155 MVS update

August 1999

In this issue

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

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

Contributions

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

Editor
Jaime Kaminski

Disclaimer

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

MVS Update on-line

Code from *MVS Update* can be downloaded from our Web site at http://www.xephon.com; you will need the user-id shown on your address label.

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Printed in England.

Automated system or TSO shutdown

INTRODUCTION

The following utility is an exit designed to aid the operators when shutting down the system or TSO address space. Whenever an operator issues ‘P TSO’, and assuming users are still logged on, the operator will receive message IKT010D to which they reply FSTOP or SIC to end. Tise exit will auto reply to the IKT010D with FSTOP, thus ending the TSO ASID. To enable the exit, you need to add the message ID and the ENDTSO exit to the MPFLSTxx member.

ENDTSO

```
//jobcard ….
//DOIT   EXEC ASMHCL,PARM.C='NODECK',REGION=4Ø96K,
// PARM.L='RENT,REUS'
//C.SYSPRINT DD SYSOUT=*
//C.SYSIN  DD *
 TITLE 'ENDTSO - COMMUNICATIONS TASK INSTALLATION EXIT FOR IKTØ1ØD'
***START OF SPECIFICATIONS*********************************************
* MODULE NAME      =  ENDTSO                                           *
*                                                                      *
* DESCRIPTIVE NAME =  COPY FROM IECAWAIT FROM SYS1.SAMPLIB             *
*                                                                      *
* FUNCTION         =  LOOK FOR TSO SHUTDOWN MESSAGE AND REPLY TO       *
*                     FSTOP TO FORCE IT TO QUIT.                       *
*                                                                      *
* OPERATION        =  WAIT FOR THE TSO SHUTDOWN MESSAGE AND THEN       *
*                     ANSWER THE REPLY WITH FSTOP.                     *
*                     THIS MAKES SURE THAT TSO QUITS WHEN THE          *
*                     OPERARORS WANT IT DOWN.                          *
*                                                                      *
* INVOCATION       =  TO INVOKE, SPECIFY THIS PROGRAM AS AN MPF EXIT   *
*                     FOR IKTØ1ØD IN YOUR MPFLSTXX MEMBER OF           *
*                     SYS1.PARMLIB.                                    *
*                                                                      *
* ENTRY POINT      =  SALSHTSO                                         *
*                                                                      *
* INPUT DATA       =  R1 POINTS TO THE ADDRESS OF THE CTXT             *
*                     R13 ADDRESS OF STANDARD SAVE AREA                *
*                     R15 ENTRY POINT                                  *
*                                                                      *
* REGISTERS SAVED  =  RØ - R15                                         *
*                                                                      *
* REGISTER USAGE   =  RØ    - PARAMETER REGISTER                       *
```

```
*                     R1    - PARAMETER REGISTER                      *
*                     R2    - CTXT                                    *
*                     R3    - WORK REGISTER                           *
*                     R4    - WORK REGISTER                           *
*                     R5    - MESSAGE TEXT AREA WITHIN CTXT           *
*                     R6    - WORK REGISTER                           *
*                     R7    - WORK REGISTER                           *
*                     R8    - WORK REGISTER                           *
*                     R9    - RETURN REGISTER FOR SUBROUTINES         *
*                     R10 - NOT USED                                  *
*                     R11 - MODULE DATA REGISTER                      *
*                     R12 - MODULE BASE REGISTER                      *
*                     R13 - POINTER TO A STANDARD SAVE AREA           *
*                     R14 - RETURN POINT                              *
*                     R15 - PARAMETER REGISTER                        *
*                                                                     *
*   CONTROL BLOCKS    =                                               *
*     NAME     MAPPING MACRO   REASON USED                  USAGE     *
*     --     -------   ------                 ---    *
*     CTXT      IEZVX100        USER EXIT PARAMETER LIST     R,W      *
*                                                                     *
*   MESSAGES          =  USER001I                                     *
*                                                                     *
*   MODULE TYPE       = CSECT                                         *
*                                                                     *
*   ATTRIBUTES        =  REENTRANT, REUSABLE, AMODE 31, RMODE ANY     *
***********************************************************************
*         REGISTER ASSIGNMENTS                                        *
***********************************************************************
R0       EQU   0                       REGISTER 0
R1       EQU   1                       REGISTER 1
R2       EQU   2                       REGISTER 2
R3       EQU   3                       REGISTER 3
R4       EQU   4                       REGISTER 4
CTXTPTR  EQU   5                       REGISTER 5 - POINTS TO CTXT
R6       EQU   6                       REGISTER 6
R7       EQU   7                       REGISTER 7
R8       EQU   8                       REGISTER 8
R9       EQU   9                       REGISTER 9
R10      EQU   10                      REGISTER 10 - CURRENTLY UNUSED
DATAREG  EQU   11                      REGISTER 11 - DYNAMIC DATA AREA
R12      EQU   12                      REGISTER 12
BASEREG  EQU   12                      REGISTER 12 - MODULE BASE
R13      EQU   13                      REGISTER 13
R14      EQU   14                      REGISTER 14
R15      EQU   15                      REGISTER 15
         SPACE 1
***********************************************************************
*        EQUATES                                                      *
***********************************************************************
```

```
ROUTBYTE EQU   16                      16 BYTES OF ROUTING CODES
SPINPRVT EQU   230                     SUBPOOL VALUE FOR GETMAIN
         EJECT
*********************************************************************
*        STANDARD ENTRY LINKAGE                                     *
*********************************************************************
ENDTSO   CSECT
ENDTSO   AMODE 31                      31-BIT ADDRESSING MODE
ENDTSO   RMODE ANY                     31-BIT RESIDENCE
         SPACE 1
         STM   R14,R12,12(R13)         SAVE CALLER'S REGISTERS
         BALR  BASEREG,R0              ESTABLISH MODULE BASE
         USING *,BASEREG               ESTABLISH ADDRESSABILITY
         L     CTXTPTR,0(,R1)          ESTABLISH ADDRESSABILITY
         USING CTXT,CTXTPTR            TO THE CTXT
*********************************************************************
*        DETERMINE IF THIS IS OUR MESSAGE, IKT010D                  *
*********************************************************************
         SPACE 1
         L     R2,CTXTTXPJ             MESSAGE ATTRIBUTES AREA
         USING CTXTATTR,R2             ESTABLISH ADDRESSABILITY
         LA    R4,CTXTTMSG             ADDRESS OF TEXT AREA
*********************************************************************
*        REGISTER 4 POINTS TO THE MESSAGE TEXT BEING PROCESSED      *
*********************************************************************
         CLC   0(L'IKT010D,R4),IKT010D  MESSAGE IKT010D?
         BNE   FINISHED                NO, EXIT
*********************************************************************
*        WORKING WITH MESSAGE IKT010D THEREFORE REPLY TO THE        *
*        MESSAGE AND ISSUE ANOTHER STATING TSO IS BEING STOPPED     *
*        FIRST, OBTAIN DYNAMIC STORAGE FOR THE MGCRE AND WTO        *
*        PARAMETER LISTS.                                           *
*********************************************************************
         LA    R0,DATAEND                LENGTH OF DATA AREAS
         GETMAIN  RU,LV=(R0),SP=SPINPRVT  OBTAIN DYNAMIC STORAGE
         LR    DATAREG,R1              ADDRESS RETURNED IN R1
         USING DATAAREA,DATAREG        ADDRESSABILITY TO DYNAMIC      X
                                       STORAGE
         OI    CTXTRFB2,CTXTRHCO       SUPPRESS MESSAGE IEF235D
         BAL   R9,ISSMGCRE             REPLY TO IEF235D 'R XX,NO'
         BAL   R9,ISSUWTO              ISSUE MESSAGE USER001I
*********************************************************************
*        RETURN THE DYNAMIC STORAGE OBTAINED                        *
*********************************************************************
         LA    R0,DATAEND                LENGTH OF DATA AREAS
         FREEMAIN  RU,LV=(R0),A=(DATAREG),SP=SPINPRVT                 X
                                        FREE THE DYNAMIC STORAGE AREA
*********************************************************************
*        STANDARD EXIT LINKAGE, AND EXIT FROM THIS MODULE           *
*********************************************************************
```

```
FINISHED EQU   *
*        WTO   'SALAH WAS HERE',ROUTCDE=2,DESC=6                   ØØ1
         LM    R14,R12,12(R13)          RESTORE CALLER'S              X
                                        REGISTERS
         BR    R14                      RETURN TO CALLER
         EJECT
*********************************************************************
*   PROCEDURE - ISSMGCRE                                            *
*                                                                   *
*   FUNCTION  - REPLIES TO MESSAGE IKDØ1ØD WITH A 'FSTOP' RESPONSE  *
*                                                                   *
*   INPUT     - REGISTER 11 POINTS TO THE DYNAMIC AREA WHICH CONTAINS *
*               STORAGE FOR THE MGCRE PARAMETER LIST                *
*                                                                   *
*   OUTPUT    - A REPLY TO IKTØ1ØD IS ISSUED VIA MGCRE              *
*                                                                   *
*   NOTES     - LIST AND EXECUTE FORMS OF MGCRE ARE REQUIRED        *
*             - THE LONG FORM OF THE REPLY VERB IS USED.            *
*********************************************************************
         SPACE 1
ISSMGCRE EQU   *
         LA    R7,DYNCMDR1              ADDRESS COMMAND REPLY AREA
         MVI   Ø(R7),BLANK              BLANK FIRST MESSAGE CHARACTER
         MVC   1(CMDRLENG-1,R7),Ø(R7)   BLANK ENTIRE MESSAGE FIELD
         MVC   Ø(L'CMDLNGTH,R7),CMDLNGTH  INITIALIZE COMMAND LENGTH
         MVC   L'CMDLNGTH(L'TXINSRT1,R7),TXINSRT1
*                                         MOVE REPLY VERB TO AREA
*********************************************************************
*        CTXTRPYI CONTAINS THE EBCDIC REPLY ID VALUE. CTXTRPYL      *
*        CONTAINS THE LENGTH OF THE REPLY ID. USE THESE FIELDS TO   *
*        MOVE THE REPLY ID INTO THE TEXT AREA                       *
*********************************************************************
         LH    R8,CTXTRPYL              REPLY ID LENGTH
         BCTR  R8,Ø                     DECREMENT BY 1 FOR EXECUTE
         EX    R8,MOVERYID              MOVE REPLY ID VALUE INTO TEXT
         LA    R8,TXI1LENG+1(R8,R7) POINT TO CHARACTER AFTER ID
         MVC   Ø(L'TXINSRT3,R8),TXINSRT3  COMPLETE REPLY TEXT
         LA    R4,DYNMGCRE              ADDRESS MGCRE PARAMETER LIST
         MVC   Ø(REPLEN,R4),REPAREA     COPY MGCRE LIST TO DYNAMIC
         XR    R6,R6                    CONSOLE ID VALUE OF ZERO
         MGCRE TEXT=(R7),CONSID=(R6),TOKEN=CTXTTOKN,                  X
               MF=(E,(R4))
         BR    R9                       RETURN TO CALLER
*********************************************************************
*        OBJECT OF AN EXECUTE                                       *
*********************************************************************
MOVERYID MVC   TXINSRT2-CMDREPLY(Ø,R7),CTXTRPYI    MOVE REPLY ID
*                                        INTO REPLY COMMAND
         EJECT
*********************************************************************
```

```
*   PROCEDURE - ISSUWTO                                                 *
*                                                                       *
*   FUNCTION  - ISSUES A MESSAGE INFORMING OPERATOR THAT A JOB IS       *
*               BEING CANCELLED                                         *
*                                                                       *
*   INPUT     - NONE                                                    *
*                                                                       *
*   OUTPUT    - MESSAGE STATING JOB CANCELLED                           *
************************************************************************
         SPACE 1
ISSUWTO  EQU   *
************************************************************************
*        INITIALIZE MESSAGE TEXT FOR TEXT= PARAMETER ON WTO             *
************************************************************************
         LA    R8,DYNAMTXT              DYNAMIC MESSAGE AREA
         MVC   Ø(USERLENG,R8),USERMSG1  INITIALIZE MESSAGE TEXT
         MVC   DYNUSERS,USERSTAT        MOVE WTO STATIC AREA
         WTO   TEXT=(R8),ROUTCDE=(2),DESC=(2),MF=(E,DYNUSERS)
*                                       ISSUE THE MESSAGE
         BR    R9                       RETURN TO CALLER
         EJECT
************************************************************************
*        CHARACTER CONSTANTS                                            *
************************************************************************
IKTØ1ØD  DC    CL8'IKTØ1ØD '            MESSAGE IKTØ1ØD
         SPACE 1
************************************************************************
*        LENGTH AND TEXT OF REPLY TO WTOR                               *
*        MAXIMUM REPLY ID VALUE IS 9999                                 *
************************************************************************
CMDREPLY EQU   *
CMDLNGTH DC    AL2(ACMDLEN)             LENGTH OF WTOR REPLY COMMAND
TXINSRT1 DC    CL6'REPLY '              FIRST STATIC INSERT
TXI1LENG EQU   *-CMDREPLY               OFFSET TO TXINSRT1
TXINSRT2 DC    CL4'XXXX'                PLACE REPLY ID HERE
TXINSRT3 DC    C',FSTOP'                FINAL STATIC INSERT
ACMDLEN  EQU   *-TXINSRT1               ACTUAL LENGTH OF COMMAND   ØØ1
CMDRLENG EQU   *-CMDREPLY               COMMAND REPLY LENGTH
         EJECT ,
************************************************************************
*        LIST FORM OF MGCRE MACRO (STATIC)                              *
************************************************************************
USERREP  DS    ØH
REPAREA  MGCRE MF=L                     LIST FORM OF MACRO
REPLEN   EQU   *-USERREP                LENGTH OF MGCRE PARAMETER LIST
************************************************************************
*        LIST FORM OF WTO MACRO (STATIC)                                *
************************************************************************
USERSTAT DS    ØH
         WTO TEXT=USERMSG1,ROUTCDE=(11),DESC=(6),MF=L
```

```
CNCLMSGL EQU   *-USERSTAT               LENGTH OF PARAMETER LIST
         SPACE 1
*********************************************************************
*        CONSTANTS and equates                                      *
*********************************************************************
BLANK    EQU   X'4Ø'                   BLANK CHARACTER
         SPACE 1
*********************************************************************
*        LAYOUT OF MESSAGE TEXT FOR USERØØ1I                        *
*********************************************************************
USERMSG1 DS    ØH
USERM1LN DC    AL2(USERLENG-2)         LENGTH OF MESSAGE (IN HEX)
USERM1S1 DC    C'*** TSO *** WILL STOP NOW!!! ***'
USERLENG EQU   *-USERMSG1              LENGTH OF AREA FOR GETMAIN
         EJECT
*********************************************************************
*        STORAGE DEFINITIONS                                        *
*********************************************************************
DATAAREA DSECT
         DS    ØF
DYNMGCRE DS    CL(REPLEN)              DYNAMIC MGCRE AREA
DYNAMTXT DS    CL(USERLENG)            DYNAMIC MESSAGE TEXT AREA
DYNUSERS DS    CL(CNCLMSGL)            WTO MACRO AREA
DYNUMSG1 DS    CL(USERLENG)            DYNAMIC USERØØ1I MESSAGE
DYNCMDR1 DS    CL(CMDRLENG)            DYNAMIC COMMAND REPLY AREA
         DS    ØH
         ORG
DATAEND  EQU   *-DATAAREA
         SPACE 2
*********************************************************************
*         MAPPING OF THE MESSAGE TEXT FOR THE WTO TEXT= PARAMETER    *
*********************************************************************
MSGTXT   DSECT
MSGLEN   DS    CL2                     LENGTH OF MESSAGE TEXT
MSGID    DS    CL8                     MESSAGE ID
JOBDATA  DS    CL8                     JOB DATA (ID AND NAME)
         ORG   MSGTXT
         DS    CL4
WTORID   DS    CL8
         PRINT NOGEN
         IEZVX1ØØ                      CTXT
         END   ENDTSO
//L.SYSLMOD DD DSN=xxxxx.xxxx.LOAD,DISP=OLD
//L.SYSIN DD *
  SETCODE AC(1)
  NAME ENDTSO(R)
/*
```

