the use of TSO commands on PF keys, but PF keys can also be set to execute ISPF edit line commands. To do this you must prefix the line command with a **colon** (:) in the PF key definition. This will cause the command to be entered at the first input field of the line where the cursor is positioned when the function key is pressed instead of on the Command line, as is the default. This is useful for such edit commands as **ts** (text split) and **tf** (text flow).

Use the **KEYS** command to display the panel for modifying the current PF key definitions.

CUT and PASTE

The edit **CUT** and **PASTE** commands make use of multiple clipboards. On the command line, enter **CUT *'name'*** where *name* is a label you have chosen for the clipboard where the data is to be stored. Then select the lines you want to copy, or move, by using normal line commands (eg CC-CC/C/MM-MM/M). You can then repeat the exercise with more lines, cutting them to the same clipboard or to a different clipboard using a different *name*. Then you can paste the cut items wherever you want by using the **PASTE *'name'*** command and A(after)/B(before) line command. Omitting the *name* will select the 'DEFAULT' clipboard.

Other options available on the CUT/PASTE commands are:

- Use the **REPLACE** operand on the **CUT** command to replace the contents of the clipboard. Otherwise the lines are added to any lines already in the clipboard.
- With **CUT**, use the **X** or **NX** operands to copy only excluded or unexcluded lines to the clipboard.
- **CUT DISPLAY** will display the existing clipboards and allow you to rename them or browse their contents. When DISPLAY is specified, other **CUT** operands are ignored.
- On **PASTE**, use the **KEEP** keyword to copy the lines from the clipboard instead of moving them.
- The ISPF default clipboard is used unless a clipboard name is specified on the command.

In Edit settings (see EDSET command) you can set defaults for **CUT** and **PASTE** to indicate whether **CUT** should APPEND or REPLACE data and whether **PASTE** should DELETE or KEEP the data after use. To override the specified defaults use **CUT** APPEND/REPLACE and **PASTE** KEEP/DELETE.

& sign

When doing multiple **find** commands, or **change** commands etc, you can have ISPF retain the command on the command line by prefixing it with an **&** sign.

Excluding data lines – X, NX, F, L, S, and FLIP

Most people know the line command **X** (exclude) to exclude data lines from processing. It can also be used on the command line to perform the opposite function of the **FIND** command. For example:

```
X // 1 all
```

will exclude all lines containing the characters // in column 1, ie all JCL statements.

Exclude is also useful when used in conjunction with **FIND** commands, eg **F abcd X**, to locate the next occurrence of 'abcd', but only on an excluded line. Or the inverse, eg **F abcd NX**, to locate the next occurrence of 'abcd' but only on a non-excluded line. **X** and **NX** can be used in conjunction with the **DELETE** command, eg **DEL ALL X** to delete all excluded lines.

To show excluded lines, use the **F**, **L**, and **S** line commands. **F** will show the first line from a group of excluded lines, while **L** will show the last line. **F2** and **L5** will show the first 2 and last 5 respectively. The **S** command shows the most significant line(s) from a block of excluded lines. Data indentation is used to determine which lines will be shown and the lines with the left-most indentations are displayed. If several lines are indented equally, the first lines are shown, and, if the number used with an **S** is greater than the number of excluded lines, all of the excluded lines will be displayed.

FLIP: the FLIP command reverses the exclude status of a range of lines.

COPY/MOVE commands

The **COPY** command is used to copy one or more lines of data from a sequential dataset or member of a partitioned dataset, into the member or dataset currently being edited or viewed.

The **MOVE** command is used to move a partitioned dataset member or a sequential dataset. The external data is actually copied into the current data, and then the 'external' dataset or member is deleted.

From OS/390 2.10, both these commands will take a 'dataset(member)' parameter, eg **COPY *data-set(member)***. Standard TSO prefix/quotes rules apply to the dataset-name.

If you enter COPY or MOVE with no parameters you will be asked to specify the dataset name on the following edit/view copy panel.

CREATE/REPLACE commands

From OS/390 2.10, the target dataset for the Edit **CREATE** and **REPLACE** commands can be allocated automatically if it does not exist. A panel will be displayed asking whether the new dataset should inherit the attributes of the source dataset or allow you to specify the allocation attributes. This is the same as the feature described earlier for the Move/Copy utility (Option 3.3).

CREATE/REPLACE commands will take the 'dataset(member)' parameter, as described for the **copy**/**move** commands above.

View Replace

When the **REPLACE** command is used in **VIEW** to update the member being viewed, the confirmation panel shows whether the member has been updated by someone else during the **VIEW** session.

Using the **REPLACE** command to write data from a **VIEW** session can overwrite changes which were made to the dataset after the **VIEW** session began, because **VIEW** does not provide ENQ protection at the beginning of the **VIEW** session.

EDSET COMMAND

The Edit settings dialog can be displayed via the **EDSET** and **EDITSET** primary commands as well as from the Edit_Setting pulldown choice when editing data. This enables the user to change:

- The display of action bars in ISPF edit and view panels – removal gives an extra two lines of data in the edit window.
- The **CUT** and **PASTE** defaults (as described above).
- The line that Edit positions the target of a **FIND**, **CHANGE**, or **EXCLUDE** command.

- Whether or not the Editor always scrolls the target of a **FIND**, **CHANGE**, or **EXCLUDE** command to the target line specified, or only when the target line is not on the current display.
- The user session initial macro – a macro to be run whenever an edit session is started.
- Confirm Cancel/Move/Replace – whether or not a confirmation panel should be displayed whenever a **CANCEL**, **MOVE**, or **REPLACE** command is issued.
- Preserve VB record length – if the editor is to save trailing blanks for variable length files, or truncate to a single trailing blank.

SDSF

SDSF is an optional feature of OS/390. Information about SDSF commands and panels can be found online in SDSF HELP, via PF1, and in the manual *OS/390 SDSF Guide and Reference*.

Show job JCL

The line command **SJ** entered next to a job on the I, O, H, and DA panels will display the JCL for the job in an ISPF edit session. From here you can make changes and resubmit the JCL if you wish.

Edit job output

The line command **SE** entered next to a job or output dataset will display the output in an ISPF EDIT window. This allows you to use the full range of ISPF edit commands to manipulate the data – **exclude** and **find** commands are particularly useful when viewing job output. Any changes made are to the display only and are not saved.

As an alternative you can use **SB** to display the output in an ISPF BROWSE window.

Maximum return code

SDSF can display the maximum return code for each job. The column **Max-RC** on the H, O, and ST panels shows information about the maximum return code or abend code. This column allows you, easily,

to see whether a job has run without errors. By default, Max-RC is not displayed on the first screen and you will have to scroll right (PF11) to view this field, or alter the column order.

Next/previous dataset

When browsing the entire output of a job in SDSF you can jump to the start of the next dataset by using the **NEXT** command (abbreviated to **N**). If you are at the start of the job and want to bypass the JES output and go straight to the first 'real' output dataset enter '**N 3**' to go to the third dataset. N with a number will scroll the number of dataset specified in the number, so if you're already positioned at dataset 3 and you enter N 3, you will be taken to dataset 6. If the display is positioned at the bottom of the output you can get to the start of that dataset by entering **N**.

The reverse of the **NEXT** command is **PREV**, for previous. It can be abbreviated to **P** and works in the same way as **N** – a number is optional.

Sort order

The order of the items displayed on each SDSF screen varies depending on the content. For example, output jobs (H) are sorted with the earliest first, active jobs (DA) by jobname, etc. You can change the default sort order using the SORT primary command. The syntax is: **SORT field-name**

Column width

The width of columns in the SDSF displays can be changed from the default using the ARRANGE command. For example, **ARRANGE DEST 8** alters the width of the 'destination' column from 24 to 8 characters.

ARRANGE ? will display a pop-up scollable list of columns for the current panel. You can use this to alter the width of any column and also reorder the columns displayed.

Extra information

On most screens, extra information can be displayed by scrolling right using **PF11** (default).

Entering **?** on the command line will display the alternative field list, giving additional information.

Column order

The columns displayed on each SDSF screen can be customized so that the most useful columns appear in the first screen, ie without having to scroll right to view them. Select the **View** option from the action bar by positioning the cursor at the word View and pressing *Enter*. From the drop-down menu select **2. Arrange**. Select the fields required using /. Further information on how to do this can be found via PF1.

Processing multiple jobs – //

Instead of keying in the same command repeatedly on several jobs, SDSF allows you do a **block repeat** to process the same command on a range of jobs.

Type a // at the start of a block of rows and another at the end of the block of rows to process the block. Make all overtypes you want repeated throughout the block on either the first or last row of the block. You can overtype as many columns as are visible.

To repeat an action character, type a // at the start of a block of rows, followed by the action character. Then type // at the end of the block of rows to be processed. For example, **//p** will purge all jobs between there and the next // delimiter.

Displaying in hex

To view output in hexadecimal enter the command **SET HEX ON**. All output will now be displayed in hex. Enter **SET HEX OFF** to switch it off.

Display prefix, etc

You can use a number of commands, for example **PREFIX**, **OWNER**, **DEST**, to limit the jobs displayed on each screen and the **SORT** command can customize the order in which the jobs are displayed. The command **SET DISPLAY ON** displays the values of these and other fields, just above the scrollable area, as a visual reminder of the values currently in effect. **SET DISPLAY OFF** removes the line.

Running ISPF in batch

There may be occasions when you wish to run ISPF in batch. The following sample JCL can be used; just add a suitable job card and place the command to be executed on the ISPSTART command at the end of the example.

```
//*
//* DOC: ISPF IN BATCH
//*
//ISPF1 EXEC PGM=IKJEFT01,DYNAMNBR=100
//ISPPLIB  DD DISP=SHR,DSN=SYS1.ISPPLIB
//ISPPROF  DD DISP=(,DELETE),DSN=&&ISPPROF,
// LRECL=80,RECFM=FB,BLKSIZE=0,
// UNIT=SYSDA,SPACE=(TRK,(5,5,5))
//ISPMLIB  DD DISP=SHR,DSN=SYS1.ISPMLIB
//ISPSLIB  DD DISP=SHR,DSN=SYS1.ISPSLIB
//ISPTLIB  DD DISP=SHR,DSN=SYS1.ISPTLIB
//ISPTABL  DD DDNAME=ISPPROF
// DD DISP=SHR,DSN=SYS1.ISPTLIB
//SYSPROC  DD DISP=SHR,DSN=SYS1.ISPCLIB
//ISPLOG DD SYSOUT=*,
// DCB=(RECFM=VBA,LRECL=125,BLKSIZE=0)
//ISPLIST  DD DUMMY
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
ISPSTART CMD(PROF)
```

BOOKMANAGER

Bookshelves, Bookcases, and Collections

BookManager books are collected into Bookshelves. Generally there is a Bookshelf for each version of a product. Bookshelves are grouped into Bookcases or Collections. A Bookcase is a group of logically connected products.

Some books are more difficult to find than others. The following table lists some of the most common books by subject, bookcase, shelf, and book.

Utilities, eg IEBGENER, IEBCOPY etc – OS/390 Collection – DFSMS/MVS Bookshelf – DFSMSdfp utilities.

IDCAMS commands, eg DELETE, DEFINE, ALTER etc – OS/390

Collection – DFSMS/MVS Bookshelf – DFSMS Access Method Services for catalogs.

VSAM macro return and reason codes (aka feedback codes) – OS/390 Collection – DFSMS/MVS Bookshelf – DFSMS macro instructions for datasets.

MVS messages – OS/390 Collection – OS/390 MVS Bookshelf – OS/390 MVS System Messages Volumes 1-5.

JCL OS/390 Collection – OS/390 MVS Bookshelf – JCL Reference, JCL User's Guide.

LE (Language Environment) – OS/390 Collection – Language Environment Bookshelf –.

DB2 error messages – DB2 Collection – DB2 UDB for OS/390 Bookshelf – Messages and Codes.

Moira Hunter
Technical Consultant (UK

Call C functions from Assembler

PROBLEM ADDRESSED

In addition to the standard C runtime library, which contains many useful functions (for example, printf() to produce formatted output, sscanf() to parse an input string, and various mathematical functions), an increasing number of application-oriented functions exist written in C. This is problematic when legacy (non-C) programs need to invoke such functions because of C's non-standard calling convention. IBM's published method of invoking such C-language functions from Assembler involves three program levels:

1 A C language stub program that establishes the C environment and calls an Assembler-language program.

2 The Assembler-language program calls an intermediate C language program

3 The C language program calls the required C function

In most cases, this solution is needlessly complex, especially if more than one C function needs to be called. This article discusses the basic requirements that an Assembler language program must observe in order to directly call a C language function.

SOLUTION

An Assembler language program that directly calls a C language function must satisfy several conditions, indicating:

1 The Assembler language program must create or, if called from a Language Environment program, retain the Language Environment. The standard CEEENTRY macro provides this functionality.

2 The arguments passed to the C language function must observe the C language calling convention. This article describes this calling convention.

3 The Assembler language program must terminate using the CEETERM macro.

4 The Assembler language program uses the CEEPPA (program prologue area), CEEDSA (dynamic storage area), and CEECAA (common anchor area) macros to create the required data definitions.

C LANGUAGE CALLING CONVENTION

The C language calling convention is based on standard MVS calling conventions:

1 Register 15 contains the address of the called function.

2 Register 14 contains the return address to the calling program.

3 Register 1 points to the argument list. Unless explicitly specified otherwise in the function prototype, the argument list contains the address of a passed string and the contents of other argument types. Depending on the form of the return value, the argument list can also contain the address of the return value.

For example, the C statement:


```
memchr(pStr,' ',6);
```

has the following equivalent call in Assembler:

```
BAL   1,*+16
DC    AL4(PSTR),AL3(Ø),C' ',FL4'6'
L     15,=V(MEMCHR)
BALR  14,15
```

Re-entrant program

If a program is to be re-entrant, all variables must be defined in the Dynamic Storage Area (CEEDSA macro) and the correct macro form (MF=E or MF=L) used when appropriate. Any required variable initializations must be performed at runtime.

Dynamic storage area

The CEEDSA macro defines the dynamic storage area that is allocated at runtime. Each instance is assigned its own dynamic storage area. The AUTO keyword of the CEEENTRY macro specifies the length of the dynamic storage area.

ARGUMENT LIST

The argument list contains either copies of simple data types (eg short or char data types), the addresses of derived data types (eg strings, char[]) or the explicit addresses of data fields (& operator). The first word of the argument list contains the address of the return value when its type is not a simple data type or a pointer.

PROGRAM ENTRY

The CEEENTRY macro generates the appropriate entry code for the program (allocate dynamic storage area, chain save-areas, etc). The MAIN keyword specifies whether the program is a main program (ie create the LE environment).

PROGRAM EXIT

The CEETERM macro generates the appropriate exit code for the program (invoke the termination routine, pass the specified return code back to the calling program).

C function return value

Depending on its form, the return value (function value) is returned in one of three ways:

- In register 15 (for a simple data type, eg short or char).
- Register 15 contains the address of the return value (*, pointer).
- In the field whose address is contained in the first word of the argument list (eg float or double).

Standard C functions that do not return a simple data type (eg short or char) or a pointer (eg (atof(), ldiv()) use their own convention (not described in this article).

C inline functions

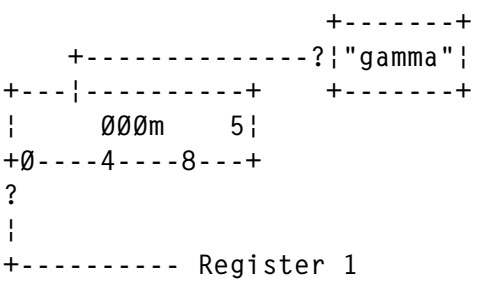
Although many short C standard functions (eg memset()) are realized as inline code, equivalent external functions also exist in the runtime library. Such external functions are less efficient because of the linkage and non-optimized code.

Example 1:

```
char string = "gamma", *pc;

pc = memchr(string, 'm', 5);
```

has the following argument list:



The fields shown above have the following representations:

```
000m X'00 00 00 94' = 3X'0',C'm'
5    X'00 00 00 05' = F'5'
"gamma"   C'gamma'
```

Note: in this particular case, the ‘gamma’ string does not need the terminating X'0'.

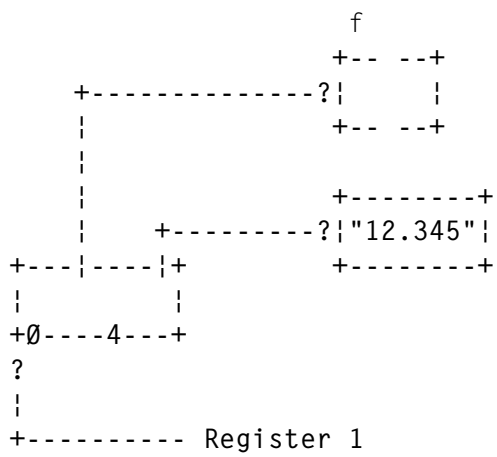
Example 2:

```
float MYATOF(char *); /* function prototype */

float f;

f = MYATOF("12.345"); /* function invocation */
```

This function's invocation has the following argument list:



The '12.345' field is actually represented as C'12.345',X'00'

Sample programs

The following section shows some trivial sample programs that serve to illustrate the mechanics involved and do not, in themselves, represent practical applications. However, the final program shows how an Assembler program can use C runtime functions in a practical application.

Sample program 1

```
TESTCEE1 CEEENTRY PPA=MAINPPA,MAIN=YES,AUTO=DSASIZE
         SPACE 1
** printf("%s %Ø5d",str,int);
         LA    1,ARGLIST   set address of argument list
         L     Ø,INT
         ST    Ø,8(1)      store INT in argument list
         L     15,=V(PRINTF)
         BALR  14,15
* set program return code as printed length
         ST    15,RTC
         SPACE
EXIT     CEETERM RC=RTC,MF=(E,TERMLIST)
         SPACE 2
MASK     DC    C'%s %Ø5d',AL1(NULL)
```

```
STR      DC    C'My string',AL1(NULL)
INT      DC    F'77'
**
MAINPPA  CEEPPA            program prolog area
         CEEDSA            dynamic storage area
TERMLIST CEETERM MF=L
*
ARGLIST  DS    ØA
         DC    A(MASK)
         DC    A(STR)
         DS    F
*
RTC      DS    F           return code
DSASIZE  EQU   *-CEEDSA
         CEECAA            common anchor area
         END
```

This program shows how the printf() function can be invoked to produce formatted output; it produces the following output

```
My string ØØ77
```

Sample program 2

```
TESTCEE2 CEEENTRY PPA=MAINPPA,MAIN=YES,AUTO=DSASIZE
         SPACE 1
**  void *memchr(char *pc, int ch, size_t count);
         LA    1,ARGLIST1
         L     15,=V(MEMCHR)
         BALR  14,15
         ST    15,PTR
** display returned string: mma lamda
         LA    1,ARGLIST
         L     15,=V(PUTS)
         BALR  14,15
         SPACE 2
**  char *strrchr(char *pc, int ch);
         LA    1,ARGLIST2
         L     15,=V(STRRCHR)
         BALR  14,15
         ST    15,PTR   set returned pointer in argument list
** display returned substring: mda
         LA    1,ARGLIST
         L     15,=V(PUTS)
         BALR  14,15
         SPACE 2
         CEETERM RC=Ø
         SPACE 2
ARGLIST1 DS    ØA
         DC    A(STR)
         DC    XL3'Ø'
```

```
         DC    C'm'
COUNT    DC    FL4'11'
         SPACE 1
ARGLIST2 DS    ØA
         DC    A(STR)
         DC    XL3'Ø'
         DC    C'm'
         SPACE 1
**
NULL     EQU   X'ØØ'
**
STR      DC    C'gamma lamda',AL1(NULL)
CH       DC    C'm'
**
MAINPPA  CEEPPA               program prolog area
         CEEDSA               dynamic storage area
ARGLIST  DS    ØA
PTR      DS    A
         SPACE 1
DSASIZE  EQU   *-CEEDSA
         CEECAA               common anchor area
         END
```

This Assembler program corresponds to the following C language program:

```
#include <string.h>
#include <stdio.h>

char *pc;
char str[] = "gamma lamda";
char ch = 'm';
int count = 11;

main()
{
  pc = memchr(str, ch, count);
  puts(pc); /* display returned substring: mma lamda */
  pc = strrchr(str, ch);
  puts(pc); /* display returned substring: mda */
}
```

Sample program 3:

```
TESTCEE3 CEEENTRY PPA=MAINPPA,MAIN=YES,AUTO=DSASIZE
         SPACE 1
         MVC   RTC,=F'1'      preset return code for fopen() error
** fp = fopen("DD:DD1","r");
         LA    1,ARGLIST
         LA    Ø,PARM1        C'DD:DD1',AL1(NULL)
         ST    Ø,Ø(1)         address of first argument
```

```
         LA    Ø,PARM2         C'r',AL1(NULL)
         ST    Ø,4(1)          address of second argument
         L     15,=V(FOPEN)
         BALR  14,15
         ST    15,FP           store file pointer
         LTR   15,15           test for error return
         JZ    EXIT
** pstr = fgets(record,sizeof(record),fp);
         MVC   RTC,=F'2'       preset return code (for error return)
READLOOP LA    1,ARGLIST
         LA    Ø,RECORD
         ST    Ø,Ø(1)
         MVC   4(4,1),=A(L'RECORD)
         L     Ø,FP
         ST    Ø,8(1)
         L     15,=V(FGETS)
         BALR  14,15
         ST    15,PSTR
         LTR   15,15           test for error return
         JZ    EXIT
** puts(record)
         MVC   RTC,=F'Ø'       reset return code
         LA    1,ARGLIST
         LA    Ø,RECORD
         ST    Ø,Ø(1)
         L     15,=V(PUTS)
         BALR  14,15
**
         ICM   15,15,PSTR      test for EOF
         JNZ   READLOOP        no, read next record
**
EXIT     CEETERM RC=RTC,MF=(E,TERMLIST)
         SPACE 2
NULL     EQU   X'Ø'
         SPACE 2
PARM1    DC    C'DD:DD1',AL1(NULL)
PARM2    DC    C'r',AL1(NULL)
         SPACE 2
MAINPPA  CEEPPA                program prolog area
         CEEDSA                dynamic storage area
TERMLIST CEETERM MF=L
         SPACE 1
ARGLIST  DS    3A
         SPACE 1
PSTR     DS    A               string pointer
FP       DS    A               file handle pointer
         SPACE 1
RTC      DS    F               return code
         SPACE 1
RECORD   DS    CL8Ø,X
DSASIZE  EQU   *-CEEDSA
```

```
         CEECAA ,              common anchor area
         END
```

This program shows how an Assembler program can use C runtime functions to perform input/output.

The Assembler program corresponds generally to the following C language fragment:

```
FILE *fp;
char record[81];
char *pstr;

int main() {
  fp = fopen("dd:dd1","r");
  if (fp == 0) exit(1); /* fopen() error */
  do {
    pstr = fgets(record,sizeof(record),fp);
  } while (pstr != 0);
  return 0;
}
```

Practical program

There are some situations where it is opportune to use C functions to provide Assembler programs with functionality not available with standard operating system services, for example, sprintf() to format a string, sscanf() to parse a string. For example, the following TRANSFRM subprogram performs a locale-based transformation on the passed field. The transformed string can then be used to properly collate accented strings (eg Müller). The Assembler function avoids the extensive initialization required for a C program. The Assembler function uses the standard SETLOCALE() and STRXFRM() C-runtime functions to perform the required conversion.

Note: the MAIN=NO option for the CEEENTRY macro in this example specifies that the TRANSFRM subprogram is called from within the Language Environment.

```
* Perform locale-based string transformation
* Call SETLOCALE() and STRXFRM() to perform a string transformation
* using the De_DE.IBM-1141 locale.
TRANSFRM CEEENTRY MAIN=NO,AUTO=DSASIZE
         SPACE 1
         LM    2,5,0(1)    pointers to parameters
* R2: A(input field)
* R3: L(input field)
```

```
* R4: A(output field)
* R5: L(output field)
         SPACE 1
* convert input field to C-string in intermediate area
         LA    14,INSTR
         LA    15,L'INSTR
         MVCL  14,2
         MVI   Ø(14),NULL       append X'ØØ'-delimiter
         SPACE 1
**       CALL  SETLOCAL(LC_ALL,PLOCNAME)
         MVC   RTC,=F'-2'       preset return code: SETLOCALE() error
         LA    1,ARGLIST1       address of argument list
         L     15,=V(SETLOCAL)  note: name truncated to 8 characters
         BASR  14,15
         LTR   15,15            test SETLOCALE() return code
         JZ    EXIT             locale could not be set
**       CALL  STRXFRM,(OUTSTR,INSTR,COUNT)
         LA    1,ARGLIST2       address of argument list
         LA    Ø,OUTSTR
         ST    Ø,Ø(1)           set argument 1
         LA    Ø,INSTR
         ST    Ø,4(1)           set argument 2
         MVC   COUNT,=A(L'OUTSTR) maximum length
         L     15,=V(STRXFRM)
         BASR  14,15
         ST    15,RTC           set return code as length
         SPACE 1
         CR    15,5             converted string overflow?
         JNH   LENGTHOK         :no
         MVC   RTC,=F'-1'       overflow
         LR    15,5             truncate
LENGTHOK LA    14,OUTSTR
         LR    5,15             actual length
         MVCL  4,14             return to caller
         SPACE 1
EXIT     CEETERM RC=RTC         terminate
         SPACE 2
LC_ALL   EQU   -1
NULL     EQU   Ø
         SPACE 1
ARGLIST1 DS    ØA               argument list 1
         DC    A(LC_ALL)
         DC    A(LOCNAME)
**
LOCNAME  DC    C'De_DE.IBM-1141',X'Ø'
**
PPA      CEEPPA ,               program prolog area
         CEEDSA ,               dynamic storage area
OUTSTR   DS    CL256            output string
INSTR    DS    CL256,X          input string + delimiter
```

```
RTC      DS    F                 return code
         SPACE 1
ARGLIST2 DS    ØA                argument list 2
         DS    A(OUTSTR,INSTR)
COUNT    DS    F
DSASIZE  EQU   *-CEEDSA
         CEECAA ,                common anchor area
         END
```

SAMPLE TEST PROGRAMS

The two sample test programs show how the TRANSFRM subprogram can be called from a C program:

```
int TRANSFRM(const char *pistr, const int *listr,
             char const *postr, const int *lostr);
#include <stdio.h>
#include <string.h>
int ilen, olen;
char istr[] = "Müller";
char ostr[256];
void main()
{
 ilen = strlen(istr);
 olen = sizeof(ostr);
 /* olen = TRANSFRM(istr, strlen(istr), ostr, sizeof(ostr)); */
 olen = TRANSFRM(istr, &ilen, ostr, &olen);
 printf("olen: %d\n",olen);
}
```

and a COBOL program:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. TRANSFRA.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
Ø1 ILEN PIC 9(8) BINARY VALUE 6.
Ø1 OLEN PIC 9(8) BINARY VALUE 256.
Ø1 ISTR PIC X(6) VALUE 'Müller'.
Ø1 OSTR PIC X(256).
PROCEDURE DIVISION.
    CALL 'TRANSFRM' USING ISTR, ILEN, OSTR, OLEN END-CALL
    DISPLAY 'RESULT:' RETURN-CODE
    STOP RUN.
```

The Initialization Parameter Area

The Initialization Parameter Area (IPA) became available in OS/390 V1R2. It is mapped by the IHAIPA macro in SYS1.MACLIB and contains initialization parameters defined in:

1 The load parameter used to IPL.

2 The LOAD*xx* member used to IPL.

3 All IEASYS*xx* members used to IPL.

The following REXX EXEC can be used to display the information in the IPA and also display other useful system information like subsystems, CPU information, operating system software version levels, system symbols, page dataset usage, storage sizes, and a virtual storage map. If executed from ISPF, the display will be put in a scrollable browse dataset. This EXEC can also be executed from Unix System Services or a Unix System Services Web server. Various execution options are supported to show only subsets of all the displays.

```
/* REXX */
/* AUTHOR: Mark Zelden                    */
/* Trace ?r */
/*********************************************************************/
/* DISPLAY SYSTEM INFORMATION ON TERMINAL                            */
/*********************************************************************/
/* EXECUTION SYNTAX:                                                 */
/* TSO %SYSINFO <option>                                             */
/* VALID OPTIONS ARE 'ALL' (default), 'IPL', 'VERSION', 'STOR',      */
/*                   'CPU', 'IPA', 'SYMBOLS', 'VMAP', 'PAGE', and    */
/*                   'SUB'.                                          */
/* Examples:                                                         */
/*  TSO %SYSINFO         (Display all information)                   */
/*  TSO %SYSINFO VMAP    (Display a Virtual Storage Map)             */
/*  TSO %SYSINFO SYMBOLS (Display Static System Symbols)             */
/*  TSO %SYSINFO SUB     (Display Subsystem Information)             */
/*                                                                   */
/* Sample Unix System Services WEB Server execution via links:       */
/*  <a href="/cgi-bin/SYSINFO">MVS Information</a>                   */
/*  <a href="/cgi-bin/SYSINFO?vmap">Virtual Storage Map</a>          */
/*  <a href="/cgi-bin/SYSINFO?symbols">Static System Symbols</a>     */
/*  <a href="/cgi-bin/SYSINFO?sub">Subsystem Information</a>         */
/*********************************************************************/
Arg OPTION
Parse source . . . . . ENV . .
```

```
If ENV <> 'OMVS' then                          /* are we under unix ?  */
  If Sysvar(SYSISPF)='ACTIVE' then do          /* no, is ISPF active?  */
    Address ISREDIT "MACRO (OPTION)"    /* YES - allow use as macro  */
    OPTION = Translate(OPTION)   /* ensure upper case for edit macro */
    Address ISPEXEC "VGET ZENVIR"              /* ispf version        */
    SYSISPF = 'ACTIVE'                         /* set SYSISPF = ACTIVE */
  End
If OPTION <> 'IPL'     & OPTION <> 'VERSION' & OPTION <> 'STOR'    &,
   OPTION <> 'CPU'     & OPTION <> 'IPA'     & OPTION <> 'SYMBOLS' &,
   OPTION <> 'VMAP'    & OPTION <> 'PAGE'    & OPTION <> 'SUB' ,
   then OPTION = 'ALL'
Numeric digits 1Ø                             /* dflt of 9 not enough  */
Call HEADING                                  /* Heading sub-routine   */
Call COMMON            /* control blocks needed by multiple routines */
If OPTION = 'ALL' then do
  Call IPL                                    /* IPL information       */
  Call VERSION                                /* Version information   */
  Call STOR                                   /* Storage information   */
  Call CPU                                    /* CPU information       */
  Call IPA                                    /* Initialization info.  */
  Call SYMBOLS                                /* Symbols information   */
  Call VMAP                                   /* Virt. Storage Map     */
  Call PAGE                                   /* Page DSN information  */
  Call SUB                                    /* Subsystem information */
End
Else interpret call OPTION
/*********************************************************************/
/* Done looking at all control blocks                                */
/*********************************************************************/
Queue ''  /* null queue to end stack   */
/*********************************************************************/
/* If ISPF is active, browse output - otherwise write to the terminal*/
/*********************************************************************/
If SYSISPF = 'ACTIVE' then call BROWSE_ISPF  /* ISPF active?         */
Else do queued()                              /* ISPF is not active   */
  Parse pull line                             /* pull queued lines    */
  Say line                                    /* say lines            */
End /* else do  */
Exit Ø                                        /* End SYSINFO - RC Ø   */
/*********************************************************************/
/*  End of main SYSINFO code                                         */
/*********************************************************************/
/*  Start of sub-routines                                            */
/*********************************************************************/
HEADING:             /* Heading sub-routine                          */
If ENV = 'OMVS' then do                     /* Are we under OMVS?    */
  Do CKWEB = __ENVIRONMENT.Ø to 1 by -1     /* check env. vars       */
     If pos('HTTP_',__ENVIRONMENT.CKWEB) <> Ø then do  /* web server */
       Say 'Content-type: text/html'
       Say ''
       Say '<title>SYSINFO</title>'
```