Salah Balboul
Systems Programmer (USA)

A modification to indicate the maintenance level

INTRODUCTION

In an OS/390 environment where multiple machines (or LPAR machine images) co-exist, it becomes cumbersome for a system programmer to keep track of the maintenance level applied to a certain system. At our shop for instance we differentiate between cloned systems based on the reason the clone was built. We differentiate between:

- *S-Clone* – a fully functioning new OS/390 build for a customer, for example, a Year 2000 application environment. This type of clone will run in a separate LPAR or on another machine. An entirely new SMP/E environment is created so that different PTFs or even complete other Program Product (PP) levels can be installed. The S refers to system.
- *M-Clone* – this is a clone created from our main installation machine for the testing of maintenance before propagating it through our different Quality Assurement (QA) steps. All software upgrades as well as all PTFs are first checked on this clone in a flip-flop fashion. See also: *MVS/ESA configuration management redesign MVS update*, Issue 117, June 1996, pages 27-42, and *More on configuration management redesign, MVS update,* Issue 119, August 1996, pages 56-57.

  This type of clone is created by copying the resident volumes – who contain only the target datasets – from a stand-alone system (see also *Building a one-pack MVS system MVS update*, 115, April 1996, pages 7-20). Afterwards, the SMP/E environment is backed up and maintenance is applied using the standard procedures with, as the most significant difference, all the DDDEFs pointing to the cloned volume and not to the active system files.
- *C-Clone* – this is maintenance clone for one of our customers. In our design, all machines are defined as customers. Since they are not all on the same level, we maintain a complete SMP/E environment for each of them. In the case that a PTF that can have

a high pervasive impact cannot be tested fully on the installation machine, we flip-flop the resident volume of the production or test machines before applying the fixes.

- *F-Clone* – sometimes it happens that one of the four Installation Packaging Offerings (IPOs) like CICS, DBD (IMS and/or DB2) or NCP has to be installed on an existing machine. This implies the set-up of an SMP/E environment on the installation machine and the copying of the necessary files. The F in the naming convention stands for function.

In the first three cases we end up with different IPL addresses, the only item we could not solve automatically. Since we are working with a large number of machines in a 24 by 7 environment, it could possibly be that a wrong IPL address was used for one of the MVS systems after cloning. In a large LPAR environment like ours, it is not always clear who performed the last changes. In order to solve this question we decided to implement a user modification as part of our ISPF/PDF application that does the cloning and the maintenance.This usermod inserts the 'cloning level' automatically in the nucleus of OS/390 after a new system is generated.

The field we use is called CVTVERID and is to be found in the Communications Vector Table (CVT). According to the documentation in OS/390 V2R4 MVS Data Areas Volume 1 (ABEP-DALT) SY28-1164, this field contains 'Optional user personalization of software system version (MDC415)' Although we think that this field is has not changed for a long time, the bytes were reserved in the oldest (1983) *Data Areas* handbook we could find: *OS/VS2 System Programming Library: Debugging Handbook Volume 2* [GC28-0709].

During IPL, the system issues message IEA247I during NIP. If the field is not changed, the system issues the message with the Function Management Identification (FMID), otherwise CVTVERID is used.

We use the 8-byte field as follows. The first byte indicates the type of clone (S for system, M for maintenance and C for customer). The next three bytes contain the customer identification (this is part of the standards of our site, every machine has a unique 3-byte &SYSTEML. and a unique 1-byte &SYSTEMS. identification). Bytes 5 and 6 are used to indicate the last two bytes of the year in EBCDIC. Since we do not do calculations with this value because it is only meant for

display reasons a Year 2000 problem cannot occur. A value of 01 (X'F0F1') indicates 2001. The next byte indicates the month in hexadecimal but written in EBCDIC. The value 1 (X'F1') is January while A (X'C1') for instance is October. The last byte is the sequence number in hexadecimal. Since it seems unlikely that we will clone the same system more than 16 times a month this should be sufficient, if it is not, we will look for a further refinement. If we see for example MJED98B2 appear, we know that it is the third (the last character) maintenance clone (from the M at the front) of the installation machine (JED) build in November 1998.

The advantage of this method is that the 'clone level' not only appears in the spool but that it can also be checked using a small REXX or Assembler program. Normally we always code programs that access OS/390 control blocks in Assembler but because of the simplicity of displaying the contents of a field in the CVT prefix we used REXX this time.

REXX PROGRAM SAMPLE

```
/* REXX program to display the contents of the CVTVERID field        */
/* and the FMID. Refer to Data Area's PSA and CVT                    */

cvt = storage(10,4)                    /* Pick-up pointer from PSA    */
d_cvt = c2d(cvt)                       /* Decimal CVT pointer         */
cvtpfx = d_cvt - x2d(100)              /* Point to CVT prefix         */
                                       /* Get FMID                    */
cvtprodi = storage(c2x(d2c(cvtpfx+x2d(E0))), 8)
                                       /* Get CVTVERID field          */
cvtverid = storage(c2x(d2c(cvtpfx+x2d(E8))), 8)
                                       /* Let the world know          */
say 'OS/390 FMID =' cvtprodi
say 'OS/390 Cloning level =' cvtverid

exit
```

THE USER MODIFICATION

```
//*
//* CUSTOMER SOFTWARE VERSION ZAP INTO THE NUCLEUS
//* BECOMES PART OF THE CVT
//*
//SMPE     EXEC PGM=GIMSMP
//SMPCSI   DD  DSN=JED.MVS.V240.GLOBAL.CSI,DISP=SHR
//SMPHOLD  DD  DUMMY
//SMPCNTL  DD  *
```

```
  SET BDY(MVST100).
     RESTORE S(UMOD001)         /* REMOVE USERMOD IF APPLIED  */ .
  RESETRC                       /* RESET RETURN CODE          */ .
  SET BDY(GLOBAL) .
    REJECT  S(UMOD001) BYPASS(APPLYCHECK) .
  RESETRC                       /* RESET RETURN CODE          */ .
  SET BDY(GLOBAL) .
    RECEIVE S(UMOD001)          /* RECEIVE USERMOD            */ .
  SET BDY(MVST100) .
    APPLY S(UMOD001)            /* APPLY USERMOD              */ .
    LIST  SYSMOD(UMOD001)       /* LIST USERMOD               */ .
/*
//*
//SMPPTFIN DD  *                   DATA,DLM=$$
++ USERMOD (UMOD001) .
++ VER (Z038) FMID(JBB6604).
++ ZAP(IEAVCVT) DISTLIB(AOSC5) /*

    8 BYTES ARE AVAILABLE IN THE CVT TO IDENTIFY THE CUSTOMER
    SOFTWARE VERSION. SEE DATA AREA'S CVT FIELD CVTVERID
    THIS USERMOD PUTS IN A LEVEL IN THE FOLLOWING FORMAT:
    XCCCYYMS WHERE
    X   TYPE OF CLONE: M MAINTENANCE
                       C CUSTOMER
                       S SYSTEM
                       F FUNCTION
    CCC SYSTEM LONG ID E.G. JED FOR THE INSTALLATION MACHINE
    YY  YEAR
    M   MONTH IN HEX
    S   SEQUENCE NUMBER
    */ .
    NAME IEANUC01 IEAVCVT
    VER  28  4040404040404040 CHECK OFFSETS
    REP  28  D4E2D7C2F9F8F4F1 INSERT PUT LEVEL MSPB9841
    IDRDATA UMOD001
//*
//* DUMP THE MODIFIED MODULE
//*
//S0       EXEC PGM=AMASPZAP
//SYSLIB   DD DSN=SYS1.NUCLEUS,DISP=SHR,VOL=SER=I24CL1,UNIT=SYSDA
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 NAME IEANUC01 IEAVCVT
 DUMPT IEANUC01 IEAVCVT
/*
```

Jan de Decker
System Engineer
JED:SP NV (Belgium)

Retrieving catalog information

INTRODUCTION

MVS storage management technology is evolving at an accelerating pace. Originally, storage management meant simply the management of DASD. As the volume of data increases the data centre has to manage the rapidly increasing storage requirements; without a corresponding increases in staff levels and storage devices. In response to this need, I wrote utilities to manage ICF catalogs, and files on DASDs.

Three utilities are shown below CATMOV, MVTCAT, and SASJCAT. CATMOV is used to change a defined alias from one ICF user catalog to another one, without renaming or deleting all files with a name beginning with this alias name. MVTCAT used an IBM utility to create IDCAMS files for recreating ICF user catalogs. It can be useful in case of disaster recovery. SASJCAT is a job that reads SMF records and logs catalog activity (files created, deleted, renamed, or altered).

CATMOV

```
//CATMOV   JOB   (ACCT£),
//         'PGMRNAME',                                          /*OPER */
//         CLASS=A,                                             /*JCLAS*/
//         MSGLEVEL=(1,1),
//         MSGCLASS=X,
//         NOTIFY=,
//         TIME=3Ø,
//         USER=,GROUP=,PASSWORD=                               /*RACF */
//*                                                             /*JCTRL*/
//*
//*
//* LIB: IPO1.JCLLIB(CATMOV)
//* GDE: CBIPO MVS CUSTOMIZATION
//* DOC: THIS MEMBER CONTAINS SAMPLE JCL TO EXPORT AND IMPORT
//*      A USER CATALOG.
//*
//* NOTE: YOU MUST CHANGE THE CATALOG NAME YOU WISH TO EXPORT
//*       AND IMPORT BEFORE RUNNING THIS JOB, OTHERWISE YOU
//*       WILL BE REMOVING THE CATALOG CATALOG.MVSICF1.VMVSSMP
//*       FROM YOUR SYSTEM.
//*
//CATMOV1 EXEC PGM=IDCAMS,REGION=512K
```

```
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
     EXPORT -
          CATALOG.MVSICF1.VMVSSMP -
          DISCONNECT
     IMPORT -
          OBJECTS((CATALOG.MVSICF1.VMVSSMP -
          VOLUME(MVSSMP) DEVICETYPE(3380))) -
          CONNECT  -
          CATALOG(CATALOG.MVSICFM.VMVSCAT)
/*
```

MVTCAT

```
//MVTCAT JOB EXP,'MVTCAT',CLASS=W,MSGCLASS=O,MSGLEVEL=(1,1),
//         NOTIFY=DUNAND,USER=SYSOP8,PASSWORD=MANXX
//DELDSNL EXEC PGM=IDCAMS,REGION=2048K
//SYSPRINT DD SYSOUT=O
//SYSIN  DD *
   DELETE 'EXPL69.MVTCAT.ICFCAT'
/*
//MVTCAT EXEC SAS,REGION=8M,
//       WORK='200,50',
//         OPTIONS='MEMSIZE=16M DMSBATCH BATCH TERMINAL'
//SOURCLIB DD DSN=SAS.LOCAL.REPORTS,DISP=SHR
//LIBRARY  DD DSN=SAS.MXG.V1313.FORMATS,DISP=SHR
//SASLIST DD DSN=EXPL69.MVTCAT.ICFCAT,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(TRK,(99,39),RLSE),
//         DCB=(RECFM=F,LRECL=133,BLKSIZE=0),MGMTCLAS=DEL32
//SYSIN   DD *
  OPTIONS PAGESIZE=60 LINESIZE=132 ;
  %LET RETCODE=.;
   %INCLUDE SOURCLIB(HIER);
LIBNAME DETAIL 'SAS.LOCAL.CPE.CAT.DETAIL' DISP=SHR;
TITLE "MOVEMENT OF ICF CATALOGS          DATE      &HIER";
FOOTNOTE "REDEVANCE LYON CATALOG              ";
OPTIONS LINESIZE=133  PAGESIZE=68;
OPTIONS NOCENTER;
PROC SORT DATA=DETAIL.XTY6156 OUT=COMPR;
    BY DATETIME HOUR;
DATA COMPR;
SET  COMPR;
KEEP ACTION CACELLS CATNAME DAACLAS DAANAME DAAVOL1 DATETIME ENRNAME
      ENTYPID HOUR ID JOB INXNAME  NEWNAME STRCLAS VOLSER1
              VOLSER2 VOLSER3 VOLSER4 VOLSER5 ;
PROC PRINT DATA=COMPR ;
  ID  DATETIME ;
VAR HOUR ACTION ENRNAME CATNAME JOB  ID STRCLAS DAACLAS VOLSER1 ENTYPID;
RUN;
```