```
       Say '<meta name="author" content="Mark Zelden">'
       Say '<meta name="description" content="' || ,
           'SYSINFO -' OPTION 'option.' ,
           'Last updated on' LASTUPD ||'. Written by' ,
           'Mark Zelden.">'
       Say '<meta http-equiv="pragma" content="no-cache">'
       Say '<body BGCOLOR="#000000" TEXT="#00FFFF">'
       Say '<pre>'
       Leave                                /* exit loop             */
     End /* if pos */
  End /* do CKWEB */
End
Call RDATE TODAY                           /* call RDATE sub-routine */
DAY      = Word(RESULT,3)                  /* weekday from RDATE     */
DATE     = Substr(RESULT,1,10)             /* date as MM/DD/YYYY     */
JUL      = Substr(RESULT,7,8)              /* date as YYYY.DDD       */
CURNNNNN = Substr(RESULT,16,5)             /* date as NNNNN          */
Queue '************************************************************' || ,
      '************************'
Queue '************************  SYSTEM INFORMATION  **********' || ,
      '************************'
Queue '************************************************************' || ,
      '************************'
Queue ' '
Queue 'Today is 'DAY DATE '('JUL'). The local time is 'TIME()'.'
return
COMMON:              /* Control blocks needed by multiple routines   */
CVT      = C2d(Storage(10,4))                /* point to CVT         */
PRODNAME = Storage(D2x(CVT - 40),7)          /* point to mvs version */
If Substr(PRODNAME,3,1) > 3 then
  ECVT     = C2d(Storage(D2x(CVT + 140),4))  /* point to CVTECVT     */
FMIDNUM  = Storage(D2x(CVT - 32),7)          /* point to fmid        */
JESCT    = C2d(Storage(D2x(CVT + 296),4))    /* point to JESCT       */
JESPJESN = Storage(D2x(JESCT + 28),4)        /* name of primary JES  */
CSD      = C2d(Storage(D2x(CVT + 660),4))    /* point to CSD         */
SMCA     = Storage(D2x(CVT + 196),4)         /* point to SMCA        */
SMCA     = Bitand(SMCA,'7FFFFFFF'x)          /* zero high order bit  */
SMCA     = C2d(SMCA)                         /* convert to decimal   */
MODEL    = C2d(Storage(D2x(CVT - 6),2))      /* point to cpu model   */
/*********************************************************************/
/*  The CPU model is stored in packed decimal format with no sign,   */
/*  so to make the model printable, it needs to be converted back    */
/*  to hex.                                                          */
/*********************************************************************/
MODEL    = D2x(MODEL)                        /* convert back to hex  */
If Substr(FMIDNUM,4,4) >= 6602 then do
  ECVTIPA  = C2d(Storage(D2x(ECVT + 392),4)) /* point to IPA         */
  IPASCAT  = Storage(D2x(ECVTIPA + 224),63)  /* SYSCAT  card image   */
End
If Substr(FMIDNUM,4,4) >  6609 then ,        /* OS/390 R10 or above  */
  IPAARCHL = Storage(D2x(ECVTIPA + 2143),1)  /* ARCHLVL (1 or 2)     */
```

```
Return
IPL:                  /* IPL information sub-routine                  */
Queue ' '
/*********************************************************************/
/*  The IPL date is stored in packed decimal format - so to make     */
/*  the date printable, it needs to be converted back to hex and     */
/*  the packed sign needs to be removed.                             */
/*********************************************************************/
IPLTIME  = C2d(Storage(D2x(SMCA + 336),4))   /* IPL Time - binary    */
IPLDATE  = C2d(Storage(D2x(SMCA + 340),4))   /* IPL Date - 0CYYDDDF  */
If IPLDATE  >= 16777231 then do              /*          is C = 1 ?  */
  IPLDATE  = D2x(IPLDATE)                    /* convert back to hex  */
  IPLDATE  = Substr(IPLDATE,2,5)             /* keep YYDDD           */
  IPLDATE  = '20'IPLDATE                     /* use 21st century date*/
End
Else do
  IPLDATE  = D2x(IPLDATE)                    /* convert back to hex  */
  IPLDATE  = Left(IPLDATE,5)                 /* keep YYDDD           */
  IPLDATE  = '19'IPLDATE                     /* use 20th century date*/
End
IPLYYYY  = Substr(IPLDATE,1,4)               /* YYYY portion of date */
IPLDDD   = Substr(IPLDATE,5,3)               /* DDD  portion of date */
Call RDATE IPLYYYY IPLDDD                    /* call RDATE subroutine*/
IPLDAY   = Word(RESULT,3)                    /* weekday from RDATE   */
IPLDATE  = Substr(RESULT,1,10)               /* date as MM/DD/YYYY   */
IPLJUL   = Substr(RESULT,7,8)                /* date as YYYY.DDD     */
IPLNNNNN = Substr(RESULT,16,5)               /* date as NNNNN        */
IPLHH    = Right(IPLTIME%100%3600,2,'0')     /* IPL hour             */
IPLMM    = Right(IPLTIME%100//3600%60,2,'0') /* IPL minute           */
IPLSS    = Right(IPLTIME%100//60,2,'0')      /* IPL seconds          */
IPLTIME  = IPLHH':'IPLMM':'IPLSS             /* time in HH:MM:SS     */
/*                                                                   */
ASMVT    = C2d(Storage(D2x(CVT + 704),4))    /* point to ASMVT       */
CLPABYTE = Storage(D2x(ASMVT + 1),1)         /* point to CLPA byte   */
CHKCLPA  = Bitand(CLPABYTE,'8'x)             /* check for B'1000'    */
CHKCLPA  = C2d(CHKCLPA)                      /* convert to decimal   */
If CHKCLPA < 8 then IPLCLPA = 'with CLPA'    /* bit off - CLPA       */
  Else IPLCLPA = 'without CLPA'              /* bit on  - no CLPA    */
RESUCB   = C2d(Storage(D2x(JESCT + 4),4))    /* point to SYSRES UCB  */
IPLVOL   = Storage(D2x(RESUCB + 28),6)       /* point to IPL volume  */
If Substr(PRODNAME,3,1) < 5 then ,
  IPLADDR  = Storage(D2x(RESUCB + 13),3)     /* point to IPL address */
Else do
  CVTSYSAD = C2d(Storage(D2x(CVT + 48),4))   /* point to UCB address */
  IPLADDR  = Storage(D2x(CVTSYSAD + 4),2)    /* point to IPL UCB     */
  IPLADDR  = C2x(IPLADDR)                    /* convert to EBCDIC    */
End
GRSNAME  = Storage(D2x(CVT + 340),8)         /* point to system name */
GRSNAME  = Strip(GRSNAME,T)                  /* del trailing blanks  */
SMFNAME  = Storage(D2x(SMCA + 16),4)         /* point to SMF name    */
SMFNAME  = Strip(SMFNAME,T)                  /* del trailing blanks  */
If Substr(FMIDNUM,4,4) <  6604 then do       /* use CAXWA B4 OS390R4 */
```

```
  AMCBS    = C2d(Storage(D2x(CVT + 256),4))  /* point to AMCBS       */
  ACB      = C2d(Storage(D2x(AMCBS + 8),4))  /* point to ACB         */
  CAXWA    = C2d(Storage(D2x(ACB + 64),4))   /* point to CAXWA       */
  MCATDSN  = Storage(D2x(CAXWA + 52),44)     /* master catalog dsn   */
  MCATDSN  = Strip(MCATDSN,T)                /* remove trailing blnks*/
  MCATUCB  = C2d(Storage(D2x(CAXWA + 28),4)) /* point to mcat UCB    */
  MCATVOL  = Storage(D2x(MCATUCB + 28),6)    /* master catalog VOLSER*/
End
Else do                                      /* OS/390 R4 and above  */
  MCATDSN  = Strip(Substr(IPASCAT,11,44))    /* master catalog dsn   */
  MCATVOL  = Substr(IPASCAT,1,6)             /* master catalog VOLSER*/
End
Queue 'The last IPL was 'IPLDAY IPLDATE '('IPLJUL')' ,
      'at 'IPLTIME' ('CURNNNNN - IPLNNNNN' days ago).'
Queue 'The IPL was done 'IPLCLPA'.'
Queue 'The system IPL address was 'IPLADDR' ('IPLVOL').'
If Substr(PRODNAME,3,1) > 3 then do
  ECVTSPLX = Storage(D2x(ECVT+8),8)          /* point to SYSPLEX name*/
  ECVTLOAD = Storage(D2x(ECVT+160),8)        /* point to LOAD PARM   */
  IPLPARM  = Strip(ECVTLOAD,T)               /* del trailing blanks  */
  SEPPARM  = Substr(IPLPARM,1,4) Substr(IPLPARM,5,2),
             Substr(IPLPARM,7,1) Substr(IPLPARM,8,1)
  SEPPARM  = Strip(SEPPARM,T)                /* del trailing blanks  */
  Queue 'The IPL LOAD PARM used was 'IPLPARM' ('SEPPARM').'
  If Substr(FMIDNUM,4,4) >= 5520 then do
    ECVTHDNM = Storage(D2x(ECVT+336),8)      /* point to hardware nam*/
    ECVTLPNM = Storage(D2x(ECVT+344),8)      /* point to LPAR name   */
    If Substr(FMIDNUM,4,4) >  6609 then do   /* OS/390 R10 or above  */
      If IPAARCHL <> 2 then ,                /* ESA/390 mode         */
        Queue 'The system is running in ESA/390 mode (ARCHLVL = 1).'
      Else ,
        Queue 'The system is running in z/Architecture mode' ,
              '(ARCHLVL = 2).'
    End /* if Substr(FMIDNUM,4,4) >  6609 */
    If ECVTHDNM <> ' ' & ECVTLPNM <> ' ' then do
      CSDPLPN  = C2d(Storage(D2x(CSD + 252),1)) /* point to LPAR #   */
      Queue 'The Processor name is' Strip(ECVTHDNM)'.' ,
            'The LPAR name is' Strip(ECVTLPNM)' (LPAR #'CSDPLPN').'
    End
    Else if ECVTHDNM <> ' ' then ,
      Queue 'The Processor name is' Strip(ECVTHDNM)'.'
    Queue 'The SYSPLEX name is' Strip(ECVTSPLX)'.'
  End  /* if Substr(FMIDNUM,4,4) >= 5520*/
End
Queue 'The GRS system id is 'GRSNAME'. The SMF system id is 'SMFNAME'.'
If Substr(PRODNAME,3,1) < 5 then do
  IOCON    = Storage(D2x(CVTEXT2 + 6),2)       /* HCD IODFxx or MVSCP*/
                                               /* IOCONFIG ID=xx     */
  Queue 'The currently active IOCONFIG or HCD IODF is 'IOCON'.'
End
Else do
```

```
  CVTIXAVL = C2d(Storage(D2x(CVT+124),4))      /* point to IOCM      */
  IOCIOVTP = C2d(Storage(D2x(CVTIXAVL+208),4)) /* pt to IOS Vect Tbl */
  CDA      = C2d(Storage(D2x(IOCIOVTP+24),4))  /* point to CDA       */
  IODF     = Storage(D2X(CDA+32),44)           /* point to IODF name */
  IODF     = Strip(IODF,T)                     /* del trailing blanks*/
  CONFIGID = Storage(D2X(CDA+92),8)            /* point to CONFIG    */
  EDT      = Storage(D2X(CDA+104),2)           /* point to EDT       */
  IOPROC   = Storage(D2X(CDA+124),8)           /* point to IODF Proc */
  IODATE   = Storage(D2X(CDA+156),8)           /* point to IODF date */
  IOTIME   = Storage(D2X(CDA+164),8)           /* point to IODF time */
  IODESC   = Storage(D2X(CDA+172),16)          /* point to IODF desc */
  Queue 'The currently active IODF dataset is 'IODF'.'
  Queue '  Configuration ID =' CONFIGID ' EDT ID =' EDT
  If Substr(IOPROC,1,1) <> '00'x  & ,
     Substr(IOPROC,1,1) <> '40'x then do       /* is token there?    */
    Queue '  TOKEN: Processor  Date      Time      Description'
    Queue '         'IOPROC'   'IODATE'  'IOTIME'  'IODESC
  End
End
Queue 'The Master Catalog is 'MCATDSN' on 'MCATVOL'.'
/*If OPTION = 'IPL' then interpret call 'VERSION' */ /* incl version*/
Return
VERSION:             /* Version information sub-routine              */
Queue ' '
Call SUB FINDJES  /* call SUB routine with FINDJES option            */
If JESPJESN = 'JES3' then do                 /* Is this JES3?        */
  If ENV = 'OMVS' then do  /* running under Unix System Services     */
    JES3FMID = Storage(D2x(JESSSVT+644),8)      /* JES3 FMID         */
    Select  /* determine JES3 version from FMID  */
      When JES3FMID = 'HJS5521' then JESLEV = 'SP 5.2.1'
      When JES3FMID = 'HJS6601' then JESLEV = 'OS 1.1.0'
      When JES3FMID = 'HJS6604' then JESLEV = 'OS 2.4.0'
      When JES3FMID = 'HJS6606' then JESLEV = 'OS 2.6.0'
      When JES3FMID = 'HJS6608' then JESLEV = 'OS 2.8.0'
      When JES3FMID = 'HJS6609' then JESLEV = 'OS 2.9.0'
      When JES3FMID = 'HJS7703' then JESLEV = 'OS 2.10'
      Otherwise JESLEV = JES3FMID /* if not in tbl, use FMID as ver  */
    End /* select */
    JESNODE  = '*not_avail*'                 /* can't do under USS   */
  End /* if env = 'omvs' */
  Else do /* if not running under Unix System Services, use TSO VARs */
    JESLEV   = SYSVAR(SYSJES)                /* TSO/E VAR for JESLVL */
    JESNODE  = SYSVAR(SYSNODE)               /* TSO/E VAR for JESNODE*/
  End
End
Else do  /* JES2 */
  JESLEV   = Strip(Storage(D2x(JESSUSE),8))  /* JES2 Version         */
  Select
    When Substr(JESLEV,1,6) == 'OS 1.1' | , /* OS/390 1.1  or        */
      Substr(JESLEV,1,4) == 'SP 5' then ,    /* ESA V5 JES2          */
      JESNODE  = Strip(Storage(D2x(JESSUS2+336),8)) /*   JES2 NODE   */
```

```
    When Substr(JESLEV,1,6) == 'OS 2.1' then, /* OS/390 2.10 and >   */
      JESNODE  = Strip(Storage(D2x(JESSUS2+452),8)) /* JES2 NODE     */
    When Substr(JESLEV,1,5) == 'OS 1.' | ,   /* OS/390 1.2           */
      Substr(JESLEV,1,5) == 'OS 2.' then,    /*  through OS/390 2.9  */
      JESNODE  = Strip(Storage(D2x(JESSUS2+372),8)) /* JES2 NODE     */
    Otherwise ,                              /* Lower than ESA V5    */
      JESNODE  = SYSVAR(SYSNODE)             /* TSO/E VAR for JESNODE*/
  End  /* select */
End /* else do */
CVTRAC   = C2d(Storage(D2x(CVT + 992),4))    /* point to RACF CVT    */
RCVTID   = Storage(D2x(CVTRAC),4)            /* point to RCVTID      */
                                             /* RCVT, ACF2, or RTSS  */
If RCVTID = 'RCVT' then RCVTID = 'RACF'      /* RCVT is RACF         */
If RCVTID = 'RTSS' then RCVTID = 'Top Secret'  /* RTSS is Top Secret */
RACFVRM  = Storage(D2x(CVTRAC + 616),4)      /* RACF Ver/Rel/Mod     */
RACFVER  = Substr(RACFVRM,1,1)               /* RACF Version         */
RACFREL  = Substr(RACFVRM,2,2)               /* RACF Release         */
RACFREL  = Format(RACFREL)                   /* Remove leading 0     */
RACFMOD  = Substr(RACFVRM,4,1)               /* RACF MOD level       */
RACFLEV  = RACFVER || '.' || RACFREL || '.' || RACFMOD
CVTDFA   = C2d(Storage(D2x(CVT + 1216),4))   /* point to DFP ID table*/
DFAPROD  = C2d(Storage(D2x(CVTDFA +16),1))   /* point to relese byte */
If DFAPROD = 0 then do                       /* DFP not DF/SMS       */
  DFAREL   = C2x(Storage(D2x(CVTDFA+2),2))   /* point to DFP release */
  DFPVER   = Substr(DFAREL,1,1)              /* DFP Version          */
  DFPREL   = Substr(DFAREL,2,1)              /* DFP Release          */
  DFPMOD   = Substr(DFAREL,3,1)              /* DFP Mod Lvl          */
  DFPRD    = 'DFP'                           /* product is DFP       */
  DFLEV    = DFPVER || '.' || DFPREL || '.' || DFPMOD
End
Else do                                      /* DFSMS not DFP        */
  DFARELS  = C2x(Storage(D2x(CVTDFA+16),4))  /* point to DF/SMS rel  */
  DFAVER   = X2d(Substr(DFARELS,3,2))        /* DF/SMS Version       */
  DFAREL   = X2d(Substr(DFARELS,5,2))        /* DF/SMS Release       */
  DFAMOD   = X2d(Substr(DFARELS,7,2))        /* DF/SMS Mod Lvl       */
  DFPRD    = 'DFSMS'                         /* product is DF/SMS    */
  DFLEV    = DFAVER || '.' || DFAREL || '.' || DFAMOD
End
CVTTVT   = C2d(Storage(D2x(CVT + 156),4))    /* point to TSO vect tbl*/
TSVTLVER = Storage(D2x(CVTTVT+100),1)        /* point to TSO Version */
TSVTLREL = Storage(D2x(CVTTVT+101),2)        /* point to TSO Release */
TSVTLREL = Format(TSVTLREL)                  /* Remove leading 0     */
TSVTLMOD = Storage(D2x(CVTTVT+103),1)        /* point to TSO Mod Lvl */
TSOLEV   = TSVTLVER || '.' || TSVTLREL || '.' || TSVTLMOD
CVTEXT2  = C2d(Storage(D2x(CVT + 328),4))    /* point to CVTEXT2     */
CVTATCVT = C2d(Storage(D2x(CVTEXT2 + 65),3)) /* point to VTAM AVT    */
ISTATCVT = C2d(Storage(D2x(CVTATCVT + 0),4)) /* point to VTAM CVT    */
ATCVTLVL = Storage(D2x(ISTATCVT + 0),8)      /* VTAM Rel Lvl VOVRP   */
VTAMVER  = Substr(ATCVTLVL,3,1)              /* VTAM Version   V     */
VTAMREL  = Substr(ATCVTLVL,4,1)              /* VTAM Release    R    */
VTAMMOD  = Substr(ATCVTLVL,5,1)              /* VTAM Mod Lvl     P   */
```

```
If VTAMMOD = ' ' then VTAMLEV =  VTAMVER || '.' || VTAMREL
   else VTAMLEV =  VTAMVER || '.' || VTAMREL || '.' || VTAMMOD
If Substr(PRODNAME,3,1) < 6 then
  Queue 'The MVS version is 'PRODNAME' - FMID 'FMIDNUM'.'
Else do
  PRODNAM2 = Storage(D2x(ECVT+496),16)       /* point to product name*/
  PRODNAM2 = Strip(PRODNAM2,T)               /* del trailing blanks  */
  VER      = Storage(D2x(ECVT+512),2)        /* point to version     */
  REL      = Storage(D2x(ECVT+514),2)        /* point to release     */
  MOD      = Storage(D2x(ECVT+516),2)        /* point to mod level   */
  VRM      = VER'.'REL'.'MOD
  Queue 'The OS version is 'PRODNAM2 VRM' - FMID 'FMIDNUM'.'
End
Queue 'The primary job entry subsystem is 'JESPJESN'.'
Queue 'The 'JESPJESN 'level is 'JESLEV'.' ,
      'The 'JESPJESN 'node name is 'JESNODE'.'
If RCVTID <> 'RACF' | RACFVRM < '26Ø8' then ,
 Queue 'The security software is 'RCVTID'.' ,
       'The RACF level is 'RACFLEV'.'
Else ,
 Queue 'The security software is OS/39Ø Security Server (RACF).' ,
       'The FMID is HRF' || RACFVRM || '.'
Queue 'The' DFPRD 'level is' DFLEV'.'
Queue 'The TSO level is 'TSOLEV'.'
If SYSISPF = 'ACTIVE' then do                /* is ISPF active?      */
  Address ISPEXEC "VGET ZISPFOS"             /* yes, is it OS?39Ø?   */
  If RC = Ø then do                          /* yes, get OS/39Ø var  */
    ISPFLEV = Substr(ZISPFOS,1Ø,15)          /* only need version    */
    Queue 'The ISPF level is 'ISPFLEV'.'
  End  /* if RC */
  Else do                          /* not OS/39Ø - use old variables */
    Address ISPEXEC "VGET ZPDFREL"           /* get pdf release info */
    ISPFLEV  = Substr(ZENVIR,6,3)            /* ISPF level           */
    PDFLEV   = Substr(ZPDFREL,5,3)           /* PDF  level           */
    Queue 'The ISPF level is 'ISPFLEV'. The PDF level is' PDFLEV'.'
  End /* else do */
End  /* if SYSISPF */
Queue 'The VTAM level is 'VTAMLEV'.'
Return
STOR:                /* Storage information sub-routine             */
Queue ' '
CVTRLSTG = C2d(Storage(D2x(CVT + 856),4))    /* point to store at IPL*/
CVTRLSTG = CVTRLSTG/1Ø24                     /* convert to Megabytes */
If IPAARCHL <> 2 then do                     /* not valid in 64-bit  */
  CVTEORM  = C2d(Storage(D2x(CVT + 312),4))  /* potential real high  */
  CVTEORM  = (CVTEORM+1)/1Ø24/1Ø24           /* convert to Megabytes */
  RCE      = C2d(Storage(D2x(CVT + 1168),4)) /* point to RCE         */
  ESTOR    = C2d(Storage(D2x(RCE + 16Ø),4))  /* point to ESTOR frames*/
  ESTOR    = ESTOR*4/1Ø24                    /* convert to Megabytes */
End
Call STORAGE_GDA_LDA
```

```
Queue 'The real storage size at IPL time was 'Format(CVTRLSTG,,Ø)'M.'
If IPAARCHL <> 2 then do    /* not valid in 64-bit */
  Queue 'The potential real storage size is' ,
        Format(CVTEORM,,Ø)'M.'
  If ESTOR > Ø then
    Queue 'The expanded storage size is 'ESTOR'M.'
  Else
    Queue 'The system has no expanded storage.'
End
Queue 'The private area size <16M is 'GDAPVTSZ'K.'
Queue 'The private area size >16M is 'GDAEPVTS'M.'
Queue 'The CSA size <16M is 'GDACSASZ'K.'
Queue 'The CSA size >16M is 'GDAECSAS'K.'
Queue 'The SQA size <16M is 'GDASQASZ'K.'
Queue 'The SQA size >16M is 'GDAESQAS'K.'
Queue 'The maximum V=R region size is 'GDAVRSZ'K.'
Queue 'The default V=R region size is 'GDAVREGS'K.'
Queue 'The maximum V=V region size is 'LDASIZEA'K.'
Return
CPU:                 /* CPU information sub-routine                  */
Queue ' '
NUMCPU   = C2d(Storage(D2x(CSD + 1Ø),2))     /* point to # of CPUS   */
Queue 'The CPU model number is 'MODEL'.'
Queue 'The number of online CPUs is 'NUMCPU'.'
PCCAVT    = C2d(Storage(D2x(CVT + 764),4))   /* point to PCCA vect tb*/
CPNUM     = Ø
FOUNDCPUS = Ø
Do until FOUNDCPUS = NUMCPU
PCCA = C2d(Storage(D2x(PCCAVT + CPNUM*4),4)) /* point to PCCA        */
  If PCCA <> Ø then do
    CPUVER  = Storage(D2x(PCCA + 4),2)       /* point to VERSION     */
    CPUID   = Storage(D2x(PCCA + 6),1Ø)      /* point to CPUID       */
    IDSHORT = Substr(CPUID,2,5)
    Queue 'The CPU serial number for CPU 'CPNUM' is ' || ,
     CPUID' ('IDSHORT'), version code' CPUVER'.'
    FOUNDCPUS = FOUNDCPUS + 1
  End
CPNUM = CPNUM + 1
End  /* do until  */
/*************************************************/
/* SUs/SEC and MIPS calculations                 */
/* SYS1.NUCLEUS(IEAVNP1Ø) CSECT IRARMCPU         */
/*************************************************/
RMCT     = C2d(Storage(D2x(CVT+6Ø4),4))      /* point to RMCT        */
SU       = C2d(Storage(D2x(RMCT+64),4))      /* CPU Rate Adjustment  */
SUSEC    = Format((16ØØØØØØ/SU),7,2)         /* SUs per second       */
MIPS     = Format((SUSEC/48.5) * NUMCPU,6,2) /* SRM MIPS calculation */
Queue 'The service units per second per online CPU is' Strip(SUSEC)'.'
Queue 'The approximate total MIPS (SUs/SEC / 48.5 * #CPUs)' ,
      'is' Strip(MIPS)'.'
/*************************************************/
```

```
/* Central Processing Complex Node Descriptor     */
/************************************************/
If Substr(PRODNAME,3,1) >= 5 then do
  CVTHID   = C2d(Storage(D2x(CVT + 1068),4))   /* point to SHID      */
  CPCND_FLAGS = Storage(D2x(CVTHID+22),1)      /* pnt to CPCND FLAGS */
  If CPCND_FLAGS <> 0 then do                  /* Is there a CPC?    */
    CPCND_VALID = Bitand(CPCND_FLAGS,'E0'x)    /* Valid flags        */
    CPCND_INVALID = Bitand('40'x)              /* Invalid flag       */
    If CPCND_VALID <> CPCND_INVALID then do    /* Is it valid?       */
      CPCND_TYPE  = Storage(D2x(CVTHID+26),6)  /* Type               */
      CPCND_MODEL = Storage(D2x(CVTHID+32),3)  /* Model              */
      CPCND_MAN   = Storage(D2x(CVTHID+35),3)  /* Manufacturer       */
      CPCND_PLANT = Storage(D2x(CVTHID+38),2)  /* Plant of manufact. */
      CPCND_SEQNO = Storage(D2x(CVTHID+40),12) /* Sequence number    */
      CPC_ID      = C2x(Storage(D2x(CVTHID+55),1))  /* CPC ID        */
      Queue ' '
      Queue 'Central Processing Complex (CPC) Node Descriptor:'
      Queue '  CPC ND =',
       CPCND_TYPE'.'CPCND_MODEL'.'CPCND_MAN'.'CPCND_PLANT'.'CPCND_SEQNO
      Queue '  CPC ID =' CPC_ID
      Queue '  Type('CPCND_TYPE') Model('CPCND_MODEL')',
            'Manufacturer('CPCND_MAN') Plant('CPCND_PLANT')',
            'Seq Num('CPCND_SEQNO')'
    End /* if CPCND_VALID <> CPCND_INVALID */
  End  /* if CPCND_FLAGS <>0  */
End
Return
IPA:                /* IPA information sub-routine                */
Queue ' '
/*********************************************************************/
/* IPL parms from the IPA                                            */
/*********************************************************************/
If Substr(FMIDNUM,4,4) >= 6602 then do
  IPALPARM = Storage(D2x(ECVTIPA + 16),8)    /* point to LOAD PARM   */
  IPALPDSN = Storage(D2x(ECVTIPA + 48),44)   /* load parm dsn name   */
  IPAHWNAM = Storage(D2x(ECVTIPA + 24),8)    /* point to HWNAME      */
  IPAHWNAM = Strip(IPAHWNAM,T)               /* del trailing blanks  */
  IPALPNAM = Storage(D2x(ECVTIPA + 32),8)    /* point to LPARNAME    */
  IPALPNAM = Strip(IPALPNAM,T)               /* del trailing blanks  */
  IPAVMNAM = Storage(D2x(ECVTIPA + 40),8)    /* point to VMUSERID    */
  /*************************/
  /* PARMS in LOADxx       */
  /*************************/
  IPANUCID = Storage(D2x(ECVTIPA + 23),1)    /* NUCLEUS ID           */
  IPAIODF  = Storage(D2x(ECVTIPA + 96),63)   /* IODF    card image   */
  IPASPARM = Storage(D2x(ECVTIPA + 160),63)  /* SYSPARM card image   */
  /*IPASCAT= Storage(D2x(ECVTIPA + 224),63)*//* SYSCAT  card image   */
  IPASYM   = Storage(D2x(ECVTIPA + 288),63)  /* IEASYM  card image   */
  IPAPLEX  = Storage(D2x(ECVTIPA + 352),63)  /* SYSPLEX card image   */
  IPAPLNUM = Storage(D2x(ECVTIPA + 2148),2)  /* number of parmlibs   */
  IPAPLNUM = C2x(IPAPLNUM)                   /* convert to EBCDIC    */
```

```
  POFF = Ø
  Do P = 1 to IPAPLNUM
    IPAPLIB.P = Storage(D2x(ECVTIPA+416+POFF),63) /* PARMLIB cards   */
    IPAPLFLG.P = Storage(D2x(ECVTIPA+479+POFF),1)  /* flag bits      */
    If bitand(IPAPLFLG.P,'2Ø'x) = '2Ø'x then ,   /* volser from cat? */
      IPAPLIB.P = Overlay('      ',IPAPLIB.P,46) /* no, clear it     */
    POFF = POFF + 64
  End
  IPANLID  = Storage(D2x(ECVTIPA + 2144),2)  /* NUCLSTxx member used */
  IPANUCW  = Storage(D2x(ECVTIPA + 2146),1)  /* load wait state char */
  Queue 'Initialization information from the IPA:'
  Queue '  IPLPARM =' IPALPARM   '(merged)'
  Queue '  IPL load parameter dataset name: 'IPALPDSN
  Queue '  HWNAME='IPAHWNAM '  LPARNAME='IPALPNAM ,
        '  VMUSERID='IPAVMNAM
  Queue '  LOADxx parameters (LOAD' || Substr(IPALPARM,5,2) || '):'
  Queue '    *--+----1----+----2----+----3----+----4' || ,
           '----+----5----+----6----+----7'
  If Substr(FMIDNUM,4,4) >  66Ø9 then ,      /* OS/39Ø R1Ø or above  */
    Queue '    ARCHLVL  'IPAARCHL
  If IPASYM   <> '' then queue '    IEASYM   'IPASYM
  If IPAIODF  <> '' then queue '    IODF     'IPAIODF
  If IPANUCID <> '' then queue '    NUCLEUS  'IPANUCID
  If IPANLID  <> '' then queue '    NUCLST   'IPANLID' 'IPANUCW
  Do P = 1 to IPAPLNUM
    Queue '    PARMLIB  'IPAPLIB.P
  End
  If IPASCAT  <> '' then queue '    SYSCAT   'IPASCAT
  If IPASPARM <> '' then queue '    SYSPARM  'IPASPARM
  If IPAPLEX  <> '' then queue '    SYSPLEX  'IPAPLEX
  /*************************/
  /* PARMS in IEASYSxx     */
  /*************************/
  Queue '  IEASYSxx parameters:                        ',
        '                      (Source)'
  Call BUILD_IPAPDETB           /* Build table for init parms       */
  TOTPRMS = Ø           /* tot num of specified or defaulted parms  */
  Do I = 1 to IPAPDETB.Ø
    Call EXTRACT_SYSPARMS IPAPDETB.I  /* extract parms from the IPA */
  End
  Do I = 1 to TOTPRMS                 /* add parms                  */
    If I = TOTPRMS then ,             /*   to stack and             */
      PRMLINE.I = Translate(PRMLINE.I,' ',',') /* remove comma      */
    Queue PRMLINE.I                   /*           from last parm   */
  End
End
Return
SYMBOLS:             /* System Symbols information sub-routine       */
Queue ' '
/*********************************************************************/
/* Find System Symbols  - ASASYMBP MACRO                             */
```