SASJCAT

```
//SASJCAT JOB COM,'SASDIV',CLASS=W,MSGCLASS=O
//*
//DELDEB   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE EXPL69.CPECAT
 IF MAXCC <= 8 THEN SET MAXCC=Ø
/*
//SMFECLA  EXEC PGM=IFASMFDP,REGION=4Ø96K
//DUMPIN   DD  DISP=SHR,DSN=EXPL69.CPESMF
//CAT      DD  DSN=EXPL69.CPECAT,DISP=(,CATLG,DELETE),UNIT=SYSDA,
//         DCB=(RECFM=VBS,LRECL=32760,BLKSIZE=4Ø96),
//         SPACE=(CYL,(5Ø,99),RLSE)
//SYSPRINT DD  SYSOUT=*
//SYSIN DD *
INDD(DUMPIN,OPTIONS(DUMP))
SID(LYO1)
OUTDD(CAT,TYPE(6Ø,61,62,63,64,66,67,68,69))
//*
//SASCAT EXEC SAS,REGION=8M,
//       WORK='15Ø,2Ø',
//         OPTIONS='MEMSIZE=16M DMSBATCH BATCH TERMINAL'
//SMF    DD DSN=EXPL69.CPECAT,DISP=SHR
//REPORT DD DSN=SAS.LOCAL.REPORTS,DISP=SHR
//SASLIST DD SYSOUT=O
//SYSIN  DD *

  OPTIONS PAGESIZE=6Ø LINESIZE=132 ;

  %CPSTART(MODE=BATCH,
           SYSTEM=MVS,
             ROOT=SAS.SAS6Ø9.TS45Ø.CPE.,
             PDB=SAS.LOCAL.CPE.CAT.,
             DISP=OLD,
             ROOTSERV=,
             SHARE=N/A,
             MXGSRC=('SAS.LOCAL.SOURCLIB' 'SAS.MXG.V1313.SOURCLIB'),
),
             MXGLIB=SAS.MXG.V1313.FORMATS
            ) ;

   %INCLUDE SOURCLIB(TYPE6156);
  /* %INCLUDE SOURCLIB(TYPE5Ø);  */
  RUN;

  %LET CMPRRC=.;
  %CMPROCES(,
            COLLECTR=SMF,
            TOOLNM=MXG,
            _RC=CMPRRC
```

```
          );

  %PUT CMPROCES RETURN CODE IS &CMPRRC;

  %LET CPREDRC=.;

  %CPREDUCE(,_RC=CPREDRC);

  %PUT CPREDUCE RETURN CODE IS &CPREDRC;

                  /****  JOURNAL REPORTS  *****/

/*
/*
//DELFIN   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE EXPL69.CPECAT
 IF MAXCC <= 8 THEN SET MAXCC=Ø
//*
```

Claude Dunand (France)

If you want to contribute an article to *MVS Update*, a copy of our *Notes for contributors* can be downloaded from our Web site. Point your browser at www.xephon.com/contnote.html. Alternatively, why not call now for a free copy of our *Notes for contributors*?

MVS Update is looking for contributions, including hints and tips, performance tuning, undocumented features, utilities for making life easier, back-up and restore, archiving, Web enablement, and reviews of the latest versions and releases of software tools.

Articles can be of any length, although shorter contributions are always in demand, and can be sent or e-mailed to Jaime Kaminski at any of the addresses shown on page 2.

IEASYMBQ Assembler program

INTRODUCTION

The objective of the IEASYMBQ program is to show the impact of changing a system symbol or symbols on a parmlib member.

The program produces an ISPF browse report showing, for a member, the effects of changing one or more symbolic variables. Multiple PARMLIBs may be processed. The program uses the standard BLDL to locate the member – therefore, the first parmlib is searched before the second. One or more symbols may be supplied as input. The symbols must be encoded with a value; the symbol may also be prefixed with an ampersand and suffixed with a full stop, but these are optional.

The following are identical as far as this program is concerned with respect to symbol names. PLEX#, &PLEX#, PLEX#., &PLEX#.

The symbol name and a value must be coded, for example, PLEX#='D'. It also possible to use the same syntax as in IEASYM00, for example PLEX#='&SYSPLEX(1:1)'. In this program, the apostrophes are optional, unless, of course, the symbolic contains embedded spaces as part of its value.

Output from the program is an 80-byte format browse on the proposed modified member.

The program is designed to be executed in foreground under the control of a REXX EXEC. I have developed a REXX EXEC and an ISPF panel to enable foreground execution. The program may equally well be executed in background. There are two restrictions:

- The maximum number of parmlibs that can be processed is 20.
- The maximum number of symbols that may be processed is 128.

(Hopefully, these values are far in excess of any requirements).

The following JCL can be used to execute the program:

```
//IEASYMBP EXEC PGM=IEASYMBQ,PARM='IEASYS00'
//STEPLIB  DD DISP=SHR,DSN=your.load
//SYSIN    DD *,DCB=BLKSIZE=80
&LPAR#=3
SYSPLEX='PRDPLEX'
PLEX#='&SYSPLEX#(1:1)'
//SYSPARM  DD DISP=SHR,DSN=SYS1.PARMLIB
//         DD DISP=SHR,DSN=SYS1.OTHER.PARMLIB
//         DD DISP=SHR,DSN=SYS1.BACK.PARMLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
```

The steplib DDname is there for testing purposes, and will not be required in production.

In the above example SYS1.PARMLIB, SYS1.OTHER.PARMLIB, and SYS1.BACK.PARMLIB will be searched for a member named IEASYS00. The DDname for the parmlib must be SYSPARM. The symbols &LPAR, &SYSPLEX, and &PLEX# will be modified as indicated. The symbols may be entered with or without the leading ampersand, one or more on each line.

FOREGROUND EXECUTION

You may also execute this program in foreground. To use this EXEC, you must be in ISPF. If you are not, the EXEC will inform you of that state, and not function any further. Enter the following to execute the EXEC:

```
TSO EX 'your.exec(IEASYMBQ)' EXEC
```

on the command line, or from option 6 (without the leading TSO). This will cause a fairly self-explanatory panel to be displayed. The panel limits the number of parameter libraries and search arguments, but, otherwise, is identical to running the program in batch. At program completion, a BROWSE will be initiated on the output report. Another search can be launched immediately, if you wish.

AN EXAMPLE

```
──────────── Parmlib Symbolic Change Impact ──-
 COMMAND ===>

 Enter symbols ===> &sysclone='&sysplex(1:2)' sysplex=tdrplex
  Values must be provided.  For example: SYSPLEX=PRDPLEX
 More symbols  ===>
```

```
 MEMBER to be processed    ===> ieasys00

 PARMLIB to be searched    ===> 'SYS1.PARMLIB'

 PARMLIB to be searched    ===>

 PARMLIB to be searched    ===>

 PARMLIB to be searched    ===>
```

This will produce output similar to the following (only part is shown):

```
 BROWSE    SYS98334.T074854.RA000.MYID.R0129343
 Command ===>
******************************** Top of Data ******************
ALLOC=00,                      VIO AND DYNAMIC ALLOCATION DEFAULTS
CLOCK=W7,                      SET TOD CLOCK
CLPA,                          ALWAYS RELOAD LPA AND VIO
CMB=(UNITR,1024),              COLLECT MEASUREMENT DATA FOR THESE
CMD=TD,                        COMMANDS AT MSTR SCHEDULER INIT
CON=10,                        CONSOLE SPECIFICATION
COUPLE=D0,                     SYSPLEX ENVIRONMENT
CSCBLOC=ABOVE,                 CSCB CONTROL BLOCK CHAIN LOCATION
CVIO,                          ALWAYS RELOAD VIO
DEVSUP=00,                     IDRC CONTROL
DIAG=00,                       CSA, SQA TRACKING AND GFS TRACE
DUMP=NO,                       DUMP DATASETS ON DASD (DEFAULT)
FIX=(00,NOPROT),               FIX MODULES IN LPA
GRS=TRYJOIN,                   INITIALIZE GRS
GRSCNF=00,                     DEFAULT GRS CONFIG
GRSRNL=EXCLUDE,                ENABLE MII TO PERFORM GRS FUNCTIONS
ICS=TD,                        SELECT IEAICSXX INSTALL CNTL SPECS FOR
IOS=00,                        SELECT HOT I/O OPTIONS
IPS=TD,                        SELECT IEAIPSXX INSTALL PERF SPECS FOR
```