```
/*  ECVT+X'128' = ECVTSYMT                                              */
/*  2nd half word = # of symbols , after that each entry is 4 words  */
/*  1st word = offset to symbol name                                    */
/*  2nd word = length of symbol name                                    */
/*  3rd word = offset to symbol value                                   */
/*  4th word = length of symbol value                                   */
/*********************************************************************/
If Substr(FMIDNUM,4,4) >= 552Ø then do
  ECVTSYMT = C2d(Storage(D2x(ECVT + 296),4)) /* point to ECVTSYMT    */
  NUMSYMBS = C2d(Storage(D2x(ECVTSYMT + 2),2))  /* number of symbols */
  Queue 'Static System Symbol Values:'
  Do I = 1 to NUMSYMBS
    SOFF = I*16-16
    NAMOFF  = C2d(Storage(D2x(ECVTSYMT+4+SOFF),4))  /*offset to name */
    NAMLEN  = C2d(Storage(D2x(ECVTSYMT+8+SOFF),4))  /*length of name */
    VALOFF  = C2d(Storage(D2x(ECVTSYMT+12+SOFF),4)) /*offset to value*/
    VALLEN  = C2d(Storage(D2x(ECVTSYMT+16+SOFF),4)) /*length of value*/
    SYMNAME = Storage(D2x(ECVTSYMT+4+NAMOFF),NAMLEN) /*symbol name   */
    If VALLEN = Ø then VALNAME = ''                 /* null value    */
    Else ,
    VALNAME = Storage(D2x(ECVTSYMT+4+VALOFF),VALLEN) /* symbol value */
    Queue ' ' Left(SYMNAME,1Ø,' ') '=' VALNAME
  End  /* do NUMSYMBS */
End
Return
VMAP:                /* Virtual Storage Map sub-routine             */
Queue ' '
If option <> 'ALL' then,
  Call STORAGE_GDA_LDA                       /* GDA/LDA stor routine */
SYSEND  = X2d(LDASTRTS) + (LDASIZS*1Ø24) - 1 /* end of system area   */
SYSEND  = D2x(SYSEND)                        /* display in hex       */
If GDAVRSZ = Ø then do                       /* no v=r               */
  VRSTRT = 'N/A     '
  VREND  = 'N/A     '
  VVSTRT = LDASTRTA                          /* start of v=v         */
  VVEND  =  X2d(LDASTRTA) + (LDASIZEA*1Ø24) - 1 /* end of v=v        */
  VVEND  =  D2x(VVEND)                       /* display in hex       */
End
Else do
  VRSTRT =  LDASTRTA                         /* start of v=r         */
  VREND  =  X2d(LDASTRTA) + (GDAVRSZ*1Ø24) - 1 /* end of v=r         */
  VREND  =  D2X(VREND)                       /* display in hex       */
  VVSTRT =  LDASTRTA                         /* start of v=v         */
  VVEND  =  X2d(LDASTRTA) + (LDASIZEA*1Ø24) - 1 /* end of v=v        */
  VVEND  =  D2x(VVEND)                       /* display in hex       */
End
GDACSA   = C2d(Storage(D2x(CVTGDA + 1Ø8),4)) /* start of CSA addr    */
GDACSAH  = D2x(GDACSA)                       /* display in hex       */
CSAEND   = (GDACSASZ*1Ø24) + GDACSA - 1      /* end of CSA           */
CSAEND   = D2x(CSAEND)                       /* display in hex       */
CVTSMEXT = C2d(Storage(D2x(CVT +1196),4))    /* point to stg map ext.*/
```

```
CVTMLPAS = C2d(Storage(D2x(CVTSMEXT+ 8),4))  /* start of MLPA addr   */
CVTMLPAS = D2x(CVTMLPAS)                     /* display in hex       */
If CVTMLPAS <> Ø then do
  CVTMLPAE = C2d(Storage(D2x(CVTSMEXT+12),4))  /* end of MLPA addr   */
  CVTMLPAE = D2x(CVTMLPAE)                     /* display in hex     */
  MLPASZ   = X2d(CVTMLPAE) - X2d(CVTMLPAS) + 1 /* size of MLPA       */
  MLPASZ   = MLPASZ/1Ø24                       /* convert to Kbytes  */
End
Else do /* no MLPA */
  CVTMLPAS = 'N/A     '
  CVTMLPAE = 'N/A     '
  MLPASZ   = Ø
End
CVTFLPAS = C2d(Storage(D2x(CVTSMEXT+16),4))  /* start of FLPA addr   */
CVTFLPAS = D2x(CVTFLPAS)                     /* display in hex       */
If CVTFLPAS <> Ø then do
  CVTFLPAE = C2d(Storage(D2x(CVTSMEXT+2Ø),4))  /* end of FLPA addr   */
  CVTFLPAE = D2x(CVTFLPAE)                     /* display in hex     */
  FLPASZ   = X2d(CVTFLPAE) - X2d(CVTFLPAS) + 1 /* size of FLPA       */
  FLPASZ   = FLPASZ/1Ø24                       /* convert to Kbytes  */
End
Else do /* no FLPA */
  CVTFLPAS = 'N/A     '
  CVTFLPAE = 'N/A     '
  FLPASZ   = Ø
End
CVTPLPAS = C2d(Storage(D2x(CVTSMEXT+24),4))  /* start of PLPA addr   */
CVTPLPAS = D2x(CVTPLPAS)                     /* display in hex       */
CVTPLPAE = C2d(Storage(D2x(CVTSMEXT+28),4))  /* end of PLPA addr     */
CVTPLPAE = D2x(CVTPLPAE)                     /* display in hex       */
PLPASZ   = X2d(CVTPLPAE) - X2d(CVTPLPAS) + 1 /* size of PLPA         */
PLPASZ   = PLPASZ/1Ø24                       /* convert to Kbytes    */
GDASQA   = C2d(Storage(D2x(CVTGDA + 144),4)) /* start of SQA addr    */
GDASQAH  = D2x(GDASQA)                       /* display in hex       */
SQAEND   = (GDASQASZ*1Ø24) + GDASQA - 1      /* end of SQA           */
SQAEND   = D2x(SQAEND)                       /* display in hex       */
CVTRWNS  = C2d(Storage(D2x(CVTSMEXT+32),4))  /* start of R/W nucleus */
CVTRWNS  = D2x(CVTRWNS)                      /* display in hex       */
CVTRWNE  = C2d(Storage(D2x(CVTSMEXT+36),4))  /* end of R/W nucleus   */
CVTRWNE  = D2x(CVTRWNE)                      /* display in hex       */
RWNUCSZ  = X2d(CVTRWNE)  - X2d(CVTRWNS)  + 1 /* size of R/W nucleus  */
RWNUCSZ  = Format(RWNUCSZ/1Ø24,,Ø)           /* convert to Kbytes    */
CVTRONS  = C2d(Storage(D2x(CVTSMEXT+4Ø),4))  /* start of R/O nucleus */
CVTRONS  = D2x(CVTRONS)                      /* display in hex       */
CVTRONE  = C2d(Storage(D2x(CVTSMEXT+44),4))  /* end of R/O nucleus   */
CVTRONE  = D2x(CVTRONE)                      /* display in hex       */
RONUCSZ  = X2d(CVTRONE)  - X2d(CVTRONS)  + 1 /* size of R/O nucleus  */
RONUCSZ  = Format(RONUCSZ/1Ø24,,Ø)           /* convert to Kbytes    */
RONUCSZB = X2d('FFFFFF') - X2d(CVTRONS) + 1  /* size of R/O nuc <16M */
RONUCSZB = Format(RONUCSZB/1Ø24,,Ø)          /* convert to Kbytes    */
RONUCSZA = X2d(CVTRONE) - X2d('1ØØØØØØ') + 1 /* size of R/O nuc >16M */
```

```
RONUCSZA = Format(RONUCSZA/1Ø24,,Ø)          /* convert to Kbytes    */
CVTERWNS = C2d(Storage(D2x(CVTSMEXT+48),4))  /* start of E-R/W nuc   */
CVTERWNS = D2x(CVTERWNS)                     /* display in hex       */
CVTERWNE = C2d(Storage(D2x(CVTSMEXT+52),4))  /* end of E-R/W nuc     */
CVTERWNE = D2x(CVTERWNE)                     /* display in hex       */
ERWNUCSZ = X2d(CVTERWNE) - X2d(CVTERWNS) + 1 /* size of E-R/W nuc    */
ERWNUCSZ = ERWNUCSZ/1Ø24                     /* convert to Kbytes    */
GDAESQA  = C2d(Storage(D2x(CVTGDA + 152),4)) /* start of ESQA addr   */
GDAESQAH = D2x(GDAESQA)                      /* display in hex       */
ESQAEND  = (GDAESQAS*1Ø24) + GDAESQA - 1     /* end of ESQA          */
ESQAEND  = D2x(ESQAEND)                      /* display in hex       */
CVTEPLPS = C2d(Storage(D2x(CVTSMEXT+56),4))  /* start of EPLPA addr  */
CVTEPLPS = D2x(CVTEPLPS)                     /* display in hex       */
CVTEPLPE = C2d(Storage(D2x(CVTSMEXT+6Ø),4))  /* end of EPLPA addr    */
CVTEPLPE = D2x(CVTEPLPE)                     /* display in hex       */
EPLPASZ  = X2d(CVTEPLPE) - X2d(CVTEPLPS) + 1 /* size of EPLPA        */
EPLPASZ  = EPLPASZ/1Ø24                      /* convert to Kbytes    */
CVTEFLPS = C2d(Storage(D2x(CVTSMEXT+64),4))  /* start of EFLPA addr  */
CVTEFLPS = D2x(CVTEFLPS)                     /* display in hex       */
If CVTEFLPS <> Ø then do
  CVTEFLPE = C2d(Storage(D2x(CVTSMEXT+68),4))  /* end of EFLPA addr  */
  CVTEFLPE = D2x(CVTEFLPE)                     /* display in hex     */
  EFLPASZ  = X2d(CVTEFLPE) - X2d(CVTEFLPS) + 1 /* size of EFLPA      */
  EFLPASZ  = EFLPASZ/1Ø24                      /* convert to Kbytes  */
End
Else do /* no EFLPA */
  CVTEFLPS = 'N/A     '
  CVTEFLPE = 'N/A     '
  EFLPASZ  = Ø
End
CVTEMLPS = C2d(Storage(D2x(CVTSMEXT+72),4))  /* start of EMLPA addr  */
CVTEMLPS = D2x(CVTEMLPS)                     /* display in hex       */
If CVTEMLPS <> Ø then do
  CVTEMLPE = C2d(Storage(D2x(CVTSMEXT+76),4))  /* end of EMLPA addr  */
  CVTEMLPE = D2x(CVTEMLPE)                     /* display in hex     */
  EMLPASZ  = X2d(CVTEMLPE) - X2d(CVTEMLPS) + 1 /* size of EMLPA      */
  EMLPASZ  = EMLPASZ/1Ø24                      /* convert to Kbytes  */
End
Else do /* no EMLPA */
  CVTEMLPS = 'N/A     '
  CVTEMLPE = 'N/A     '
  EMLPASZ  = Ø
End
GDAECSA  = C2d(Storage(D2x(CVTGDA + 124),4)) /* start of ECSA addr   */
GDAECSAH = D2x(GDAECSA)                      /* display in hex       */
ECSAEND  = (GDAECSAS*1Ø24) + GDAECSA - 1     /* end of ECSA          */
ECSAEND  = D2x(ECSAEND)                      /* display in hex       */
GDAEPVT  = C2d(Storage(D2x(CVTGDA + 168),4)) /* start of EPVT addr   */
GDAEPVTH = D2x(GDAEPVT)                      /* display in hex       */
EPVTEND  = (GDAEPVTS*1Ø24*1Ø24) + GDAEPVT - 1 /* end of EPVT         */
EPVTEND  = D2x(EPVTEND)                      /* display in hex       */
```

```
Queue 'Virtual Storage Map:'
Queue '          '
Queue '     Storage Area     Start      End           Size' ,
      '     Used     Conv'
Queue '          '
If IPAARCHL = 2 then ,
  Queue '              PSA     00000000   00001FFF        8K'
Else ,
  Queue '              PSA     00000000   00000FFF        4K'
Queue '           System    '     Right(LDASTRTS,8,'0') ' ' ,
   Right(SYSEND,8,'0')            Right(LDASIZS,8,' ')'K'
Queue '      Private V=R    '     Right(VRSTRT,8,'0') ' ' ,
   Right(VREND,8,'0')             Right(GDAVRSZ,8,' ')'K'
Queue '      Private V=V    '     Right(VVSTRT,8,'0') ' ' ,
   Right(VVEND,8,'0')             Right(LDASIZEA,8,' ')'K'
Queue '              CSA    '     Right(GDACSAH,8,'0') ' ' ,
   Right(CSAEND,8,'0')            Right(GDACSASZ,8,' ')'K' ,
   Right(GDA_CSA_ALLOC,8,' ')'K'
Queue '             MLPA    '     Right(CVTMLPAS,8,'0') ' ' ,
   Right(CVTMLPAE,8,'0')          Right(MLPASZ,8,' ')'K'
Queue '             FLPA    '     Right(CVTFLPAS,8,'0') ' ' ,
   Right(CVTFLPAE,8,'0')          Right(FLPASZ,8,' ')'K'
Queue '             PLPA    '     Right(CVTPLPAS,8,'0') ' ' ,
   Right(CVTPLPAE,8,'0')          Right(PLPASZ,8,' ')'K'
Queue '              SQA    '     Right(GDASQAH,8,'0') ' ' ,
   Right(SQAEND,8,'0')            Right(GDASQASZ,8,' ')'K' ,
   Right(GDA_SQA_ALLOC,8,' ')'K'  Right(GDA_CSA_CONV,7,' ')'K'
Queue '      R/W Nucleus    '     Right(CVTRWNS,8,'0') ' ' ,
   Right(CVTRWNE,8,'0')           Right(RWNUCSZ,8,' ')'K'
Queue '      R/O Nucleus    '     Right(CVTRONS,8,'0') ' ' ,
   Right('FFFFFF',8,'0')          Right(RONUCSZB,8,' ')'K',
   '(Spans 16M line)'
Queue '             16M line ---------------------------'
Queue ' Ext. R/O Nucleus    '     Right('1000000',8,'0') ' ' ,
   Right(CVTRONE,8,'0')           Right(RONUCSZA,8,' ')'K' ,
   '(Total' RONUCSZ'K)'
Queue ' Ext. R/W Nucleus    '     Right(CVTERWNS,8,'0') ' ' ,
   Right(CVTERWNE,8,'0')          Right(ERWNUCSZ,8,' ')'K'
Queue '         Ext. SQA    '     Right(GDAESQAH,8,'0') ' ' ,
   Right(ESQAEND,8,'0')           Right(GDAESQAS,8,' ')'K' ,
   Right(GDA_ESQA_ALLOC,8,' ')'K' Right(GDA_ECSA_CONV,7,' ')'K'
Queue '        Ext. PLPA    '     Right(CVTEPLPS,8,'0') ' ' ,
   Right(CVTEPLPE,8,'0')          Right(EPLPASZ,8,' ')'K'
Queue '        Ext. FLPA    '     Right(CVTEFLPS,8,'0') ' ' ,
   Right(CVTEFLPE,8,'0')          Right(EFLPASZ,8,' ')'K'
Queue '        Ext. MLPA    '     Right(CVTEMLPS,8,'0') ' ' ,
   Right(CVTEMLPE,8,'0')          Right(EMLPASZ,8,' ')'K'
Queue '         Ext. CSA    '     Right(GDAECSAH,8,'0') ' ' ,
   Right(ECSAEND,8,'0')           Right(GDAECSAS,8,' ')'K' ,
   Right(GDA_ECSA_ALLOC,8,' ')'K'
Queue '     Ext. Private    '     Right(GDAEPVTH,8,'0') ' ' ,
```