In the examples, the whole line of the display is not shown. The program follows:

```
TITLE 'PARMLIB MEMBER PROCESS - SHOW EFFECT OF SYMBOL CHANGES'
*        SHOW THE CONTENT OF A PARMLIB MEMBER SHOWING SYMBOL CHANGES
*
*        THIS PROGRAM WILL PROCESS ONE OR MORE PARMLIBS LOOKING FOR A
*        SPECIFIC MEMBER.
*        THE MEMBER NAME IS INPUT VIA JOB-STEP PARAMETER.
*        THE MEMBER WILL BE SHOWN WITH USER-SUPPLIED AND DEFAULT
*        SYMBOL SUBSTITUTION UNLESS THE USER-SUPPLIED SYMBOL CONTAINS
*        A SYMBOL IN WHICH CASE THE OUTPUT WILL SHOW THE FURTHER
*        SUBSTITUTION.
         SPACE 1
*        A MAXIMUM OF 128 SYMBOLS WITH VALUES MAY BE PROVIDED.
*        THE USER-SUPPLIED SYMBOLS MAY BE ENTERED WITH OR
*        WITHOUT A LEADING AMPERSAND, AND WITH OR WITHOUT A TRAILING
```

```
*        PERIOD.  THE PROGRAM WILL ADD THE MISSING & AND PERIOD.
         SPACE 2
*        INPUT FILES:
         SPACE 1
*              CARD INPUT
*                 DDNAME=SYSIN
*                 LRECL=8Ø
*                 BLKSIZE=?
*                 RECFM=F/BF
         SPACE 2
*              PARTITIONED DATA SET(S) (MAY BE CONCATENATED)
*                 DDNAME=SYSPARM
*                 LRECL=8Ø
*                 BLKSIZE=?
*                 RECFM=F/BF
         SPACE 2
*              JOB STEP PARAMETER (MEMBER NAME TO PROCESS)
*                 PARM='MEMBER'
         SPACE 2
*        OUTPUT FILES:
         SPACE 1
*              REPORT
*                 DDNAME=SYSPRINT
*                 LRECL=8Ø
*                 BLKSIZE=?
*                 RECFM=AF/ABF/ABFS
         SPACE 2
*        SAMPLE JCL
         SPACE 1
*              //IEASYMBQ  EXEC PGM=IEASYMBQ,PARM='IEASYSØØ'
*              //STEPLIB  DD DISP=SHR,DSN=MY.LOAD
*              //SYSIN    DD *,DCB=BLKSIZE=8Ø
*              SYSPLEX='PRDPLEX'
*              //SYSPARM  DD DISP=SHR,DSN=SYS1.PARMLIB
*              //         DD DISP=SHR,DSN=SYS1.OLD.PARMLIB
*              //SYSPRINT DD SYSOUT=*
         EJECT
*        PROGRAM ATTRIBUTES
         SPACE 1
*                REENTRANT
*                AMODE: 31
*                RMODE: 24 (DCBS)
         SPACE 3
*        EXTERNAL ROUTINES USED:
*                ASASYMBM               . SYMBOLIC SUBSTITUTION
         TITLE 'MACROS AND CONTROL BLOCKS USED'
*        MACROS USED
         SPACE 1
*                ABEND                  . ABEND USER TASK
*                BLDL                   . BUILD DIRECTORY ENTRY LIST
*                CHECK                  . WAIT FOR I/O COMPLETION
*                CLOSE         MF=E     . TERMINATE FILE PROCESSING
```

```
*                FREEPOOL                . RELEASE BUFFER POOL
*                LINKX    SF=L/MF=E      . LINK TO PROGRAM
*                OPEN          MF=L/E    . PREPARE FILE FOR PROCESSING
*                PUT                     . WRITE A RECORD (QSAM)
*                READ          MF=L/E    . READ A BLOCK (BSAM / BPAM)
*                WTO           MF=L/E    . OUTPUT MESSAGE
          SPACE 3
*         CONTROL BLOCKS USED
          SPACE 1
*                ASASYMBP                . MAP ASASYMBM PARAMETER LIST
*                DCBD (IHADCB)           . DCB MAPPING  (DSECT)
*                IEZIOB                  . IOB MAPPING  (DSECT)
*                IHADECB                 . DECB MAPPING (DSECT)
          TITLE 'REGISTER EQUATES AND GENERAL USAGE'
RØ        EQU   Ø                       . ARGUMENT ADDRESS FOR MACROS
R1        EQU   1                       . ARGUMENT ADDRESS FOR MACROS
R2        EQU   2
R3        EQU   3
R4        EQU   4
R5        EQU   5
R6        EQU   6
R7        EQU   7
R8        EQU   8
R9        EQU   9
R1Ø       EQU   1Ø                      . BRANCH & SAVE RETURN ADDRESS
R11       EQU   11                      . CSECT BASE REGISTER
R12       EQU   12                      . CSECT BASE REGISTER
R13       EQU   13                      . -> SAVE AREA
R14       EQU   14                      . RETURN ADDRESS
*                                       . AMODE SWITCH
R15       EQU   15                      . ENTRY POINT ADDRESS
*                                       . RETURN CODE
          TITLE 'CSECT CONTROL'
          PRINT NOGEN
IEASYMBQ  CSECT
IEASYMBQ  AMODE 31
IEASYMBQ  RMODE 24
          LA    R14,Ø(,R14)             . VALIDITY OF R14
          BSM   R14,RØ                  . CURRENT ADDRESSING MODE
          BAKR  R14,RØ                  . ESTABLISH LINKAGE
          LR    R11,R15                 . 11 -> EPA
          LA    R12,4Ø95(,R11)
          LA    R12,1(,R12)             . 12 - SECOND BASE REGISTER
          USING IEASYMBQ,R11,R12        . CSECT ADDRESSABILITY
          STORAGE OBTAIN,               . GET DYNAMIC AREA            *
                ADDR=(R13),                                           *
                LENGTH=DYNLEN,                                        *
                LOC=(BELOW,ANY),                                      *
                SP=Ø
          MVC   4(4,R13),=C'F1SA'       . FORMAT OF SAVE AREA
          BAS   R14,AMODE31             . GO AMODE 31
          USING DYNAREA,R13             . DSECT ADDRESSABILITY
          XC    RETCODE,RETCODE         . INITIALIZE RETURN CODE
```

```
         BAS   R1Ø,INITIAL              . PERFORM INITIALIZATION
         BAS   R1Ø,PROCESS              . PERFORM MAIN PROCESSING
         BAS   R1Ø,TERMIN               . PERFORM TERMINATION
         L     R4,RETCODE               . 4 - RETURN CODE
         STORAGE RELEASE,               . RELEASE DYNAMIC AREA         *
               ADDR=(R13),                                             *
               LENGTH=DYNLEN,                                          *
               SP=Ø
         LR    R15,R4                   . 15 - RETURN CODE
         PR                             . RETURN TO CALLER
         TITLE 'INITIALIZATION PROCESSING LOGIC'
*        REGISTER USAGE
         SPACE 1
*        1                              . WORK
*        2                              . -> BLDL LIST
*        3                              . -> ASASYMBM PARAMETER LIST
*                                       . ->         SUBSTITUTION ARRAY
*        4                              . WORK
*        5                              . -> INPUT DCB SYSIN
*        6                              . -> INPUT DCB SYSPARM
*        7                              . -> OUTPUT DCB SYSPRINT
         SPACE 2
*        PROCESS JOB-STEP PARAMETER
*        MOVE THE FORM DCBS INTO THE DYNAMIC AREA
*        OPEN THE INPUT AND OUTPUT FILES
*        GET THE BUFFER FOR THE INPUT PDS
*        OBTAIN AND FORMAT ASASYMBM PARAMETER LIST STORAGE
*        OBTAIN AND INITIALIZE ASASYMBM SUBSTITUTION ARRAY STORAGE
*        PROCESS SYSIN - SYMBOLICS TO BE PROCESSED
         SPACE 2
INITIAL  EQU   *
         ST    R1Ø,BSAVE                . SAVE RETURN ADDRESS
         XC    BLDL,BLDL                . INITIALIZE BLDL AREA
         LA    R2,BLDL                  . 2 -> BLDL AREA
         USING BLDLIST,R2               . DSECT ADDRESSABILITY
         MVC   BLDL#ENT,=H'1'           . INITIALIZE # ENTRIES
         MVC   BLDLENTL,=Y(LENBLDL-2)   . LENGTH OF ENTRY
         MVI   DIRMEMNM,C' '            . INITIALIZE MEMBER NAME
         MVC   DIRMEMNM+1(L'DIRMEMNM-1),DIRMEMNM
         EREG  R1,R1                    . GET CALLER'S REGISTER 1
         L     R1,Ø(,R1)                . 1 -> PARAMETER
         LH    R4,Ø(,R1)                . 4 - LENGTH OF PARAMETER
         LTR   R4,R4                    . Q. PARAMETER LENGTH ZERO?
         BZ    BØØØ1                    . A. YES - ERROR
         CH    R4,=H'8'                 . Q. PARAMETER LENGTH > EIGHT?
         BH    BØØØ1                    . A. YES - ERROR
         BCTR  R4,RØ                    . DECREMENT FOR EXECUTE
         EX    R4,MVCPARM               . SAVE MEMBER NAME
         MVC   DYNDCB1,SYSIN            . SYSIN DCB
         MVC   DYNDCB2,SYSPARM          . SYSPARM DCB
         MVC   DYNDCB3,SYSPRINT         . SYSPRINT DCB
         LA    R5,DYNDCB1               . 5 -> DCB FOR SYSIN
```

```
         LA    R6,DYNDCB2               . 6 -> DCB FOR SYSPARM
         LA    R7,DYNDCB3               . 7 -> DCB FOR SYSPRINT
         MVC   DMALIST(LENOPEN),OPENL   . MOVE DMA LIST TO DYNAMIC
         BAS   R14,AMODE24              . SWITCH TO AMODE 24
         OPEN  ((R5),,(R6),,(R7),OUTPUT),                              *
               MF=(E,DMALIST)
         BAS   R14,AMODE31              . SWITCH TO AMODE 31
         USING IHADCB,R5                . CHECK THE OPEN OF SYSIN
         TM    DCBOFLGS,DCBOFOPN        . Q. OPEN OKAY?
         BNO   B@@@2                    . A. NO
         DROP  R5                       . DSECT NO LONGER REQ'D
         USING IHADCB,R6                . CHECK THE OPEN OF SYSPARM
         TM    DCBOFLGS,DCBOFOPN        . Q. OPEN OKAY?
         BNO   B@@@3                    . A. NO
         MVC   #BLK,DCBBLKSI            . SAVE THE BLOCK SIZE
         DROP  R6                       . DSECT NO LONGER REQ'D
         USING IHADCB,R7                . CHECK THE OPEN OF SYSPRINT
         TM    DCBOFLGS,DCBOFOPN        . Q. OPEN OKAY?
         BNO   B@@@4                    . A. NO
         DROP  R7                       . DSECT NO LONGER REQ'D
         LH    R2,#BLK                  . 2 - BLOCKSIZE OF PDS
         STORAGE OBTAIN,                . GET BUFFER                   *
               ADDR=(R1),                                              *
               LENGTH=(R2),                                            *
               LOC=(BELOW,ANY),                                        *
               SP=@
         ST    R1,@BLK                  . SAVE ADDRESS OF STORAGE
         STORAGE OBTAIN,                . OBTAIN ASASYMBM PARAMETER    *
               ADDR=(R3),               .   LIST                       *
               LENGTH=SYMBP_LEN,                                       *
               SP=@
         ST    R3,@SYMBP                . SAVE -> PARAMETER LIST
         USING SYMBP,R3                 . DSECT ADDRESSABILITY
         XC    SYMBP(SYMBP_LEN),SYMBP   . CLEAR PARAMETER LIST
         L     R4,=F'71'                . 4 - SEVENTY ONE
         ST    R4,SYMBPPATTERNLENGTH    . SAVE LENGTH
         LA    R4,SYMLENTR              . 4 -> TARGET LENGTH
         ST    R4,SYMBPTARGETLENGTH@    . SAVE ADDRESS
         LA    R4,SYMRETCD              . 4 -> RETURN CODE
         ST    R4,SYMBPRETURNCODE@      . SAVE ADDRESS
         DROP  R3                       . DSECT NOT REQUIRED
         LA    R1,SYMBTE_LEN            . 1 - LENGTH OF SYMBOL ENTRY
         SLA   R1,7                     . MULTIPLY BY 128
         LA    R1,SYMBT_LEN(,R1)        . ADD IN LENGTH OF HEADER
         ST    R1,#SYMBT                . SAVE LENGTH
         STORAGE OBTAIN,                . OBTAIN ASASYMBM SUBSTITUTION *
               ADDR=(R3),               .   ARRAY                      *
               LENGTH=(R1),                                            *
               SP=@
         ST    R3,@SYMBT                . SAVE -> SUBSTITUTION ARRAY
         USING SYMBT,R3                 . DSECT ADDRESSABILITY
         XC    SYMBTHEADER,SYMBTHEADER  . INITIALIZE HEADER
         DROP  R3
```

```
         BAS   R1Ø,PROCINPT             . PROCESS SYSIN DATA
         L     R1Ø,BSAVE                . RESTORE RETURN ADDRESS
         BR    R1Ø                      . EXIT
MVCPARM  MVC   DIRMEMNM(*-*),2(R1)      . EXECUTED MOVE
         DROP  R2                       . DSECT NOT REQUIRED
         SPACE 2
BØØØ1    EQU   *                        . JOB-STEP PARAMETER ERROR
         WTO   MF=(E,WTO1L)
         B     BØØØ5
         SPACE 2
BØØØ2    EQU   *                        . SYSIN OPEN FAILURE
         WTO   MF=(E,WTO2L)
         B     BØØØ5
         SPACE 2
BØØØ3    EQU   *                        . SYSPARM OPEN FAILURE
         WTO   MF=(E,WTO3L)
         B     BØØØ5
         SPACE 2
BØØØ4    EQU   *                        . SYSPRINT OPEN FAILURE
         WTO   MF=(E,WTO4L)
         SPACE 2
BØØØ5    EQU   *
         ABEND 1ØØ,                     . ABEND THE TASK              *
               DUMP
         TITLE 'MAIN PROCESSING LOGIC'
*        REGISTER USAGE
         SPACE 1
*        3                              . -> SYSPARM DCB
*        4                              . -> BLDL LIST
*        6                              . -> DECB
*        8                              . -> PDS BLOCK
         SPACE 2
*        ISSUE BLDL
*        POSITION ON THE FIRST BLOCK FOR THIS ENTRY
*        FOR EVERY BLOCK FOR THE MEMBER
*             PROCESS CURRENT BLOCK
*             READ NEXT BLOCK
PROCESS  EQU   *
         ST    R1Ø,CSAVE                . SAVE R1Ø
         LA    R3,DYNDCB2               . 3 -> DCB FOR PDS
         LA    R4,BLDL                  . 4 -> BLDL LIST
         USING BLDLIST,R4               . DSECT ADDRESSABILITY
         BLDL  (R3),                    . ISSUE BLDL                  *
               (R4)
         MVC   TTR(3),DIRTTR            . STORE -> MEMBER'S FIRST BLOCK
         MVI   TTR+3,X'ØØ'              . ZAP LAST BYTE OF TTR
         FIND  (R3),                    . LOCATE FIRST BLOCK OF MEMBER *
               TTR,                                                    *
               C                        . NO DIRECTORY SEARCH
C0001    EQU   *                        . RETRIEVE BLOCK FROM MEMBER
*
         MVC   DECBD(L'DECBD),READL     . MOVE IN LIST FORM
```

```
         LA    R6,DECBD                  . 6 -> DECB
         L     R8,@BLK                   . 8 -> BLOCK OF STORAGE
         READ  (R6),                     . READ BLOCK FROM PDS MEMBER   *
               SF,                       . SEQUENTIAL FORWARDS          *
               (R3),                                                    *
               (R8),                                                    *
               'S',                                                     *
               MF=E
         CHECK DECBD                     . AWAIT COMPLETION
         BAS   R1Ø,MEMBPROC              . PROCESS CURRENT BLOCK
         B     CØØØ1                     . GET NEXT BLOCK FROM MEMBER
         DROP  R4                        . DSECT NO LONGER REQUIRED
CØØØ2    EQU   *
         L     R1Ø,CSAVE                 . RESTORE R1Ø
         BR    R1Ø                       . EXIT
         TITLE 'END OF PROCESSING - CLOSE FILES, FREE BUFFER POOL'
*        REGISTER USAGE
         SPACE 1
*        1                               . WORK
*        4                               . -> ASASYMBM PARAMETER LIST
*                                        . ->        SUBSTITUTION ARRAY
*                                        . -> SYMBOL ARRAY
*        5                               . -> SYSIN DCB
*        6                               . -> SYSPARM DCB
*        7                               . -> SYSPRINT DCB
         SPACE 2
*        RELEASE ASASYMBM STORAGE AREAS
*        RELEASE SYMBOL ARRAY STORAGE
*        CLOSE THE FILES THAT ARE OPEN
*        FREE THE BUFFER POOLS ACQUIRED DYNAMICALLY AT OPEN
         SPACE 2
TERMIN   EQU   *
*
         L     R1,#SYMBT                 . 1 - LENGTH OF ARRAY
         L     R4,@SYMBT                 . 4 -> SUBSTITUTION ARRAY
         STORAGE RELEASE,                . RELEASE ASASYMBM SUBSTITUTION*
               ADDR=(R4),                .   ARRAY                      *
               LENGTH=(R1)
         L     R4,@SYMBP                 . 4 -> ASASYMBM PARAMETER LIST
         STORAGE RELEASE,                . RELEASE STORAGE              *
               ADDR=(R4),                                               *
               LENGTH=SYMBP_LEN
         LA    R1,SYMENT                 . 1 - ENTRY LENGTH SYMBOL ARRAY
         SLA   R1,7                      . 128 ENTRIES
         L     R4,@SYMBOLS               . 4 -> SYMBOL ARRAY
         STORAGE RELEASE,                . RELEASE STORAGE SYMBOL ARRAY *
               ADDR=(R4),                                               *
               LENGTH=(R1)
         MVC   DMALIST(LENOPEN),OPENL    . TWO FILES TO CLOSE
         LA    R5,DYNDCB1                . 5 -> SYSIN DCB
         LA    R6,DYNDCB2                . 6 -> SYSPARM DCB
         LA    R7,DYNDCB3                . 7 -> SYSPRINT DCB
         BAS   R14,AMODE24               . SWITCH TO AMODE 24
```

```
         CLOSE ((R5),,(R6),,(R7)),      . CLOSE FILES                  *
               MF=(E,DMALIST)
         FREEPOOL (R5)                  . RELEASE BUFFER POOL
         FREEPOOL (R6)                  . RELEASE BUFFER POOL
         FREEPOOL (R7)                  . RELEASE BUFFER POOL
         BAS   R14,AMODE31              . BACK TO AMODE 31
         BR    R1Ø                      . EXIT
         TITLE 'PROCESS INPUT SYSIN - SYMBOLS TO BE PROCESSED'
*        REGISTER USAGE
         SPACE 1
*        1                              . WORK
*        3                              . LENGTH OF CURRENT SYMBOL
*        4                              . -> CURRENT SAVE SYMBOL BYTE
*        5                              . -> CURRENT BYTE IN RECORD
*        6                              . BYTES IN INPUT RECORD
*        7                              . -> INPUT DCB
*        8                              . -> CURRENT ENTRY IN ARRAY
*        9                              . NUMBER OF SYMBOLS
         SPACE 2
*        ACQUIRE STORAGE FOR SYMBOL ARRAY
*        READ THE INPUT SYSIN RECORD
*        FOR EVERY SPACE DELIMITED VALUE
*            VALIDATE - MAX 8 CHARACTERS - EXCLUDING OPTIONAL
*            LEADING AMPERSAND AND TRAILING PERIOD
*            IF VALID, STORE IN ARRAY
*            READ NEXT INPUT SYSIN RECORD
         SPACE 2
PROCINPT EQU   *
*
         ST    R1Ø,ESAVE                . SAVE RETURN ADDRESS
         SLR   R9,R9                    . 9 - ZERO (COUNT OF SYMBOLS)
         LA    R7,DYNDCB1               . 7 -> SYSIN DCB
         LA    R1,SYMENT                . 1 - ENTRY LENGTH SYMBOL ARRAY
         SLA   R1,7                     . 128 ENTRIES
         STORAGE OBTAIN,                . GET STORAGE FOR SYMBOL ARRAY *
               ADDR=(R8),                                              *
               LENGTH=(R1),                                            *
               SP=Ø
         ST    R8,@SYMBOLS              . SAVE ADDRESS OF SYMBOL ARRAY
         USING SYMARRAY,R8              . DSECT ADDRESSABILITY
E0001    EQU   *
         BAS   R14,AMODE24              . SWITCH TO AMODE 24
         GET   (R7),                    . GET SYSIN RECORD             *
               WORKREC
         BAS   R14,AMODE31              . BACK TO AMODE 31
         LA    R5,WORKREC               . 5 -> INPUT RECORD
         LA    R6,72                    . 6 - BYTES TO PROCESS
E0002    EQU   *
         CLI   Ø(R5),C' '               . Q. SPACE FOUND?
         BNE   E0003                    . A. NO
         LA    R5,1(,R5)                . 5 -> NEXT BYTE
         BCT   R6,E0002                 . LOOP
```

```
         B     E0001                    . GET NEXT RECORD
E0003    EQU   *
         SLR   R3,R3                    . 3 - ZERO (LENGTH OF DATA)
         MVI   SAVESYM,C' '             . SYMBOL WORK AREA
         MVC   SAVESYM+1(L'SAVESYM-1),SAVESYM
         LA    R4,SAVESYM               . 4 -> SYMBOL WORK AREA
         CLI   0(R5),C'&&'              . Q. AMPERSAND FOUND?
         BE    E0004                    . A. YES
         CLI   0(R5),C'.'               . Q. INPUT BYTE A PERIOD?
         BE    E0002                    . A. YES - IGNORE THE PERIOD
         MVI   0(R4),C'&&'              . MOVE OUT AMPERSAND
         LA    R3,1(,R3)                . INCREMENT BYTES IN SYMBOL
         LA    R4,1(,R4)                . 4 -> NEXT OUTPUT BYTE
         B     E0005
E0004    EQU   *
         MVI   0(R4),C'&&'              . MOVE THE AMPERSAND
         LA    R5,1(,R5)                . 5 -> NEXT INPUT BYTE
         BCTR  R6,R0                    . DECREMENT BYTES TO PROCESS
         CLI   0(R5),C' '               . Q. NEXT INPUT BYTE A SPACE?
         BE    E0002                    . A. YES - IGNORE THE AMPERSAND
         CLI   0(R5),C'.'               . Q. NEXT INPUT BYTE A PERIOD?
         BE    E0002                    . A. YES - IGNORE THE AMPERSAND
         LA    R3,1(,R3)                . INCREMENT BYTES IN SYMBOL
         LA    R4,1(,R4)                . 4 -> NEXT OUTPUT BYTE
E0005    EQU   *
         MVC   0(1,R4),0(R5)            . MOVE THE BYTE
         LA    R3,1(,R3)                . INCREMENT BYTES IN SYMBOL
         LA    R5,1(,R5)                . 5 -> NEXT INPUT BYTE
         LA    R4,1(,R4)                . 8 -> NEXT OUTPUT BYTE
         CLI   0(R5),C' '               . Q. NEXT INPUT BYTE A SPACE?
         BE    E0014                    . A. YES - ERROR
         CLI   0(R5),C'='               . Q. NEXT INPUT BYTE AN EQUAL?
         BE    E0006                    . A. YES - END OF SYMBOL
         CLI   0(R5),C'.'               . Q. NEXT INPUT BYTE A PERIOD?
         BE    E0006                    . A. YES - END OF SYMBOL
         CH    R3,=H'9'                 . Q. MAX LENGTH?
         BH    E0011                    . A. YES
         BCT   R6,E0005                 . LOOP
E0006    EQU   *
         MVI   SUBFLAG,C'N'             . FLAG NO SUBSTITUTION
         MVI   0(R4),C'.'               . MOVE OUT A PERIOD
         CLI   0(R5),C'.'               . Q. PERIOD INPUT?
         BNE   E0007                    . A. NO
         LA    R5,1(,R5)                . NEXT INPUT BYTE
         BCTR  R6,R0                    . DECREMENT BYTES TO PROCESS
E0007    EQU   *
         CLI   0(R5),C'='               . Q. EQUAL SIGN FOUND?
         BNE   E0014                    . A. NO - ERROR
         LA    R5,1(,R5)                . STEP OVER EQUAL
         BCTR  R6,R0                    . DECREMENT BYTES TO PROCESS
         MVI   SUBFLAG,C'Y'             . FLAG SUBSTITUTION
E0008    EQU   *
```

```
         LA    R9,1(,R9)               . INCREMENT NUMBER OF SYMBOLS
         CH    R9,=H'128'              . Q. MORE THAN 128?
         BH    E0012                   . A. YES
         ST    R3,SYMLEN               . SAVE THE LENGTH OF THE SYMBOL
         MVC   SYMBOL,SAVESYM          . SAVE THE SYMBOL
         CLI   SUBFLAG,C'Y'            . Q. SUBSTITUTION VALUE?
         BNE   E0009                   . A. NO
         BAS   R10,SUBVAL              . PROCESS SUBSTITUTION VALUE
         CLC   RETCODE,=F'0'           . Q. RETURN CODE ZERO?
         BNE   E0015                   . A. NO
E0009    EQU   *
         LA    R8,SYMENT(,R8)          . 8 -> NEXT SYMBOL ENTRY
         LA    R5,1(,R5)               . 5 -> NEXT INPUT BYTE