```
   Right(EPVTEND,8,'0')           Right(GDAEPVTS,8,' ')'M'
Return
PAGE:                /* Page Datasets information sub-routine      */
Queue ' '
Queue 'Page Dataset Usage:'
Queue '  Type     Full   Slots  Dev   Volser  Data Set Name'
CVT      = C2d(Storage(10,4))              /* point to CVT          */
ASMVT    = C2d(Storage(D2x(CVT + 704),4))  /* point to ASMVT        */
ASMPART  = C2d(Storage(D2x(ASMVT + 8),4))  /* Pnt to Pag Act Ref Tbl */
PARTSIZE = C2d(Storage(D2x(ASMPART+4),4))  /* Tot number of entries  */
PARTDSNL = C2d(Storage(D2x(ASMPART+24),4)) /* Point to 1st pg dsn    */
PARTENTS = ASMPART+80                      /* Point to 1st parte     */
Do I = 1 to PARTSIZE
  If I > 1 then do
    PARTENTS = PARTENTS + 96
    PARTDSNL = PARTDSNL + 44
  End
  CHKINUSE = Storage(D2x(PARTENTS+9),1)    /* in use flag           */
  If bitand(CHKINUSE,'80'x) = '80'x then iterate /* not in use      */
  PGDSN    = Storage(D2x(PARTDSNL),44)     /* page dataset name     */
  PGDSN    = Strip(PGDSN,T)                /* remove trailing blanks */
  PARETYPE = Storage(D2x(PARTENTS+8),1)    /* type flag             */
  Select
    When bitand(PARETYPE,'80'x) = '80'x then PGTYPE = ' PLPA    '
    When bitand(PARETYPE,'40'x) = '40'x then PGTYPE = ' COMMON  '
    When bitand(PARETYPE,'20'x) = '20'x then PGTYPE = ' DUPLEX  '
    When bitand(PARETYPE,'10'x) = '10'x then PGTYPE = ' LOCAL   '
    Otherwise PGTYPE = '??????'
  End  /* Select */
  If PGTYPE = ' LOCAL   ' then do
    PAREFLG1  = Storage(D2x(PARTENTS+9),1)    /* PARTE flags        */
    If bitand(PAREFLG1,'10'x) = '10'x then PGTYPE = ' LOCAL NV'
  End
  PAREUCBP = C2d(Storage(D2x(PARTENTS+44),4)) /* point to UCB       */
  PGUCB    = C2x(Storage(D2x(PAREUCBP+4),2))  /* UCB address        */
  PGVOL    = Storage(D2x(PAREUCBP+28),6)      /* UCB volser         */
  PARESZSL = C2d(Storage(D2x(PARTENTS+16),4)) /* total slots        */
  PARESZSL = Right(PARESZSL,7,' ')           /* ensure 7 digits    */
  PARESLTA = C2d(Storage(D2x(PARTENTS+20),4)) /* avail. slots       */
  PGFULL   = ((PARESZSL-PARESLTA) / PARESZSL) * 100 /* percent full  */
  PGFULL   = Format(PGFULL,3,2)              /* force 2 decimals   */
  PGFULL   = Left(PGFULL,3)                  /* keep integer only  */
  Queue  ' 'PGTYPE' 'PGFULL'% 'PARESZSL'  'PGUCB' ' ,
         PGVOL'  'PGDSN
End  /* do I=1 to partsize */
Return
SUB:                 /* Subsystem information sub-routine          */
Arg SUBOPT
SSCVT    = C2d(Storage(D2x(JESCT+24),4))     /* point to SSCVT      */
SSCVT2   = SSCVT           /* save address for second loop         */
If SUBOPT <> 'FINDJES' then do
```

```
  Queue ' '
  Queue 'Subsystem Communications Vector Table:'
  Queue '  Name   Hex        SSCTADDR   SSCTSSVT' ,
        '  SSCTSUSE   SSCTSUS2   Status'
End /* if subopt */
Do until SSCVT = Ø
  SSCTSNAM = Storage(D2x(SSCVT+8),4)         /* subsystem name       */
  SSCTSSVT = C2d(Storage(D2x(SSCVT+16),4))   /* subsys vect tbl ptr  */
  SSCTSUSE = C2d(Storage(D2x(SSCVT+2Ø),4))   /* SSCTSUSE pointer     */
  SSCTSUS2 = C2d(Storage(D2x(SSCVT+28),4))   /* SSCTSUS2 pointer     */
  If SUBOPT = 'FINDJES' & SSCTSNAM = JESPJESN then do
     JESSSVT  = SSCTSSVT   /* save SSVTSSVT for "version" section    */
                           /* this points to JES3 Subsystem Vector   */
                           /* Table, mapped by IATYSVT               */
     JESSUSE  = SSCTSUSE   /* save SSCTSUSE for "version" section    */
                           /* this points to version for JES2        */
     JESSUS2  = SSCTSUS2   /* save SSCTSUS2 for "version" section    */
                           /* this points to $HCCT for JES2          */
     Leave  /* found JES info for version section, exit loop */
  End /* if subopt */
  SSCTSNAX = C2x(SSCTSNAM)    /* chg to EBCDIC for non-display chars */
  Call XLATE_NONDISP SSCTSNAM /* translate non display chars         */
  SSCTSNAM = RESULT           /* result from XLATE_NONDISP           */
  If SSCTSSVT = Ø then SSCT_STAT = 'Inactive'
    Else SSCT_STAT = 'Active'
  If SUBOPT <> 'FINDJES' then do
    Queue ' ' SSCTSNAM ' ' SSCTSNAX  ,
          ' ' Right(D2x(SSCVT),8,Ø)    ' ' Right(D2x(SSCTSSVT),8,Ø) ,
          ' ' Right(D2x(SSCTSUSE),8,Ø) ' ' Right(D2x(SSCTSUS2),8,Ø) ,
          ' ' SSCT_STAT ' '
  End /* if SUBOPT */
 /*SSCTSSID = C2d(Storage(D2x(SSCVT+13),1)) */ /* subsys identifier  */
 /*If bitand(SSCTSSID,'Ø2'x) = 'Ø2'x then JESPJESN = 'JES2' */
 /*If bitand(SSCTSSID,'Ø3'x) = 'Ø3'x then JESPJESN = 'JES3'*/
  SSCVT    = C2d(Storage(D2x(SSCVT+4),4))    /* next sscvt or zero   */
End /* do until sscvt = Ø */
If SUBOPT <> 'FINDJES' then do
  Queue ' '
  Queue 'Supported Subsystem Function Codes:'
  Do until SSCVT2 = Ø /* 2nd loop for function codes                 */
    SSCTSNAM = Storage(D2x(SSCVT2+8),4)        /* subsystem name     */
    SSCTSSVT = C2d(Storage(D2x(SSCVT2+16),4)) /* subsys vect tbl ptr */
    SSCTSNAX = C2x(SSCTSNAM)  /* chg to EBCDIC for non-display chars */
    Call XLATE_NONDISP SSCTSNAM /* translate non display chars       */
    SSCTSNAM = RESULT           /* result from XLATE_NONDISP         */
    Queue SSCTSNAM '(X''' || SSCTSNAX || ''')'
    If SSCTSSVT <> Ø then do
      SSVTFCOD = SSCTSSVT + 4                  /* pt to funct. matrix*/
      SSFUNCTB = Storage(D2X(SSVTFCOD),255)    /* function matrix    */
      TOTFUNC = Ø       /* counter for total functions per subsystem */
      Drop FUNC.        /* init stem to null for saved functions     */
```

```
      Do SUPFUNC = 1 TO 255
        If Substr(SSFUNCTB,SUPFUNC,1) <> '00'x then do /* supported? */
          TOTFUNC = TOTFUNC + 1 /* tot functions for this subsystem  */
          FUNC.TOTFUNC = SUPFUNC  /* save function in stem           */
        End
      End /* do supfunc */
      /*************************************************************/
      /* The following code is used to list the supported functions  */
      /* on a single line by ranges. For example: 1-10,13,18-30,35   */
      /*************************************************************/
      If TOTFUNC >= 1 then do   /* begin loop to list function codes */
        ALLCODES = ''                   /* init var to nulls         */
        NEWRANGE = 'YES'                /* init newrange flag to YES */
        FIRSTRNG = 'YES'                /* init firstrng flag to YES */
        Do FCODES = 1 to TOTFUNC        /* loop though codes         */
          CHKNEXT = FCODES + 1          /* stem var to chk next code */
          If FUNC.FCODES + 1 = FUNC.CHKNEXT then do  /* next matches */
            If NEWRANGE = 'YES' & FIRSTRNG = 'YES' then do /* first  */
              ALLCODES =  ALLCODES || FUNC.FCODES || '-'   /* in new */
              NEWRANGE = 'NO'                   /* range - separate */
              FIRSTRNG = 'NO'                   /* with a dash      */
              Iterate                           /* get next code    */
            End /* if newrange = 'yes' & firstrng = 'yes'            */
            If NEWRANGE = 'YES' & FIRSTRNG = 'NO' then do /* next    */
              ALLCODES =  ALLCODES || FUNC.FCODES  /* matches, but   */
              NEWRANGE = 'NO'   /* is not the first, don't add dash  */
              Iterate                           /* get next code    */
            End /* if newrange = 'yes' & firstrng = 'no'             */
            Else iterate  /* same range + not first - get next code  */
          End /* func.fcodes + 1 */
          If FCODES = TOTFUNC then , /* next doesn't match and this  */
            ALLCODES =  ALLCODES || FUNC.FCODES  /* is the last code */
          Else do /* next code doesn't match - seperate with comma   */
            ALLCODES =  ALLCODES || FUNC.FCODES || ','
            NEWRANGE = 'YES'         /* re-init newrange flag to YES */
            FIRSTRNG = 'YES'         /* re-init firstrng flag to YES */
          End
        End /* do fcodes = 1 to totfunc */
        Queue '  Codes:' ALLCODES
      End /* if totfunc >= 1 */
    End
    Else Queue '  *Inactive*'
    SSCVT2   = C2d(Storage(D2x(SSCVT2+4),4))   /* next sscvt or zero */
  End /* do until sscvt2 = 0 */
End /* if subopt <> 'findjes' */
Return
XLATE_NONDISP:       /* translate non-display characters to a "."    */
Arg XLATEPRM
XLATELEN = Length(XLATEPRM) /* length of parm passed to routine      */
Do I = 1 to XLATELEN                      /* check each byte for     */
  If Substr(XLATEPRM,I,1) < '40'x | ,     /* non-display characters  */
```

```
     Substr(XLATEPRM,I,1) = 'FF'x  then , /* and replace each char   */
    XLATEPRM = OVERLAY('.',XLATEPRM,I)    /* that is non-displayable */
End                                       /* with a period (.)       */
Return XLATEPRM
STORAGE_GDA_LDA:     /* GDA/LDA Storage values sub-routine           */
ASCB     = C2d(Storage(224,4))               /* point to cur ASCB    */
ASCBLDA  = C2d(Storage(D2x(ASCB + 48),4))    /* point to LDA         */
CVTGDA   = C2d(Storage(D2x(CVT + 560),4))    /* point to GDA         */
LDASTRTA = Storage(D2x(ASCBLDA + 60),4)      /* point to V=V start   */
LDASTRTA = C2x(LDASTRTA)                     /* display in hex       */
LDASIZEA = C2d(Storage(D2x(ASCBLDA + 64),4)) /* point to V=V size    */
LDASIZEA = LDASIZEA/1024                     /* convert to Kbytes    */
LDASTRTS = Storage(D2x(ASCBLDA + 92),4)      /* pt. to sysarea start */
LDASTRTS = C2x(LDASTRTS)                     /* display in hex       */
LDASIZS  = C2d(Storage(D2x(ASCBLDA + 96),4)) /* pt. to sysarea size  */
LDASIZS  = LDASIZS/1024                      /* convert to Kbytes    */
GDAPVTSZ = C2d(Storage(D2x(CVTGDA + 164),4)) /* point to MAX PVT<16M */
GDAPVTSZ = GDAPVTSZ/1024                     /* convert to Kbytes    */
GDAEPVTS = C2d(Storage(D2x(CVTGDA + 172),4)) /* point to MAX PVT>16M */
GDAEPVTS = GDAEPVTS/1024/1024                /* convert to Mbytes    */
GDACSASZ = C2d(Storage(D2x(CVTGDA + 112),4)) /* point to CSA<16M     */
GDACSASZ = GDACSASZ/1024                     /* convert to Kbytes    */
GDAECSAS = C2d(Storage(D2x(CVTGDA + 128),4)) /* point to CSA>16M     */
GDAECSAS = GDAECSAS/1024                     /* convert to Kbytes    */
GDASQASZ = C2d(Storage(D2x(CVTGDA + 148),4)) /* point to SQA<16M     */
GDASQASZ = GDASQASZ/1024                     /* convert to Kbytes    */
GDAESQAS = C2d(Storage(D2x(CVTGDA + 156),4)) /* point to SQA>16M     */
GDAESQAS = GDAESQAS/1024                     /* convert to Kbytes    */
GDAVRSZ  = C2d(Storage(D2x(CVTGDA + 196),4)) /* point to V=R global  */
GDAVRSZ  = GDAVRSZ/1024                      /* convert to Kbytes    */
GDAVREGS = C2d(Storage(D2x(CVTGDA + 200),4)) /* point to V=R default */
GDAVREGS = GDAVREGS/1024                     /* convert to Kbytes    */
GDA_CSA_ALLOC  = C2d(Storage(D2x(CVTGDA + 432),4)) /* CSA amt alloc  */
GDA_CSA_ALLOC  = Format(GDA_CSA_ALLOC/1024,,0)     /* conv to Kbytes */
GDA_ECSA_ALLOC = C2d(Storage(D2x(CVTGDA + 436),4)) /* ECSA amt alloc */
GDA_ECSA_ALLOC = Format(GDA_ECSA_ALLOC/1024,,0)    /* conv to Kbytes */
GDA_SQA_ALLOC  = C2d(Storage(D2x(CVTGDA + 440),4)) /* SQA amt alloc  */
GDA_SQA_ALLOC  = Format(GDA_SQA_ALLOC/1024,,0)     /* conv to Kbytes */
GDA_ESQA_ALLOC = C2d(Storage(D2x(CVTGDA + 444),4)) /* ESQA amt alloc */
GDA_ESQA_ALLOC = Format(GDA_ESQA_ALLOC/1024,,0)    /* conv to Kbytes */
GDA_CSA_CONV   = C2d(Storage(D2x(CVTGDA + 448),4)) /* CSA => SQA amt */
GDA_CSA_CONV   = Format(GDA_CSA_CONV/1024,,0)      /* conv to Kbytes */
GDA_ECSA_CONV  = C2d(Storage(D2x(CVTGDA + 452),4)) /* ECSA=>ESQA amt */
GDA_ECSA_CONV  = Format(GDA_ECSA_CONV/1024,,0)     /* conv to Kbytes */
Return
EXTRACT_SYSPARMS:    /* Extract IEASYSxx values from the IPA         */
Arg IEASPARM
IEASPARM = Strip(IEASPARM,T)                 /* remove trailing blnks*/
IPAOFF = ((I-1) * 8)                         /* offset to next entry */
IPASTOR = D2x(ECVTIPA + 2152 + IPAOFF)       /* point to PDE addr    */
IPAPDE  = C2x(Storage((IPASTOR),8))          /* point to PDE         */
```