         BCT   R6,E0002                . LOOP
         B     E0001                   . READ NEXT RECORD
         DROP  R8                      . DSECT NOT REQUIRED
E0010    EQU   *
         LTR   R9,R9                   . Q. ANY VALID SYMBOLS?
         BZ    E0013                   . A. NO
         ST    R9,#SYMBOLS             . SAVE NUMBER OF SYMBOLS
         B     E0015
E0011    EQU   *                       . SYMBOL EXCESSIVE LENGTH
         WTO   MF=(E,WTO5L)
         MVC   RETCODE,=F'8'           . SET RETURN CODE
         B     E0015
E0012    EQU   *                       . MORE THAN 100 SYMBOLS
         WTO   MF=(E,WTO6L)
         MVC   RETCODE,=F'8'           . SET RETURN CODE
         B     E0015
E0013    EQU   *                       . NO VALID SYMBOLS
         WTO   MF=(E,WTO7L)
         MVC   RETCODE,=F'8'           . SET RETURN CODE
         B     E0015
E0014    EQU   *                       . NO VALUE SUPPLIED WITH SYMBOL
         WTO   MF=(E,WTO8L)
         MVC   RETCODE,=F'8'           . SET RETURN CODE
*
E0015    EQU   *
*
         L     R10,ESAVE               . RETURN ADDRESS
         BR    R10                     . RETURN
         TITLE 'PDS MEMBER RECORD PROCESS'
*             REGISTER USAGE
         SPACE 1
*        2                             . -> IOB
*        6                             . -> DECB
*        7                             . -> POSITION IN BLOCK
*        8                             . INDEX INCREMENT - 80
*        9                             . -> END OF BLOCK
         SPACE 2
*        DETERMINE ACTUAL LENGTH OF THE BLOCK JUST READ OF THIS
*             MEMBER.  THIS IS TO SET UP A BXLE INSTRUCTION.
```

```
*        PROCESS THE RECORDS WITHIN THE BLOCK.
*        FOR EACH RECORD IN THE BLOCK INCREMENT RECORDS IN
*              MEMBER ACCUMULATOR.
*        AT THE END OF THE BLOCK THIS ROUTINE IS EXITED.
         SPACE 2
MEMBPROC EQU   *
         STM   R6,R1Ø,FSAVE             . SAVE REGISTERS
         L     R7,@BLK                  . 7, 9 -> BLOCK
         LR    R9,R7
         LA    R6,DYNDCB2               . 6 -> SYSPARM DCB
         USING IHADCB,R6                . MAP THE DCB
         AH    R9,DCBBLKSI              . 9 -> END OF BLOCK
         LH    R8,DCBLRECL              . 8 - RECORD LENGTH
         DROP  R6                       . DSECT NOT WANTED ANY MORE
         LA    R6,DECBD                 . 6 -> DECB
         USING DECB,R6                  . MAP THE DECB
         L     R2,DECIOBPT              . 2 -> INPUT OUTPUT BLOCK
         SH    R2,=H'16'                . 2 -> IOB PREFIX
         DROP  R6                       . DSECT NOT WANTED ANY MORE
         USING IOB,R2                   . MAP THE IOB
         SH    R9,IOBCSW+5              . 9 -> ACTUAL END OF BLOCK
*                                       . FROM CHANNEL STATUS WORD
         DROP  R2                       . DSECT NOT WANTED ANY MORE
         BCTR  R9,RØ                    . 9 -> ACTUAL END - ONE
FØØØ1    EQU   *
         ST    R7,@RECORD               . SAVE START OF RECORD
         BAS   R1Ø,SYMSERCH             . SEARCH FOR SYMBOL
         BXLE  R7,R8,FØØØ1              . PROCESS NEXT RECORD IN BLOCK
*                                       . END OF BLOCK
         LM    R6,R1Ø,FSAVE             . RESTORE REGISTERS
         BR    R1Ø                      . RETURN TO CALLER
         TITLE 'PROCESS INPUT SUBSTITUTION VALUE'
*              REGISTER USAGE
         SPACE 1
*        1                              . WORK
*        2                              . WORK
*        3                              . LENGTH OF CURRENT VALUE
*        4                              . -> CURRENT SAVE VALUE BYTE
*        5                              . -> CURRENT BYTE IN RECORD
*        6                              . BYTES IN INPUT RECORD
*        7                              . DO NOT MODIFY
*        8                              . -> CURRENT ENTRY IN ARRAY
*        9                              . DO NOT MODIFY
         SPACE 2
*        THIS CODE PROCESS AN INPUT SUBSTITUTION VALUE. THIS
*        VALUE SHOULD BE ENCLOSED IN APOSTROPHES.
*        THE ASASYMBM SUBSTITUTION TABLE IS ALSO POPULATED
         SPACE 2
SUBVAL   EQU   *
         USING SYMARRAY,R8              . DSECT ADDRESSABILITY
         L     R2,@SYMBT                . 2 -> SYMBOL ARRAY HEADER
         USING SYMBT,R2                 . DSECT MAPPING HEADER
```

```
         LH    R1,SYMBTNUMBEROFSYMBOLS  . NUMBER OF SYMBOLS IN ARRAY
         LA    R2,SYMBTTABLEENTRIES     . 2 -> SYMBOL ARRAY ENTRY
         DROP  R2                       . HEADER NOT NEEDED
         USING SYMBTE,R2                . DSECT MAPPING ENTRY
         MH    R1,=Y(SYMBTE_LEN)        . DISPLACEMENT FOR ENTRY
         LA    R2,Ø(R1,R2)              . 2 -> CURRENT ENTRY
         LA    R1,SYMBOL                . 1 -> CURRENT SYMBOL
         ST    R1,SYMBTESYMBOLPTR       . SAVE -> IN ENTRY
         LA    R3,1(,R3)                . INCLUDE PERIOD IN LENGTH
         ST    R3,SYMBTESYMBOLLENGTH    . SAVE LENGTH IN ENTRY
         SLR   R3,R3                    . 3 - ZERO (LENGTH OF VALUE)
         MVI   SAVEVAL,C' '             . INITIALIZE WORK AREA
         MVC   SAVEVAL+1(L'SAVEVAL-1),SAVEVAL
         LA    R4,SAVEVAL               . 4 -> VALUE WORK AREA
         MVI   APOSFLAG,C'N'            . FLAG NO APOSTROPHE FOUND
         MVI   AMPFLAG,C'N'             . FLAG NO AMPERSAND FOUND
         CLI   Ø(R5),C''''              . Q. LEADING APOSTROPHE?
         BNE   GØØØ1                    . A. NO
         MVI   APOSFLAG,C'Y'            . FLAG APOSTROPHE FOUND
         LA    R5,1(,R5)                . SKIP OVER APOSTROPHE
         BCTR  R6,RØ                    . DECREMENT BYTES TO PROCESS
GØØØ1    EQU   *
         CLI   Ø(R5),C''''              . Q. APOSTROPHE FOUND?
         BNE   GØØØ2                    . A. NO
         CLI   APOSFLAG,C'Y'            . Q. LEADING APOSTROPHE?
         BE    GØØØ6                    . A. YES - END OF VALUE
         B     GØØØ8                    . A. NO - ERROR
GØØØ2    EQU   *
         CLI   Ø(R5),C' '               . Q. SPACE FOUND?
         BNE   GØØØ3                    . A. NO
         CLI   APOSFLAG,C'Y'            . Q. WITHIN APOSTROPHE?
         BNE   GØØØ6                    . A. NO - END OF VALUE
GØØØ3    EQU   *
         CLI   Ø(R5),C'&&'              . Q. AMPERSAND FOUND?
         BNE   GØØØ4                    . A. NO
         MVI   AMPFLAG,C'Y'             . FLAG AMPERSAND
GØØØ4    EQU   *
         CLI   Ø(R5),C'.'               . Q. PERIOD FOUND?
         BNE   GØØØ5                    . A. NO
         MVI   AMPFLAG,C'N'             . SWITCH OFF AMPERSAND FLAG
GØØØ5    EQU   *
         MVC   Ø(1,R4),Ø(R5)            . SAVE VALUE BYTE
         LA    R3,1(,R3)                . INCREMENT BYTES
         CH    R3,=H'2Ø'                . Q. MAX LENGTH?
         BH    GØØØ9                    . A. TOO GREAT
         LA    R4,1(,R4)                . 4 -> NEXT OUTPUT BYTE
         LA    R5,1(,R5)                . 5 -> NEXT INPUT BYTE
         BCT   R6,GØØØ1                 . LOOP
GØØØ6    EQU   *
         CLI   AMPFLAG,C'Y'             . Q. AMPERSAND FOUND?
         BNE   GØØØ7                    . A. NO
         MVI   Ø(R4),C'.'               . OUTPUT A PERIOD
```

```
         LA    R3,1(,R3)                . INCREMENT BYTES OUTPUT
G0007    EQU   *
         LTR   R3,R3                    . Q. ANY VALUE FOUND?
         BZ    G0010                    . A. NO
         MVC   SUBVALUE,SAVEVAL         . SAVE THE VALUE
         ST    R3,SYMBTESUBTEXTLENGTH   . STORE THE LENGTH
         LA    R1,SUBVALUE              . 1 -> SUBSTITUTION VALUE
         ST    R1,SYMBTESUBTEXTPTR      . SAVE
         DROP  R2
         DROP  R8
         L     R2,@SYMBT                . 2 -> SYMBOL ARRAY HEADER
         USING SYMBT,R2                 . DSECT MAPPING HEADER
         LH    R1,SYMBTNUMBEROFSYMBOLS  . NUMBER OF SYMBOLS IN ARRAY
         LA    R1,1(,R1)                . INCREMENT BY ONE
         STH   R1,SYMBTNUMBEROFSYMBOLS  . SAVE IT
         DROP  R2
         B     G0010
G0008    EQU   *                        . APOSTROPHE IN VALUE
         WTO   MF=(E,WTO9L)
         MVC   RETCODE,=F'8'            . SET RETURN CODE
         B     G0010
G0009    EQU   *                        . SUBSTITUTION VALUE EXCESS
         WTO   MF=(E,WTO10L)
         MVC   RETCODE,=F'8'            . SET RETURN CODE
G0010    EQU   *
         BR    R10
         TITLE 'LOOK FOR SYMBOLIC IN RECORD'
*              REGISTER USAGE
         SPACE 1
*        1                              . WORK
*                                       . USED BY TRT
*        2                              . USED BY TRT (BUT NOT REF)
*        4                              . -> ASASYMBM PARAMETER LIST
*        5                              . -> CURRENT BYTE OF RECORD
*        6                              . -> END OF RECORD
*        7                              . NUMBER OF BYTES TO SEARCH
*        9                              . LENGTH OF RETURNED DATA
*        10                             . RETURN ADDRESS
*                                       . BRANCH ADDRESS
         SPACE 2
*        THIS CODE PROCESSES THE DATA WITHIN AN INPUT RECORD,
*            LOOKING FOR ONE OF THE SPECIFIED SYMBOLICS. IF AN
*            AMPERSAND IS FOUND WITH A NON-BLANK FOLLOWING
*            IMMEDIATELY, IT IS A CANDIDATE IF BETWEEN COLS 1 && 71.
*        THE PROGRAM ASASYMBM IS THEN INVOKED TO VALIDATE THE
*            SYMBOL AND PERFORM SYMBOLIC SUBSTITUTION.
*            ASASYMBM MAY BE INVOKED MULTIPLE TIMES
*              1. DEFAULT SYMBOLS
*              2. DEFAULT SYMBOLS + USER SYMBOLS
*              3. DEFAULT + USER SYMBOLS ON MODIFIED INPUT PATTERN
         SPACE 2
SYMSERCH EQU   *
```

```
         STM   R4,R1Ø,HSAVE             . SAVE REGISTERS
         L     R4,@SYMBP                . 4 -> ASASYMBM PARAMETER LIST
         USING SYMBP,R4                 . DSECT ADDRESSABILITY
         L     R1,@SYMBT                . 1 -> SUBSTITUTION ARRAY
         ST    R1,SYMBPSYMBOLTABLE@     . SAVE IN PARAMETER LIST
         L     R5,@RECORD               . 5 -> START OF RECORD
         ST    R5,SYMBPPATTERN@         . SAVE -> INPUT RECORD
         LA    R6,71(,R5)               . 6 -> END OF RECORD
H0001    EQU   *
         LR    R7,R6                    . 7 -> END OF RECORD
         SLR   R7,R5                    . 7 - LENGTH TO SEARCH
         EX    R7,TRTAMP                . Q. AMPERSAND FOUND?
         BZ    H0003                    . A. NO - PRINT RECORD
         CR    R1,R6                    . Q. AT END OF RECORD?
         BNL   H0003                    . A. YES - PRINT RECORD
         CLI   1(R1),C' '               . Q. AMPERSAND THEN SPACE?
         BNE   H0002                    . A. YES - CANDIDATE
         LA    R5,1(,R1)                . 5 -> NEXT BYTE TO SCAN
         B     H0001                    . CHECK FOR MORE AMPERSANDS
H0002    EQU   *
         LA    R6,WORKREC               . 6 -> SUBSTITUTION AREA
         ST    R6,SYMBPTARGET@          . SAVE ADDRESS
         MVC   SYMLENTR,=F'8Ø'          . TARGET LENGTH INITIAL VALUE
         MVC   LINKX,LINKXL             . MOVE SF=L TO DYNAMIC
         LINKX EP=ASASYMBM,             . LINKX TO ASASYMBM            *
               MF=(E,SYMBP),                                           *
               SF=(E,LINKX)
         CLC   SYMRETCD,=F'Ø'           . Q. RETURN CODE ZERO?
         BNE   H0004                    . A. NO - PRINT RECORD
         L     R5,@RECORD               . 5 -> RECORD
         L     R9,SYMLENTR              . 9 - LENGTH OF RETURNED DATA
         BCTR  R9,RØ                    . DECREMENT FOR EXECUTE
         EX    R9,CLCSUB                . Q. ANY SUBSTITUTION?
         BE    H0003                    . A. NO - OUTPUT RECORD
         LA    R5,WORKREC               . 5 -> SUB 1 AREA
         ST    R5,SYMBPPATTERN@         . SAVE -> INPUT RECORD
         LA    R6,SUBREC                . 6 -> SUB 2 AREA
         ST    R6,SYMBPTARGET@          . SAVE ADDRESS
         MVC   SYMLENTR,=F'8Ø'          . TARGET LENGTH INITIAL VALUE
         MVC   LINKX,LINKXL             . MOVE SF=L TO DYNAMIC
         LINKX EP=ASASYMBM,             . LINKX TO ASASYMBM            *
               MF=(E,SYMBP),                                           *
               SF=(E,LINKX)
         CLC   SYMRETCD,=F'Ø'           . Q. RETURN CODE ZERO?
         BNE   H0004                    . A. NO - OUTPUT RECORD
         L     R9,SYMLENTR              . 9 - LENGTH OF RETURNED DATA
         BCTR  R9,RØ                    . DECREMENT FOR EXECUTE
         EX    R9,CLCSUB                . Q. ANY SUBSTITUTION?
         BE    H0003                    . A. NO - OUTPUT RECORD
         LA    R5,SUBREC                . 5 -> SUB 2 AREA
         ST    R5,SYMBPPATTERN@         . SAVE ADDRESS
         LA    R6,SUBSUBRC              . 6 -> SUB 3 AREA
         ST    R6,SYMBPTARGET@          . SAVE ADDRESS
```

```
         MVC   SYMLENTR,=F'8Ø'          . TARGET LENGTH INITIAL VALUE
         MVC   LINKX,LINKXL             . MOVE SF=L TO DYNAMIC
         LINKX EP=ASASYMBM,             . LINKX TO ASASYMBM            *
               MF=(E,SYMBP),                                           *
               SF=(E,LINKX)
         CLC   SYMRETCD,=F'Ø'           . Q. RETURN CODE ZERO?
         BNE   HØØØ4                    . A. NO - EXIT
HØØØ3    EQU   *
         LA    R6,DYNDCB3               . 6 -> OUTPUT DCB
         PUT   (R6),                    . WRITE OUTPUT                 *
               (R5)
HØØØ4    EQU   *
         LM    R4,R1Ø,HSAVE             . RESTORE REGISTERS
         BR    R1Ø                      . RETURN TO CALLER
CLCSUB   CLC   Ø(*-*,R5),Ø(R6)          . EXECUTED COMPARE
TRTAMP   TRT   Ø(*-*,R5),TRTTAB         . EXECUTED TRT
         DROP  R4
         TITLE 'AMODE SWITCH ROUTINES'
*        AMODE 24 SWITCH
*        REGISTER USAGE
AMODE24  EQU   *
         N     R14,=X'ØØFFFFFF'         . CLEAR HIGH-ORDER BYTE
         BSM   RØ,R14                   . RETURN IN AMODE 24
*        AMODE 31 SWITCH
*        REGISTER USAGE
AMODE31  EQU   *
         N     R14,=X'ØØFFFFFF'         . CLEAR HIGH-ORDER BYTE
         O     R14,=X'8ØØØØØØØ'         . SET AMODE BIT
         BSM   RØ,R14
         DROP  R13
         TITLE 'LIST FORMS OF MACROS'
LINKXL   LINKX SF=L                     . LINKX
LENLINKX EQU   *-LINKXL
OPENL    OPEN  (,,,,,),                 . OPEN / CLOSE                 *
               MF=L
LENOPEN  EQU   *-OPENL
READL    READ  DECBL,                   . BSAM READ                    *
               SF,                                                     *
               ,                                                       *
               ,                                                       *
               'S',                                                    *
               MF=L
LENDECB  EQU   *-READL                  . LENGTH OF DECB
*
WTO1L    WTO   'MISSING OR INVALID JOB-STEP PARAMETER',                 *
               ROUTCDE=11,                                             *
               MF=L
WTO2L    WTO   'PROBLEM OCCURRED OPENING SYSIN FILE',                   *
               ROUTCDE=11,                                             *
               MF=L
WTO3L    WTO   'PROBLEM OCCURRED OPENING SYSPARM FILE',                 *
               ROUTCDE=11,                                             *
```

```
                MF=L
WTO4L     WTO   'PROBLEM OCCURRED OPENING SYSPRINT FILE',              *
                ROUTCDE=11,                                           *
                MF=L
WTO5L     WTO   'SYMBOL LENGTH INVALID - &&XXXXXXXX.',                  *
                ROUTCDE=11,                                           *
                MF=L
WTO6L     WTO   'MORE THAN 128 SYMBOLS PROVIDED',                       *
                ROUTCDE=11,                                           *
                MF=L
WTO7L     WTO   'NO VALID REQUESTS ENTERED',                            *
                ROUTCDE=11,                                           *
                MF=L
WTO8L     WTO   'NO VALUE SUPPLIED WITH SYMBOL',                        *
                ROUTCDE=11,                                           *
                MF=L
WTO9L     WTO   'SUBSTITUTION VALUE CONTAINS EMBEDDED APOSTROPHE',      *
                ROUTCDE=11,                                           *
                MF=L
WTO10L    WTO   'SUBSTITUTION VALUE GREATER THAN 20 BYTES',             *
                ROUTCDE=11,                                           *
                MF=L
          TITLE 'CONSTANTS'
TRTTAB    DC    256X'00'
          ORG   TRTTAB+X'50'
          DC    X'FF'
          ORG
          TITLE 'FORM DCBS - THESE ARE MOVED TO DYNAMIC STORAGE FOR USE'
SYSIN     DCB   DDNAME=SYSIN,            . DCB FOR SYSIN                *
                DSORG=PS,                                             *
                EODAD=E0010,                                          *
                LRECL=80,                                             *
                MACRF=GM,                                             *
                RECFM=F
LENDCB1   EQU   *-SYSIN                  . LENGTH OF DCB
SYSPARM   DCB   BUFNO=1,                 . DCB FOR SYSPARM              *
                DDNAME=SYSPARM,                                       *
                DSORG=PO,                                             *
                EODAD=C0002,                                          *
                MACRF=R
LENDCB2   EQU   *-SYSPARM                . LENGTH OF DCB
SYSPRINT  DCB   DDNAME=SYSPRINT,         . DCB FOR OUTPUT               *
                DSORG=PS,                                             *
                LRECL=80,                                             *
                MACRF=PM,                                             *
                RECFM=F
LENDCB3   EQU   *-SYSPRINT               . LENGTH OF DCB
          TITLE 'DSECTS && LITERAL POOL'
          DCBD  DEVD=DA,                 . DSECT MAPPING DCB            *
                DSORG=PS
          IHADECB                        . DSECT MAPPING DECB
          IEZIOB                         . DSECT MAPPING IOB
```

```
DYNAREA  DSECT                          . DYNAMIC AREA
         DS    18F                      . STANDARD OS SAVE AREA
BSAVE    DS    1F                       . REGISTER SAVE AREA
CSAVE    DS    1F                       . REGISTER SAVE AREA
ESAVE    DS    1F                       . REGISTER SAVE AREA
FSAVE    DS    5F                       . REGISTER SAVE AREA
HSAVE    DS    7F                       . REGISTER SAVE AREA
ISAVE    DS    F                        . REGISTER SAVE AREA
#PDS     DS    F                        . NUMBER OF PDS INPUT
#SYMBOLS DS    F                        . NUMBER OF SYMBOLS INPUT
#SYMBT   DS    F                        . SUBSTITUTION ARRAY SIZE
@BLK     DS    F                        . -> PDS BLOCK
@EXLST   DS    F                        . -> DCB EXIT LIST
@RECORD  DS    F                        . -> PDS RECORD
@SYMBOLS DS    F                        . -> SYMBOL ARRAY
@SYMBP   DS    F                        . -> ASASYMBM PARAMETER LIST
@SYMBT   DS    F                        . -> SUBSTITUTION ARRAY
DMALIST  DS    XL(LENOPEN)              . DATA MANAGEMENT ADDRESS LIST
EXLST    DS    F                        . DCB EXIT LIST
RETCODE  DS    F                        . RETURN CODE
SYMLENTR DS    F                        . ASASYMBM - TARGET LENGTH
SYMRETCD DS    F                        . ASASYMBM - RETURN CODE
TTR      DS    F                        . TRACK ADDRESS OF BLOCK
#BLK     DS    H                        . BLOCKSIZE OF PDS
         DS    ØF
BLDL     DS    CL(LENBLDL)              . BLDL PARAMETER LIST
         DS    ØF
DECBD    DS    XL(LENDECB)              . DYNAMIC DECB
         DS    ØF
LINKX    DS    CL(LENLINKX)             . LINKX LIST FORM
DYNDCB1  DS    XL(LENDCB1)              . AREA FOR DYNAMIC DCB
DYNDCB2  DS    XL(LENDCB2)              . AREA FOR DYNAMIC DCB
DYNDCB3  DS    XL(LENDCB3)              . AREA FOR DYNAMIC DCB
AMPFLAG  DS    C                        . AMPERSAND FLAG
APOSFLAG DS    C                        . APOSTROPHE FLAG
SUBFLAG  DS    C                        . SUBSTITUTION VALUE FLAG
SAVESYM  DS    CL1Ø                     . SYMBOL WORK AREA
SAVEVAL  DS    CL2Ø                     . VALUE WORK AREA
OUTDATA  DS    CL8Ø                     . RECORD AREA
SUBREC   DS    CL8Ø                     . WORK RECORD AREA
SUBSUBRC DS    CL8Ø                     . WORK RECORD AREA
WORKREC  DS    CL8Ø                     . WORK RECORD AREA
DYNLEN   EQU   *-DYNAREA                . LENGTH OF DYNAMIC AREA
         ASASYMBP                       . PARAMETER LIST FOR ASASYMBM
SYMARRAY DSECT                          . SYMBOL ARRAY DSECT
SYMLEN   DS    F                        . LENGTH OF SYMBOL
SYMBOL   DS    CL1Ø                     . SYMBOL VALUE
SUBVALUE DS    CL2Ø                     . SUBSTITUTION VALUE
SYMENT   EQU   *-SYMARRAY               . LENGTH OF ONE ENTRY
*                                       . DSECT MAPPING DIRECTORY ENTRY
BLDLIST  DSECT
BLDL#ENT DS    H                        . NUMBER OF ENTRIES IN LIST
```

```
BLDLENTL DS    H                       . ENTRY LENGTH
DIRMEMNM DS    CL8                     . MEMBER NAME
DIRTTR   DS    CL3                     . TRACK AND RECORD ADDRESS FOR
*                                      .   FIRST BLOCK OF MEMBER
DIRCONCT DS    CL1                     . DATASET CONCATENATION NUMBER
DIRFIND  DS    CL1                     . FOUND FLAG
DIRFLAG  DS    CL1                     . COUNT FIELD
DIRALIAS EQU   X'8Ø'                    . MEMBER IS AN ALIAS
DIRUDATA DS    ØCL62                   . OPTIONAL USER DATA
DIRVERS  DS    XL1                     . VERSION
DIRMODL  DS    XL1                     . MODIFICATION LEVEL
         DS    XL2
DIRCDATE DS    PL4                     . CREATE DATE
DIRMDATE DS    PL4                     . MODIFICATION DATE
DIRMTIME DS    XL2                     . MODIFICATION TIME
DIRCSIZE DS    XL2                     . RECORDS - CURRENT
DIRSINIT DS    XL2                     . RECORDS - INITIAL
DIRMOD   DS    XL2                     . RECORDS - MODIFIED
DIRUSER  DS    CL8                     . USER ID
LENBLDL  EQU   *-BLDLIST               . LENGTH OF DSECT
*                                      . LITERAL POOL AND CSECT END
IEASYMBQ CSECT                         . RE-ESTABLISH CSECT
         LTORG                         . LITERAL POOL
         END
```