```
If IPAPDE = Ø then return   /* parm not specified and has no default */
TOTPRMS = TOTPRMS + 1    /* tot num of specified or defaulted parms  */
IPAADDR = Substr(IPAPDE,1,8)                 /* PARM address         */
IPALEN  = X2d(Substr(IPAPDE,9,4))            /* PARM length          */
IPAPRM  = Storage((IPAADDR),IPALEN)          /* PARM                 */
IPASRC  = Substr(IPAPDE,13,4)                /* PARM source          */
If X2d(IPASRC) = 65535 then PRMSRC = 'Operator'   /* operator parm   */
Else
  If X2d(IPASRC) = Ø     then PRMSRC = 'Default'  /* default  parm   */
Else
  PRMSRC = 'IEASYS' || X2c(IPASRC)           /* IEASYSxx parm        */
/*********************************************************************/
/* CODE to split up page dataset parms  to multiple lines           */
/*********************************************************************/
If IEASPARM = 'NONVIO'   | IEASPARM = 'PAGE' | ,
   IEASPARM = 'PAGE-OPR' | IEASPARM = 'SWAP' then do
  MORE  = 'YES'
  FIRST = 'YES'
  SPLITPOS = 1
  Do until MORE = 'NO'
    SPLITPOS = Pos(',',IPAPRM)
    If SPLITPOS = Ø then do
      If FIRST = 'YES' then PRMLINE = '    'IEASPARM'='IPAPRM || ','
        Else PRMLINE = '          'IPAPRM || ','
      MORE = 'NO'
    End
    Else do
      IPAPRM_SPLIT = Substr(IPAPRM,1,SPLITPOS)
      If FIRST = 'YES' then PRMLINE = '    'IEASPARM'='IPAPRM_SPLIT
        Else PRMLINE = '          'IPAPRM_SPLIT
      PRMLINE.TOTPRMS = PRMLINE
      TOTPRMS = TOTPRMS + 1     /* add one to num specified parms */
      IPAPRM  = Substr(IPAPRM,SPLITPOS+1,IPALEN-SPLITPOS)
      IPAPRM  = Strip(IPAPRM,T) /* remove trailing blanks */
      FIRST = 'NO'
    End
  End  /* do until */
End
Else PRMLINE = '    'IEASPARM'='IPAPRM || ','      /* not a page dsn */
PRMLINE = Overlay(PRMSRC,PRMLINE,68)
PRMLINE.TOTPRMS = PRMLINE
Return
BUILD_IPAPDETB:      /* Build table for lookup for IPA values        */
NUM=1
IPAPDETB.NUM = 'ALLOC   ' ; NUM = NUM + 1
IPAPDETB.NUM = 'APF     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'APG     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'BLDL    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'BLDLF   ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CLOCK   ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CLPA    ' ; NUM = NUM + 1
```

```
IPAPDETB.NUM = 'CMB     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CMD     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CON     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CONT    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'COUPLE  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CPQE    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CSA     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CSCBLOC ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CVIO    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'DEVSUP  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'DIAG    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'DUMP    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'DUPLEX  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'EXIT    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'FIX     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'GRS     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'GRSCNF  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'GRSRNL  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'ICS     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'IOS     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'IPS     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'LNK     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'LNKAUTH ' ; NUM = NUM + 1
IPAPDETB.NUM = 'LOGCLS  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'LOGLMT  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'LOGREC  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'LPA     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'MAXCAD  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'MAXUSER ' ; NUM = NUM + 1
IPAPDETB.NUM = 'MLPA    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'MSTRJCL ' ; NUM = NUM + 1
IPAPDETB.NUM = 'NONVIO  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'NSYSLX  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'NUCMAP  ' ; NUM = NUM + 1
If Substr(FMIDNUM,4,4) >= 66Ø3 then do
   IPAPDETB.NUM = 'OMVS    ' ; NUM = NUM + 1
End
Else do
   IPAPDETB.NUM = 'RESERVED' ; NUM = NUM + 1
End
IPAPDETB.NUM = 'OPI     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'OPT     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'PAGE-OPR' ; NUM = NUM + 1
IPAPDETB.NUM = 'PAGE    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'PAGNUM  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'PAGTOTL ' ; NUM = NUM + 1
IPAPDETB.NUM = 'PAK     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'PLEXCFG ' ; NUM = NUM + 1
IPAPDETB.NUM = 'PROD    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'PROG    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'PURGE   ' ; NUM = NUM + 1
```

```
IPAPDETB.NUM = 'RDE     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'REAL    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'RER     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'RSU     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'RSVNONR ' ; NUM = NUM + 1
IPAPDETB.NUM = 'RSVSTRT ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SCH     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SMF     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SMS     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SQA     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SSN     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SVC     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SWAP    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SYSNAME ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SYSP    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'VAL     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'VIODSN  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'VRREGN  ' ; NUM = NUM + 1
If Substr(FMIDNUM,4,4) >= 66Ø4 then do
   IPAPDETB.NUM = 'RTLS    ' ; NUM = NUM + 1
End
IPAPDETB.Ø = NUM-1
Return
BROWSE_ISPF:         /* Browse output if ISPF is active             */
Address ISPEXEC "CONTROL ERRORS RETURN"
Address TSO
prefix = sysvar('SYSPREF')        /* tso profile prefix           */
uid    = sysvar('SYSUID')         /* tso userid                   */
If prefix = '' then prefix = uid  /* use uid if null prefix       */
If prefix <> '' & prefix <> uid then /* different prefix than uid  */
   prefix = prefix || '.' || uid /* use  prefix.uid               */
ddnm1 = 'DD'||random(1,99999)   /* choose random ddname           */
ddnm2 = 'DD'||random(1,99999)   /* choose random ddname           */
junk = msg(off)
"ALLOC FILE("||ddnm1||") UNIT(SYSALLDA) NEW TRACKS SPACE(2,1) DELETE",
      " REUSE LRECL(8Ø) RECFM(F B) BLKSIZE(312Ø)"
"ALLOC FILE("||ddnm2||") UNIT(SYSALLDA) NEW TRACKS SPACE(1,1) DELETE",
      " REUSE LRECL(8Ø) RECFM(F B) BLKSIZE(312Ø) DIR(1)",
      " DA('"||prefix||".SYSINFO." ||ddnm2|| ".ISPPLIB')"
junk = msg(on)
Newstack
/*************************/
/* SYSINFOP Panel source */
/*************************/
If Substr(ZENVIR,6,1) >= 4 then
  Queue ")PANEL KEYLIST(ISRSPBC,ISR)"
Queue ")ATTR"
Queue "  _ TYPE(INPUT)   INTENS(HIGH) COLOR(TURQ) CAPS(OFF)" ,
      "FORMAT(&MIXED)"
Queue "  | AREA(DYNAMIC) EXTEND(ON)   SCROLL(ON)"
Queue "  + TYPE(TEXT)    INTENS(LOW)  COLOR(BLUE)"
```

```
Queue "  @ TYPE(TEXT)    INTENS(LOW)  COLOR(TURQ)"
Queue "  % TYPE(TEXT)    INTENS(HIGH) COLOR(GREEN)"
Queue "  ! TYPE(OUTPUT)  INTENS(HIGH) COLOR(TURQ) PAD(-)"
Queue " Ø1 TYPE(DATAOUT) INTENS(LOW)"
Queue " Ø2 TYPE(DATAOUT) INTENS(HIGH)"
Queue " ØB TYPE(DATAOUT) INTENS(HIGH) FORMAT(DBCS)"
Queue " ØC TYPE(DATAOUT) INTENS(HIGH) FORMAT(EBCDIC)"
Queue " ØD TYPE(DATAOUT) INTENS(HIGH) FORMAT(&MIXED)"
Queue " 1Ø TYPE(DATAOUT) INTENS(LOW)  FORMAT(DBCS)"
Queue " 11 TYPE(DATAOUT) INTENS(LOW)  FORMAT(EBCDIC)"
Queue " 12 TYPE(DATAOUT) INTENS(LOW)  FORMAT(&MIXED)"
Queue ")BODY EXPAND(//)"
Queue "%BROWSE  @&ZTITLE  / /  %Line!ZLINES  %Col!ZCOLUMS+"
Queue "%Command ===>_ZCMD / /           %Scroll ===>_Z   +"
Queue "|ZDATA --------------/ /------------------------|"
Queue "|                     / /                        |"
Queue "| --------------------/-/------------------------|"
Queue ")INIT"
Queue "  .HELP = SYSINFOH"
Queue "  .ZVARS = 'ZSCBR'"
Queue "  &ZTITLE = 'SYSINFO -" OPTION "option'"
Queue "  &MIXED = MIX"
Queue "  IF (&ZPDMIX = N)"
Queue "   &MIXED = EBCDIC"
Queue "  VGET (ZSCBR) PROFILE"
Queue "  IF (&ZSCBR = ' ')"
Queue "   &ZSCBR = 'CSR'"
Queue ")REINIT"
Queue "  .HELP = SYSINFOH"
Queue "  REFRESH(ZCMD,ZSCBR,ZDATA,ZLINES,ZCOLUMS)"
Queue ")PROC"
Queue "  &ZCURSOR = .CURSOR"
Queue "  &ZCSROFF = .CSRPOS"
Queue "  &ZLVLINE = LVLINE(ZDATA)"
Queue "  VPUT (ZSCBR) PROFILE"
Queue ")END"
Queue ""
/*                                    */
"ALLOC FILE(SYSINFOP) SHR REUSE",
      " DA('"||prefix||".SYSINFO." ||ddnm2|| ".ISPPLIB(SYSINFOP)')"
"EXECIO * DISKW SYSINFOP (FINIS"
/* "FREE FI(SYSINFOP)" */
Delstack
Newstack
/*************************/
/* SYSINFOH Panel source */
/*************************/
If Substr(ZENVIR,6,1) >= 4 then
  Queue ")PANEL KEYLIST(ISRSPBC,ISR)"
Queue ")ATTR DEFAULT(!+_)"
Queue "  _ TYPE(INPUT)   INTENS(HIGH) COLOR(TURQ) CAPS(OFF)" ,
```

```
      "FORMAT(&MIXED)"
Queue "  + TYPE(TEXT)    INTENS(LOW)  COLOR(BLUE)"
Queue "  @ TYPE(TEXT)    INTENS(LOW)  COLOR(TURQ)"
Queue "  ! TYPE(TEXT)    INTENS(HIGH) COLOR(GREEN)"
Queue ")BODY EXPAND(//)"
Queue "!HELP    @&ZTITLE  / / "
Queue "!Command ===>_ZCMD / / "
Queue "+                                                                "
Queue "+EXECUTION SYNTAX:                                               "
Queue "+                                                                "
Queue "!TSO %SYSINFO <option>                                           "
Queue "+                                                                "
Queue "+VALID OPTIONS ARE 'ALL' (default), 'IPL', 'VERSION'," ||,
      " 'STOR', 'CPU',"
Queue "+                   'IPA', 'SYMBOLS', 'VMAP', 'PAGE', and 'SUB'."
Queue "+Examples:                                                       "
Queue "! TSO %SYSINFO        +(Display all Information)                 "
Queue "! TSO %SYSINFO IPL    +(Display IPL Information)                 "
Queue "! TSO %SYSINFO VERSION+(Display Version Information)             "
Queue "! TSO %SYSINFO STOR   +(Display Storage Information)             "
Queue "! TSO %SYSINFO CPU    +(Display CPU Information)                 "
Queue "! TSO %SYSINFO IPA    +(Display Initialization Information)      "
Queue "! TSO %SYSINFO SYMBOLS+(Display Static System Symbols)           "
Queue "! TSO %SYSINFO VMAP   +(Display a Virtual Storage Map)           "
Queue "! TSO %SYSINFO PAGE   +(Display Page Data Set Usage",
                              "Information)"
Queue "! TSO %SYSINFO SUB    +(Display Subsystem Information)           "
Queue "+                                                                "
Queue "@&ADLINE"
Queue ")INIT"
Queue "  .HELP = ISR10000"
Queue "  &ZTITLE = 'SYSINFO -" OPTION "option'"
Queue "  &L1 = 'SYSINFO - Author: '"
Queue "  &L2 = 'Mark Zelden'"
Queue "  &ADLINE  = '&L1 &L2'"
Queue "  &MIXED = MIX"
Queue "  IF (&ZPDMIX = N)"
Queue "   &MIXED = EBCDIC"
Queue ")END"
Queue ""
"ALLOC FILE(SYSINFOP) SHR REUSE",
      " DA('"||prefix||".SYSINFO." ||ddnm2|| ".ISPPLIB(SYSINFOH)')"
"EXECIO * DISKW SYSINFOP (FINIS"
"FREE FI(SYSINFOP)"
Delstack
"EXECIO * DISKW" ddnm1 "(FINIS"
zerrsm  = 'SYSINFO' LASTUPD
zerrlm  = 'SYSINFO -' OPTION 'option.' ,
          'Last updated on' LASTUPD ||'. Written by' ,
          'Mark Zelden.'
zerralrm = 'NO'        /* msg - no alarm */
```

```
zerrhm   = 'SYSINFOH'  /* help panel */
address ISPEXEC "LIBDEF ISPPLIB LIBRARY ID("||ddnm2||") STACK"
address ISPEXEC "SETMSG MSG(ISRZ002)"
address ISPEXEC "LMINIT DATAID(TEMP) DDNAME("||ddnm1||")"
address ISPEXEC "BROWSE DATAID("||temp") PANEL(SYSINFOP)"
address ISPEXEC "LMFREE DATAID("||temp")"
address ISPEXEC "LIBDEF ISPPLIB"
junk = msg(off)
"FREE FI("||ddnm1||")"
"FREE FI("||ddnm2||")"
Return
/* rexx */
RDATE:
/* AUTHOR: Mark Zelden                    */
/*********************************************/
/* Convert MM DD YYYY , YYYY DDD, or NNNNN to   */
/* standard date output that includes the day   */
/* of the week and the number of days (NNNNN)   */
/* from January 1, 1900. This is not the same   */
/* as the Century date!                         */
/* A parm of "TODAY" can also be passed to      */
/* the date conversion routine.                 */
/* MM DD YYYY can also be specifed as           */
/* MM/DD/YYYY or MM-DD-YYYY.                    */
/* The output format is always as follows:      */
/*      MM/DD/YYYY.JJJ NNNNN WEEKDAY            */
/* The above value will be put in the special   */
/* REXX variable "RESULT"                       */
/* example: CALL RDATE TODAY                    */
/* example: CALL RDATE 1996 300                 */
/* example: CALL RDATE 10 26 1996               */
/* example: CALL RDATE 10/26/1996               */
/* example: CALL RDATE 10-26-1996               */
/* example: CALL RDATE 35363                    */
/* result:  10/26/1996.300 35363 Saturday       */
/*********************************************/
arg P1 P2 P3
If Pos('/',P1) <> 0 | Pos('-',P1) <> 0 then do
  PX =  Translate(P1,'  ','/-')
  Parse var PX P1 P2 P3
End
JULTBL = '000031059090120151181212243273304334'
DAY.0 = 'Sunday'
DAY.1 = 'Monday'
DAY.2 = 'Tuesday'
DAY.3 = 'Wednesday'
DAY.4 = 'Thursday'
DAY.5 = 'Friday'
DAY.6 = 'Saturday'
Select
  When P1 = 'TODAY' then do
```