REXX EXEC

```
/* REXX ***************************************************************/
/* You must change your.load to the name of the load library where   */
/* the IEASYMBQ program executable is stored.                        */
/* You must change your.ispplib to the name of the panel library     */
/* where the IEASYMBQ panel is stored.                               */

if sysvar(SYSISPF) <> 'ACTIVE' then
    do;
    say "This EXEC requires the ISPF environment to be active";
    exit;
    end;
address ispexec "LIBDEF ISPPLIB DATASET ID('your.ispplib')";

parmlib = '';                       /*   Get first time indicators */
address ispexec 'VGET (PRMLIB) PROFILE';
if PRMLIB <> "N" then               /*    q. First time?           */
    do;                             /*    a. Yes                   */
    PRMLIB = "N";                   /*    Not a second time        */
    PRMLB1 = "";                    /*    First parmlib            */
    PRMVL1 = "";                    /*    First volume             */
    PRMLB2 = "";                    /*    Second parmlib           */
    PRMVL2 = "";                    /*    Second volume            */
    PRMLB3 = "";                    /*    Third parmlib            */
    PRMVL3 = "";                    /*    Third volume             */
```

```
    PRMLB4 = "";                    /*    Fourth parmlib           */
    PRMVL4 = "";                    /*    Fourth volume            */
    address ispexec "VPUT",
        "(PRMLIB,PRMLB1,PRMVL1,",
                "PRMLB2,PRMVL2,",
                "PRMLB3,PRMVL3,",
                "PRMLB4,PRMVL4,",
        ") PROFILE";
    end;
address ispexec "VGET",
        "(PRMLIB,PRMLB1,PRMVL1,",
                "PRMLB2,PRMVL2,",
                "PRMLB3,PRMVL3,",
                "PRMLB4,PRMVL4,",
        ") PROFILE";
MSG="";                             /*    No message               */
BS = outtrap(FREE.);
"FREE F(SYSPRINT)";
BS = outtrap(OFF);

"ALLOC F(SYSPRINT) UNIT(VIO) SP(5 5) TR NEW LR(8Ø)",
     "DSORG(PS) REC(B F S)";
if RC <> Ø then
    do;
    say "SYSPRINT alloc failed.  RC:" RC;
    end;
address ispexec "LMINIT DATAID("IEASYM") DDNAME(SYSPRINT)";
if RC <> Ø then
    do;
    say "LMINIT return code:" RC;
    MSG = ZERRLM;
    end;
SYMBOLS1 = "";                      /*    Remove symbols from input */
SYMBOLS2 = "";
CURPOS = "SYMBOLS1";
do forever;
    address ispexec "CONTROL DISPLAY REFRESH";
    address ispexec "DISPLAY PANEL(IEASYMBQ) CURSOR("CURPOS")";
    if RC <> Ø then                 /*     Q. End requested?       */
        leave;                      /*     A. Yes - out of here    */

    msg = "";                       /*     Null out message        */
    if sysdsn(PRMLB1) <> 'OK' then
        do;
        MSG = "First parmlib not cataloged";
        CURPOS = "PRMLB1";
        end;
    if MSG = "" then
        do;                         /*   Ensure partitioned        */
        BS = listdsi(PRMLB1);
        if SYSDSORG <> "PO" then
            do;
```

```
        MSG = "First parmlib not a pds";
        CURPOS = "PRMLB1";
        end;
    end;
if MSG = "" & PRMLB2 <> "" then
    do;
    if sysdsn(PRMLB2) <> 'OK' then
        do;
        MSG = "Second parmlib not catalogued";
        CURPOS = "PRMLB2";
        end;
                                /*   Ensure partitioned         */
    BS = listdsi(PRMLB2);
    if SYSDSORG <> "PO" then
        do;
        MSG = "Second parmlib not a PDS";
        CURPOS = "PRMLB2";
        end;
    end;
if MSG = "" & PRMLB3 <> "" then
    do;
    if sysdsn(PRMLB3) <> 'OK' then
        do;
        MSG = "Third parmlib not cataloged";
        CURPOS = "PRMLB3";
        end;
                                /*   Ensure partitioned         */
    BS = listdsi(PRMLB3);
    if SYSDSORG <> "PO" then
        do;
        MSG = "Third parmlib not a PDS";
        CURPOS = "PRMLB3";
        end;
    end;
if MSG = "" & PRMLB4 <> "" then
    do;
    if sysdsn(PRMLB4) <> 'OK' then
        do;
        MSG = "Fourth parmlib not catalogued";
        CURPOS = "PRMLB4";
        end;
                                /*   Ensure partitioned         */
    BS = listdsi(PRMLB4);
    if SYSDSORG <> "PO" then
        do;
        MSG = "Fourth parmlib not a PDS";
        CURPOS = "PRMLB4";
        end;
    end;
if MSG = "" then
    do;
    address ispexec "CONTROL ERRORS RETURN";
```

```
    BS = outtrap(FREE.);
    "FREE F(SYSPARM)";
    "ALLOC F(SYSPARM) DA("PRMLB1 PRMLB2 PRMLB3 PRMLB4") SHR REU";
    if RC <> Ø then
        do;
        MSG = "Unable to allocate PARMLIBS";
        CURPOS = "PRMLB1";
        end;
    BS  = outtrap(OFF);
    end;
if MSG = "" then
    do;
    address ispexec "VPUT",
            "(PRMLB1,PRMVL1,",
             "PRMLB2,PRMVL2,",
             "PRMLB3,PRMVL3,",
             "PRMLB4,PRMVL4,",
            ") PROFILE";
    end;
if MSG = "" then
    do;
    queue SYMBOLS1;
    if SYMBOLS2 <> "" then
        do;
        queue SYMBOLS2;
        end;
    queue "";
    BS = outtrap(FREE.);
    "FREE F(SYSIN)";
    BS = outtrap(OFF);
    "ALLOC F(SYSIN) UNIT(VIO) SP(1) TR NEW LR(8Ø) REC(F)";
    "EXECIO * DISKW SYSIN (FINIS";
    "CALL 'your.load(IEASYMBQ)' '"MEMBER"'";
    if RC = Ø then
        do;
        address ispexec "BROWSE DATAID("IEASYM")";
        if RC <> Ø then
            do;
            say "BROWSE return code:" RC;
            end;
        end;
    else
        do;
        MSG = "Error occurred in executing program";
        end;
    "FREE F(SYSIN)";
    "ALLOC F(SYSIN) DA(*)";
    SYMBOLS1 = "";          /*     Remove symbols from input */
    SYMBOLS2 = "";
    MEMBER = "";            /*    Remove member from input */
    CURPOS = "SYMBOLS1";
    end;
```

```
end;
address ispexec "LMFREE DATAID("IEASYM")";
if RC <> Ø then
    do;
    say "LMFREE return code:" RC;
    end;
BS = outtrap(FREE.);
"FREE F(SYSPARM SYSPRINT)";
BS  = outtrap(OFF);
"ALLOC F(SYSPRINT) DA(*)";
exit;
```

ISPF PANEL

```
)ATTR
    ¬ TYPE(OUTPUT) INTENS(HIGH)
)BODY
%———————————— Parmlib Symbolic Change Impact ——————————-
% COMMAND ===>_ZCMD                                    +  Date &ZDATE
%   -MSG                                                 %Time &ZTIME
% Enter symbols ===>_SYMBOLS1                                      +
% Values must be provided.  For example: SYSPLEX=PRDPLEX
% More symbols  ===>_SYMBOLS2                                      +
%
% MEMBER to be processed    ===>_MEMBER                            +
%
% PARMLIB to be searched    ===>_PRMLB1                            +
%
% PARMLIB to be searched    ===>_PRMLB2                            +
%
% PARMLIB to be searched    ===>_PRMLB3                            +
%
% PARMLIB to be searched    ===>_PRMLB4                            +
%
%
%
%
%
%
%
%                 +Press %END KEY+to return to previous panel
)INIT
  &ZCMD = ' '
)PROC
   VER (&SYMBOLS1,NB)
   VER (&MEMBER,NB)
   VER (&PRMLB1,NB)
)END
```

David Loveluck
Systems Programmer (USA)

Getting temporary authorization in key 0

INTRODUCTION

You may have written an SVC to get temporary authorization in key 0 and will probably have had to call it twice in your system code, once before the lines of code that needs authorizing to turn on authorization, and once after to turn off authorization. For those of you who don't want to incur the overhead of SVC calls, a stacking PC routine using cross memory services may be just the answer. First, some background into PC routines and how they are established.

CROSS MEMORY COMMUNICATION

Cross memory communication takes place when a user program issues a PC instruction that specifies a valid PC number. The PC number identifies the PC routine that the system is to invoke. A service provider must have previously defined the PC routine and made the PC number available to the user. The PC routine can provide the requested service or can invoke other programs to provide the service. When the PC routine completes its function, it issues either the PR instruction (for a stacking PC) or a PT instruction (for a basic PC) to return control to the user program.

PC ROUTINES

A PC routine has first to be set up by a service provider. In our case we are making use of the master address space as our service provider. The master address space will make use of MVS macros to make the PC routine available to any address space, current or future. However, before other address spaces can invoke the PC routine, they must first supply the system with a PC number. The PC number will be set-up by our service provider and in this example will be anchored in the sub system vector table in field SSCTSUSE.

STACKING PC ROUTINES

Stacking PC routines are ones that make use of the system-supplied linkage stack for automatically storing/restoring status. Stacking PCs

are faster than basic ones and are recommended. The linkage stack is an area of protected storage that the system gives a program to save status information at a branch or a program call. When a user invokes a stacking PC routine, the system saves the user's environment on the linkage stack. After the PC routine completes its function, it must issue the PR instruction to restore the user's environment and return control to the user.

The following description explains one way that a stacking PC can be set-up under MVS so that it is available to all address spaces.

PARMLIB REQUIREMENTS

Add an entry in SYS1.PARMLIB(IEASSNxx) to set up the stacking PC routine and to define a subsystem vector table to anchor it in. The entry will look something like:

```
PCAP,PCAUTHPG
```

Where PCAP is the subsystem name and PCAUTHPG is the program that will be executed when the subsystem is established.

Please note that the program PCAUTHPG must reside in an authorized library in the linklist and be link-edited with authorization code AC=1.

When PCAUTHPG executes, it will run under the master address space in supervisor state and will set-up the PC routine and anchoring its number in SSCTSUSE.