```
    CURDATE = date('s')
    P1 = Substr(CURDATE,5,2)
    P2 = Substr(CURDATE,7,2)
    P3 = Substr(CURDATE,1,4)
    call CONVERT_MDY
    call THE_END
  end
  When P2 = '' & P3 = '' then do
    call CONVERT_NNNNN
    call THE_END
  end
  When P3 = '' then do
    call CONVERT_JDATE
    call DOUBLE_CHECK
    call THE_END
  end
  otherwise do
    call CONVERT_MDY
    call DOUBLE_CHECK
    call THE_END
  end
end /* end select */
/* say RDATE_VAL; exit Ø  */
return RDATE_VAL
/*********************************************/
/*  E N D    O F   M A I N L I N E   C O D E  */
/*********************************************/
CONVERT_MDY:
if P1<1 | P1>12 then do
  say 'Invalid month passed to date routine'
  exit 12
end
if P2<1 | P2>31 then do
  say 'Invalid day passed to date routine'
  exit 12
end
if (P1=4 | P1=6 | P1=9 | P1=11) & P2>3Ø then do
  say 'Invalid day passed to date routine'
  exit 12
end
if P3<19ØØ | P3>2Ø99 then do
  say 'Invalid year passed to date routine'
  exit 12
end
BASE   = Substr(JULTBL,((P1-1)*3)+1,3)
if (P3//4=Ø & P3<>19ØØ) then LEAP= 1
  else LEAP = Ø
if P1 > 2 then BASE = BASE+LEAP
JJJ = BASE + P2
MM   = P1
DD   = P2
```

```
YYYY = P3
return
CONVERT_NNNNN:
if P1<1 | P1>73Ø49 then do
  say 'Invalid date passed to date routine. NNNNN must be 1-73Ø49'
  exit 12
end
/* Determine YYYY and JJJ */
if P1>365 then P1=P1+1
YEARS_X4=(P1-1)%1461
JJJ=P1-YEARS_X4*1461
EXTRA_YEARS=(JJJ*3-3)%1Ø96
JJJ=JJJ-(EXTRA_YEARS*1Ø96+2)%3
YYYY=YEARS_X4*4+EXTRA_YEARS+19ØØ
P1 = YYYY ; P2 = JJJ ;  call CONVERT_JDATE
CONVERT_JDATE:
if P1<19ØØ | P1>2Ø99 then do
  say 'Invalid year passed to date routine'
  exit 12
end
if P2<1 | P2>366 then do
  say 'Invalid Julian date passed to date routine'
  exit 12
end
if (P1//4=Ø & P1<>19ØØ) then LEAP= 1
  else LEAP = Ø
ADJ1 = Ø
ADJ2 = Ø
Do MM = 1 to 11
   VAL1 = Substr(JULTBL,((MM-1)*3)+1,3)
   VAL2 = Substr(JULTBL,((MM-1)*3)+4,3)
   if MM >=2 then ADJ2 = LEAP
   if MM >=3 then ADJ1 = LEAP
   if P2 > VAL1+ADJ1 & P2 <= VAL2+ADJ2 then do
        DD = P2-VAL1-ADJ1
        MATCH = 'Y'
        leave
   end
end
if MATCH <> 'Y' then do
    MM = 12
    DD = P2-334-LEAP
end
YYYY = P1
JJJ  = P2
return
DOUBLE_CHECK:
if MM = 2 then do
   if DD > 28 & LEAP = Ø then do
     say 'Invalid day passed to date routine'
     exit 12
```

```
   end
   if DD > 29 & LEAP = 1 then do
     say 'Invalid day passed to date routine'
     exit 12
   end
end
if LEAP = Ø & JJJ > 365 then do
  say 'Invalid Julian date passed to date routine'
  exit 12
end
return
THE_END:
YR_19ØØ = YYYY-19ØØ
NNNNN = (YR_19ØØ*365) +(YR_19ØØ+3)%4 + JJJ
if YYYY > 19ØØ then NNNNN = NNNNN-1
INDEX   = NNNNN//7  /* index to DAY stem */
WEEKDAY =  DAY.INDEX
DD      = Right(DD,2,'Ø')
MM      = Right(MM,2,'Ø')
YYYY    = Strip(YYYY)
NNNNN   = Right(NNNNN,5,'Ø')
JJJ     = Right(JJJ,3,'Ø')
RDATE_VAL = MM||'/'||DD||'/'||YYYY||'.'||JJJ||' '||NNNNN||' '||WEEKDAY
return
```

EXAMPLE OUTPUT

```
********************************************************************************
*********************  SYSTEM INFORMATION  *********************************
********************************************************************************
Today is Friday Ø2/16/2ØØ1 (2ØØ1.Ø47). The local time is 14:12:59.
The last IPL was Sunday Ø1/28/2ØØ1 (2ØØ1.Ø28) at Ø4:34:24 (19 days ago).
The IPL was done with CLPA.
The system IPL address was B1EØ (RESEA1).
The IPL LOAD PARM used was Ø72AP1M1 (Ø72A P1 M 1).
The system is running in ESA/39Ø mode (ARCHLVL = 1).
The Processor name is IBMSYSB. The LPAR name is SYSE (LPAR #6).
The SYSPLEX name is P1.
The GRS system id is SYSE. The SMF system id is SYSE.
The currently active IODF dataset is SYS1.IODF45.
  Configuration ID = SYSE      EDT ID = E1
  TOKEN: Processor  Date      Time      Description
         IBMSYSB    Ø1-Ø1-14  16:58:54  SYS1    IODF45
The Master Catalog is SYSICF.MASTER on CATØØ1.
The OS version is OS/39Ø Ø2.1Ø.ØØ - FMID HBB77Ø3.
The primary job entry subsystem is JES2.
The JES2 level is JES2 OS 2.1Ø. The JES2 Node name is USMSZ1J2.
The security software is OS/39Ø Security Server (RACF).
The FMID is HRF77Ø3.
The DFSMS level is 2.1Ø.Ø.
```

```
The TSO level is 2.6.Ø.
The ISPF level is OS/39Ø Ø2.1Ø.ØØ.
The VTAM level is 5.1.Ø.
The real storage size at IPL time was 512M.
The potential real storage size is 1Ø24M.
The expanded storage size is 512M.
The private area size <16M is 9216K.
The private area size >16M is 1859M.
The CSA size <16M is 3Ø16K.
The CSA size >16M is 96980K.
The SQA size <16M is 1856K.
The SQA size >16M is 21892K.
The maximum V=R region size is 128K.
The default V=R region size is 64K.
The maximum V=V region size is 9192K.
The CPU model number is 9672.
The number of online CPUs is 2.
The CPU serial number for CPU Ø is Ø628149672 (62814), version code 84.
The CPU serial number for CPU 1 is 1628149672 (62814), version code 84.
The service units per second per online CPU is 5643.74.
The approximate total MIPS (SUs/SEC / 48.5 * #CPUs) is 232.73.
Central Processing Complex (CPC) Node Descriptor:
  CPC ND = ØØ9672.R46.IBM.Ø2.ØØØØØØØ12814
  CPC ID = ØØ
  Type(ØØ9672) Model(R46) Manufacturer(IBM) Plant(Ø2) Seq
Num(ØØØØØØØ12814)
Initialization information from the IPA:
  IPLPARM = Ø72AP1M1 (merged)
  IPL load parameter dataset name: SYS1.IPLPARM
  HWNAME=IBMSYSB   LPARNAME=SYSE      VMUSERID=
  LOADxx parameters (LOADP1):
   *--+----1----+----2----+----3----+----4----+----5----+----6----+----7
    ARCHLVL  1
    IEASYM   P1
    IODF     ** SYS1     SYSE     E1 Y
    NUCLEUS  1
    NUCLST   ØØ
    PARMLIB  SYS1.PARMLIB.SYSE
    PARMLIB  SYS1.PARMLIB
    PARMLIB  SYS1.IBM.PARMLIB
    SYSCAT   CATØØ1113CSYSICF.MASTER
    SYSPARM  (P1,E1,L)
  IEASYSxx parameters:
(Source)
    ALLOC=ØØ,
IEASYSP1
    APG=Ø7,
IEASYSP1
    CLOCK=P1,
IEASYSP1
    CMB=(UNITR,COMM,GRAPH,CHRDR),
```

```
IEASYSP1
    CMD=E1,
IEASYSP1
    CON=(P1,NOJES3),
IEASYSP1
    COUPLE=ØØ,
IEASYSP1
    CSA=(25ØØ,96ØØØ),
IEASYSE1
    CSCBLOC=ABOVE,
Default
    DIAG=ØØ,
IEASYSP1
    DUMP=DASD,
IEASYSP1
    FIX=ØØ,
IEASYSP1
    GRS=TRYJOIN,
IEASYSP1
    GRSCNF=ØØ,
Default
    GRSRNL=EXCLUDE,
IEASYSP1
    IPS=ØØ,
Default
    LNK=ØØ,
Default
    LNKAUTH=LNKLST,
IEASYSP1
    LOGCLS=L,
IEASYSP1
    LOGLMT=999999,
IEASYSP1
    LOGREC=LOGSTREAM,
IEASYSP1
    LPA=E1,
IEASYSP1
    MAXCAD=25,
Default
    MAXUSER=35Ø,
IEASYSP1
    MLPA=P1,
IEASYSP1
    MSTRJCL=P1,
IEASYSP1
    NONVIO=(SYS1.PAGE.SYSE.LOCAL1),
                            IEASYSE1
    NSYSLX=55,
Default
    OMVS=P1,
IEASYSP1
```

```
    OPI=YES,
IEASYSP1
    OPT=ØØ,
IEASYSP1
    PAGE=(SYS1.PAGE.SYSE.PLPA,
          SYS1.PAGE.SYSE.COMMON,
          SYS1.PAGE.SYSE.LOCAL1,
          SYS1.PAGE.SYSE.LOCAL2)
IEASYSE1
    PAGTOTL=(1Ø,5),
IEASYSP1
    PAK=ØØ,
IEASYSP1
    PLEXCFG=MULTISYSTEM,
IEASYSP1
    PROD=P1,
IEASYSP1
    PROG=P1,
IEASYSP1
    RDE=NO,
Default
    REAL=128,
IEASYSP1
    RER=NO,
Default
    RSU=32,
IEASYSP1
    RSVNONR=5,
IEASYSP1
    RSVSTRT=5,
IEASYSP1
    SCH=P1,
IEASYSP1
    SMF=P1,
IEASYSP1
    SMS=P1,
IEASYSP1
    SQA=(15,1ØØ),
IEASYSE1
    SSN=EØ,
IEASYSP1
    SVC=EØ,
IEASYSP1
    SYSNAME=SYSE,
IEASYSP1
    SYSP=(P1,EØ),
Operator
    VAL=ØØ,
IEASYSP1
    VIODSN=SYS1.STGINDEX.SYSE,
IEASYSP1
```

```
    VRREGN=64
IEASYSP1
Static System Symbol Values:
  &SYSCLONE. = E
  &SYSNAME.  = SYSE
  &SYSPLEX.  = P1
  &SYSR1.    = RESEA1
  &SYSR2.    = RESEA2
Virtual Storage Map:
     Storage Area     Start      End           Size      Used     Conv
              PSA     00000000   00000FFF        4K
           System     00002000   00005FFF       16K
      Private V=R     00006000   00025FFF      128K
      Private V=V     00006000   008FFFFF     9192K
              CSA     00900000   00BF1FFF     3016K     1920K
             MLPA     N/A        N/A             0K
             FLPA     00BF2000   00BFCFFF       44K
             PLPA     00BFD000   00DFBFFF     2044K
              SQA     00DFC000   00FCBFFF     1856K      452K       0K
      R/W Nucleus     00FCC000   00FD9B2F       55K
      R/O Nucleus     00FDA000   00FFFFFF      152K (Spans 16M line)
            16M line ----------------------------
 Ext. R/O Nucleus     01000000   0161C0AF     6256K (Total 6408K)
 Ext. R/W Nucleus     0161D000   01964FFF     3360K
         Ext. SQA     01965000   02EC5FFF    21892K    14742K       0K
        Ext. PLPA     02EC6000   05E46FFF    48644K
        Ext. FLPA     05E47000   05E49FFF       12K
        Ext. MLPA     05E4A000   05E4AFFF        4K
         Ext. CSA     05E4B000   0BCFFFFF    96980K    49676K
     Ext. Private     0BD00000   7FFFFFFF     1859M
Page Data Set Usage:
  Type     Full   Slots  Dev   Volser  Data Set Name
  PLPA     100%     360  B1E5  EPAGE1  SYS1.PAGE.SYSE.PLPA
  COMMON    19%   63000  B1E5  EPAGE1  SYS1.PAGE.SYSE.COMMON
  LOCAL NV  20%  540000  B1E6  EPAGE2  SYS1.PAGE.SYSE.LOCAL1
  LOCAL     23%  540000  B1E7  EPAGE3  SYS1.PAGE.SYSE.LOCAL2
Subsystem Communications Vector Table:
  Name   Hex        SSCTADDR   SSCTSSVT   SSCTSUSE   SSCTSUS2   Status
  JES2   D1C5E2F2   00CCF940   00C781C8   00C78D18   00C78718   Active
  MSTR   D4E2E3D9   00CCF8F8   00CCF748   00000000   00000000   Active
  CASF   C3C1E2C6   00CE8058   00000000   00FBC7B0   00CE86C0   Inactive
  TRCE   E3D9C3C5   00CE8698   094F5038   00000000   00000000   Active
  ACF2   C1C3C6F2   00CE8670   00FBC308   00F9F668   094E70B0   Active
  MACS   D4C1C3E2   00CE8818   00000000   00000000   00000000   Inactive
  CPF    C3D7C640   00CE87F0   00000000   00000000   00000000   Inactive
  SMS    E2D4E240   00CCF91C   00CCF250   00000000   00000000   Active
  TNF    E3D5C640   00CCF964   00000000   00CCF898   00002C00   Inactive
  VMCF   E5D4C3C6   00CCF988   00000000   00CAE048   00002D00   Inactive
```

Mark Zelden
Systems Programmer (USA)

October 1999 – September 2002 index

Items below are references to articles that have appeared in *MVS Update* since October 1999. References show the issue number followed by the page number(s). Subscribers can download copies of all issues in Acrobat PDF format from Xephon's Web site.

UFD Solutions AG has announced HSMSCF (a Control Facility for DFSMS/HSM), which provides a workbench for Storage Administrators, Systems Engineers, and Operators. HSMSCF simplifies many back-up, restore, and recall functions to save time, reduce errors, and minimize outages.

HSMSCF generates utility job streams, which can be run automatically as part of the standard operational cycle and includes facilities to display and interpret objects under the control of DFSMS/HSM in the Control Data Sets (MCDS, BCDS and OCDS). HSMSCF provides assistance in disaster recovery of an entire environment as well as in the daily administration of such an environment. HSMSCF produces an audit trail about performed actions.

For further information contact:
UBS AG, Hochstrasse 16, CH-4002 Basel, Switzerland.
URL: http://www.ufd.ch.

* * *

NewEra Software is shipping Stand Alone Environment (SAE) Release 11, providing five integrated applications designed to increase the ability to diagnose, repair, and recover from a system failure. It can also be used as an alternative to the floor system provided at the disaster recovery site.

The five applications are Action Services, FAST DASD ERASE, Stand Alone Restore, Hardware Confirmation, and Image Services. Fast DASD Erase gets a new Burst Mode Erase mode, designed for use with emulated RAID DASD subsystems and promising increased reliability and performance. It will limit the size of the CCW chain, resulting in many I/O starts, but it is said to provide greater load balancing and overall better erase times.

SAE 11's IPL Analysis has been changed to fully support OS/390 and z/OS IEASYSxx settings of symbolics for substitution in other IPL parameter settings. The Image Analysis component now inspects IEASYMxx members to determine the IEASYSxx members being used. HWNAME/ LPARNAME/VMUSERID filtering is fully supported within IEASYMxx.

NewEra has also announced IMAGE Focus 4.3, which systematically identifies, locates, and inspects thousands of system parameters and resources that make up an OS/390 or z/OS sysplex, including the operating system, JES2/3, VTAM, and TCP/IP.

For further information contact:
NewEra Software, Morgan Hill, CA 95037, USA.
Tel: (408) 201 7000.
URL: http://www.newera.com.

* * *

Computer Associates has announced Unicenter Database Management Solutions for IMS for z/OS and OS/390 Release 4.3, with new products and enhancements that promise improved performance, data availability, and DBA productivity, plus reduced complexity and cost of managing IMS environments.

For further information contact:
Computer Associates, One Computer Associates Plaza, Islandia, NY 11749, USA.
Tel: (631) 342 5224.
URL: http://ca.com.