STACKING PC ROUTINE

```
*********************************************************************
*     Name:  PCAUTHPG                                               *
*     Invoked by: NIP                                               *
*********************************************************************
*     Description                                                   *
*********************************************************************
*     This program acts as the initialization routine for the       *
*     'PCAP'subsystem. PCAP is in fact a pseudo subsystem which     *
*     we use only for its SSVT control block.                       *
*     It is contained in common MVS storage across IPLs and         *
*     hence is a good anchor for our purposes.                      *
```

```
*     We get control in key Ø, supervisor state, from the           *
*     subsystem initialization phase.                               *
*     The program contains code to get the number of our PC         *
*     routine and builds the PC environment using system            *
*     macros LX, ETDEF, ETCRE, and ETCON.                           *
*     The PC number is stored in field SSVTSUSE in the SSVT.        *
*********************************************************************
*     Set up entry requirements:                                    *
*     - Save caller's ARS and GPRS on the linkage stack.            *
*     - Switch to AR addressing mode.                               *
*     - Set up program base and addressing register.                *
*     - Get a re-entrant save area.                                 *
*     - Put savearea address in AR/GPR13.                           *
*     - Put acronym into the savearea backward pointer to indicate  *
*       registers saved on the inkage stack.                        *
*********************************************************************
PCAUTHPG    CSECT                     Module entry point
PCAUTHPG    AMODE 31
PCAUTHPG    RMODE ANY
            BAKR  R14,Ø
            SAC   512
            LAE   R12,Ø(R15,RØ)
            USING PCAUTHPG,R12
            STORAGE OBTAIN,LENGTH=72
            LAE   R13,Ø(RØ,R1)
            MVC   4(4,R13),=C'F1SA'
*           B     MAINSØØØ            Branch around 'eye catcher'
*********************************************************************
*     Eye catcher                                                   *
*********************************************************************
            DC    C'.',C'PCAUTHPG'    Program id
            DC    C'.',C'PC SETUP ROUTINE'
            DC    C'.',C'&SYSDATE'    Date assembled
            DC    C'.',C'&SYSTIME'    Time assembled
*********************************************************************
*     Get address of our SSVT user full word ready to               *
*     store our PC number in.                                       *
*********************************************************************
MAINSØØØ    DS    ØH
            L     R7,16               Address the CVT
            USING CVT,R7                and map it
            L     R8,CVTJESCT
            USING JESCT,R8
            L     R8,JESSSCT
            USING SSCT,R8
MAINSØ2Ø    DS    ØH
            CLC   =C'PCAP',SSCTSNAM   Is this our entry?
            BE    MAINSØ4Ø              Yes - go and process
            ICM   R8,B'1111',SSCTSCTA Next entry available?
            BNZ   MAINSØ2Ø              Yes - go check it
```

```
            B     MAINS81Ø              No - issue error message
*********************************************************************
*     Load PC routine and check it's in low LPA                     *
*********************************************************************
MAINSØ4Ø    DS    ØH
            MODESET KEY=ZERO          Get into key zero
            LOAD  EPLOC=STCKINPC      Load the stacking PC
            LTR   R15,R15             Load sucessful?
            BNZ   MAINS82Ø              No - issue error mess
*********************************************************************
*     Reserve a system Linkage Index                                *
*     to allow the service provider (master address space)          *
*     to connect an entry table to all address spaces               *
*********************************************************************
MAINSØ6Ø    DS    ØH
            LA    R2,1                Set number of LXs
            ST    R2,LXCOUNT          Move into LXRES PLIST
            LXRES LXLIST=LXL,SYSTEM=YES  Get the LX
            LTR   R15,R15             Return code OK?
            BNZ   MAINS83Ø              No - issue error mess
*********************************************************************
*     Issue the ETCRE macro to create an entry table for the        *
*     program call (PC) routine.                                    *
*     The ETD parameter list that ETCRE uses as                     *
*     input to define the name and characteristics of the           *
*     PC routine is created by the ETDEF macro.                     *
*********************************************************************
MAINSØ8Ø    DS    ØH
            ETCRE ENTRIES=ETDESC      Build entry table
            LTR   R15,R15             Build sucessful?
            BNZ   MAINS84Ø              No - issue error mess
            ST    RØ,TKVALUE            Yes - store away token
*********************************************************************
*     Issue the ETCON macro to connect the entry table to the       *
*     specified linkage table index in this the master address      *
*     space and to every linkage table of every address space,      *
*     both present and future.                                      *
*     Input to ETCON is a list of linkage index numbers and         *
*     a list of entry table tokens. The first entry table is        *
*     connected to the first linkage index, the second to the       *
*     second, and so on.                                            *
*********************************************************************
MAINS1ØØ    DS    ØH
            LA    R2,1                Number of entries to connect
            ST    R2,TKCOUNT
            ETCON TKLIST=TKL,LXLIST=LXL
            LTR   R15,R15             Connect successful?
            BNZ   MAINS85Ø              No - issue error mess
*********************************************************************
*     Store PC number in SSVTSUSE                                   *
```

```
*********************************************************************
MAINS12Ø    DS    ØH
            L     R2,LXVALUE          Get the PC number
            ST    R2,SSVTSUSE         Store PC NO in SSVT
MODESET KEY=NZERO                     Finished with key zero
            B     MAINS9ØØ            Exit  RC Ø
*********************************************************************
*     Error Messages                                                *
*********************************************************************
MAINS81Ø    DS    ØH
            WTO   'PCAUTHPG-EØØ1 Unable to find subsystem name PCAP'
            B     MAINS91Ø
MAINS82Ø    DS    ØH
            WTO   'CAUTHPG-EØØ2 Load for PC routine STCKINPC failed'
            B     MAINS91Ø
MAINS83Ø    DS    ØH
            WTO   'PCAUTHPG-EØØ3 Unable to get a system linkage  LX'
            B     MAINS91Ø
MAINS84Ø    DS    ØH
            WTO   'PCAUTHPG-EØØ4 Entry table creation failed'
            B     MAINS91Ø
MAINS85Ø    DS    ØH
            WTO   'PCAUTHPG-EØØ5 Unable to connect table to all users'
            B     MAINS91Ø
MAINS9ØØ    DS    ØH
            WTO   'PCAUTHPG-IØØ1 PC routine setup complete'
            LA    R15,Ø
            B     MAINS999
MAINS91Ø    DS    ØH
            LA    R15,4
*********************************************************************
*     Set up exit requirements                                      *
*     Load save area address and free savearea                      *
*     Restore caller's ARs and GPRS 2-14 and return to caller       *
*********************************************************************
MAINS999    DS    ØH
            LAE   R1,Ø(RØ,R13)
            STORAGE RELEASE,ADDR=(R1),LENGTH=72
            PR
*********************************************************************
*     Working storage                                               *
*********************************************************************
LXL         DS    ØF
LXCOUNT     DS    F
LXVALUE     DS    F
*
TKL         DS    ØF
TKCOUNT     DS    F
TKVALUE     DS    F
*
```

```
ETDESC      ETDEF TYPE=INITIAL
            ETDEF TYPE=ENTRY,PROGRAM=ZAFUXMAA,                          x
            SSWITCH=NO,AKM=(Ø:15),STATE=SUPERVISOR,                     x
            PC=STACKING,EK=Ø,EKM=(Ø:15)
            ETDEF TYPE=FINAL
*
STCKINPC    DC    CL8'STCKINPC'
*****************************
*           Literals         *
*****************************
            LTORG
*****************************
*           Register equates *
*****************************
            REQU
*******************************
*           MVS Dsects         *
*******************************
            IEFJSCVT
            IEFJESCT
            CVT DSECT=YES
            IHAASCB DSECT=YES
            IHAASVT DSECT=YES
            IHAETD
            END
```

A section of application code to call the PC routine:

```
MAINS1ØØ    DS    ØH
            L     R7,16               Address the CVT
                  USING CVT,R7        and map it
            L     R8,CVTJESCT
                  USING JESCT,R8
            L     R8,JESSSCT
                  USING SSCT,R8
MAINS12Ø    DS    ØH
            CLC   =C'PCAP',SSCTSNAM   Is this our entry?
            BE    MAINS14Ø              Yes - go and process
            ICM   R8,B'1111',SSCTSCTA Next entry available?
            BZ    MAINS81Ø              No issue error message
            B     MAINS12Ø              Yes, test next entry
MAINS14Ø    DS    ØH
            L     R14,SSVTSUSE        Get PC number
            LA    R1,MAINS2ØØ         Get start addr of code
*                                     That needs authorization
            BSM   R1,Ø                Store current amode
            ST    R1,WSPCADR          Store ready for PC call
            LA    R1,Ø                Load key Ø value
            SLL   R1,4                Shift ready for SPKA
            ST    R1,WSPCKEY          Store ready for PC call
```

```
            LA    R1,WSPLIST          Address parm list
            PC    Ø(R14)              Go get authorized
            B     MAINS22Ø            Continue with normal code
MAINS2ØØ    DS    ØH
            MVC   CSAFLD1,=C'THIS MOVE CAN NOW UPDATE ANY'
            MVC   CSAFLD2,=C'STORAGE IT LIKES SO BE CAREFUL'
            PR
MAINS22Ø    DS    ØH
......
program continues
.....
*********************************
*     Working storage fields    *
*********************************
.......
.......
WSPLIST     DS    ØF
WSPCADR     DS    F
WSPCKEY     DS    F
........
.......
*********************************
*           MVS DSECTS          *
*********************************
            IEFJSCVT
            IEFJESCT
            CVT DSECT=YES
            IHAASCB DSECT=YES
            IHAASVT DSECT=YES
```

The stacking PC routine:

```
*********************************************************************
*   Program name: STCKPROG                                          *
*   Description   This stacking PC code is just to get the user     *
*                 into supervisor state KEY Ø.                      *
*********************************************************************
            LM    1,2,Ø(1)        Load passed parameters
            SPKA  Ø(2)            Set key
            EREG  2,13            Get caller's registers off stack
            BSM   Ø,1             Branch back to user code
            LTORG
            END
```

An IPL subsystem (part 3)

This month we continue our look at the Initial Program Load Subsystem which reduces the errors inherent in the manual typing and entering of system commands required to activate on-line systems.

```
         SPACE 1
ZNETCANC L     R3,CVTPTR           POINT TO CVT
         USING CVT,R3              ESTABLISH CVT ADDRESSABILITY
         SPACE 1
RETRYIT  BAS   R1Ø,CMRENQ          SERIALIZE ACCESS TO CSCB CHAIN
         L     R4,CVTMSER          DATA AREA OF MSTR SCHD RES DATA AREA
         USING CHAIN,R4            SET ADDRESSABILITY TO CHAIN CSCB
GETNXT   ICM   R4,15,CHPTR         CHAIN POINTER
         BZ    ZNETCAN             IF AT END THEN Z NET,CANCEL
         SPACE 1
         CLC   CHKEY(3),=C'DB2'    TEST FOR DB2*
         BE    PATZNAP             IF SO, TARRY AWHILE
         CLC   CHKEY,=CL8'JES2'    TEST FOR JES2
         BE    GETNXT              IF SO, GET NEXT IN CHAIN
         CLC   CHKEY,PPHJNAME      TEST FOR NET
         BE    GETNXT              IF SO, GET NEXT IN CHAIN
         SPACE 1
         L     R5,CVTASVT          LOGIC TO FIND MATCHING ASCB
         USING ASVT,R5             SET ADDRESSABILITY FOR ASVT
         SPACE 1
         L     R6,ASVTENTY         RETRIEVE ADDRESS OF ASCB
         USING ASCB,R6             SET ADDRESSABILITY FOR ASCB
         SPACE 1
ASCBLOOP ICM   R8,15,ASCBJBNI      SET R8 TO ADDRESS OF JOBNAME
         CLC   Ø(Ø,R8),CHCLS       MATCH IT TO CSCB JOBNAME
         BE    CKPERF              IF A MATCH THEN CHECK PERF GROUP
         ICM   R6,15,ASCBFWDP      ELSE MOVE TO NEXT ASCB
         BNE   ASCBLOOP            CHECK AGAIN FOR A MATCH
         B     GETNXT              GET NEXT IN CSCB CHAIN
         SPACE 1
CKPERF   L     R1,ASCBOUCB         POINTER TO OUCB
         USING OUCB,R1             SET ADDRESSABILITY FOR OUCB
         CLC   OUCBSPG,=H'17'      TEST IF PERF GROUP 17(STARTED TASKS)
         BE    GETNXT              IF SO, SNUB IT
         CLC   OUCBSPG,=H'ØØ'      CHECK FOR PERF GROUP ZERO
         BE    GETNXT              IF NOT PRESENT, THEN IGNORE IT
         SPACE 1
PATZNAP  BAS   R1Ø,CMRDEQ          REMOVE SERIALIZATION OF CSCB CHAIN
         STIMER WAIT,BINTVL=CLAMTIME SLEEP FOR 6 SECONDS
         B     RETRYIT             LOOP IT AGAIN
         DROP  R3,R4,R5,R6,R1
         EJECT
ZNETCAN  BAS   R1Ø,CMRDEQ          REMOVE SERIALIZATION OF CSCB CHAIN
         MVC   WTOMSG(16),PATCANCL SET 'Z NET,CANCEL' IN COMMAND AREA
```

```
         BAS   R1Ø,PATSVC34        TERMINATE TELEPROCESSING ENVIRONMENT
         BAS   R1Ø,PATREST         MAKE IT A PREGNANT PAUSE
         SPACE 1
         MVC   COMMNDWK(2Ø),PPGPLLA STOP LLA
         BAS   R1Ø,PATSVC34        ISSUE COMMAND
         SPACE 1
         MVC   COMMNDWK+2(3),PPGPVLF STOP VLF
         BAS   R1Ø,PATSVC34        ISSUE COMMAND
         SPACE 1
         MVC   COMMNDWK(14),PPGPTM STOP THRUPUT MANAGER
         BAS   R1Ø,PATSVC34        ISSUE COMMAND
         B     DCABORT             CLEAN UP AND TERMINATE DCIPLS
         EJECT
*********************************************************************
*        BUILD ACTION TABLES FROM NCP25.AGØ3ZGEN.NCPA ON VTM2ØØ     *
*********************************************************************
         SPACE 1
READCMDS L     15,=A(TO24A)        GO BACK TO 24
         BSM   Ø,15
TO24A    DS    ØH                  BACK TO 24
         SR    R2,R2               SET SUBPOOL ZERO
         LA    RØ,8Ø               SET LENGTH FOR GETMAIN
         LR    R4,R1Ø              PRESERVE RETURN ADDRESS
         BAS   R1Ø,CPSTORA         OBTAIN STORAGE FOR 1ST COMMAND
         ST    R1,CMDA1STC         SAVE 1ST COMMAND STORAGE ADDRESS
         ST    R4,CMDBASAV         SAVE BAS RETURN
         LR    R6,R1               SAVE 1ST COMMAND STORAGE ADDRESS
         MVI   Ø(R1),C' '          PREPARE TO CLEAR ACQUIRED STORAGE
         MVC   1(79,R1),Ø(R1)      CLEAR ACQUIRED STORAGE
         XC    Ø(4,R1),Ø(R1)       ZERO 1ST CHAIN POINTER OF COMMAND
         LA    R1Ø,CMDDCB          POINT TO COMMAND DCB
         USING IHADCB,R1Ø          ESTABLISH DCB ADDRESSABILITY
         OPEN  (CMDDCB)            PREPARE DATASET FOR ACCESS
         LH    RØ,=H'8Ø'           SET DEFAULT FOR GETMAIN OF I/O AREA
         SR    R2,R2               SET TO GET STORAGE FROM SUBPOOL Ø
         TM    DCBOFLGS,DCBOFOPN   TEST IF DATASET OPENED SUCCESSFULLY
         BNO   CMDMAIN             BRANCH IF SO
         LH    RØ,DCBBLKSI         FETCH BLOCK-SIZE OF DATASET
CMDMAIN  BAS   R1Ø,CPSTORA         OBTAIN AN I/O AREA FOR CMDDCB
         LA    R1Ø,CMDDCB          POINT TO COMMAND DCB
         LR    R5,R1                 SAVE AREA ADDRESS
         LR    R4,R1                 SAVE AREA ADDRESS
         TM    DCBOFLGS,DCBOFOPN   TEST IF DATASET OPENED SUCCESSFULLY
         BNO   CMDERR1             NOTIFY MTO AND TERMINATE
         SPACE 1
         BLDL  CMDDCB,CMDPARM      GET DIRECTORY INFO
         LTR   R15,R15             TEST RETURN CODE
         BNE   CMDERR2             NOTIFY MTO AND TERMINATE
         SPACE 2
         POINT CMDDCB,CMDBLDLT     POINT TO START OF COMMAND ADDRESS
```

```
         EJECT
*************************************************************************
*        RETRIEVE NAMES OF TASKS THAT ARE TO BE TERMINATED AND THE      *
*        COMMANDS THAT WILL BE USED TO TERMINATE THEM.                  *
*************************************************************************
         SPACE
         LA    R2,8Ø               LOAD RECORD SIZE
READNEXT LR    R5,R4               RESET R5 AFTER BXLE ATTACK
         SPACE
         READ  DECB2,SF,CMDDCB,(5),'S'   READ A DATA BLOCK
         CHECK DECB2               WAIT FOR COMPLETION OF I/O
         SPACE
         LH    R3,DCBBLKSI         LOAD MAXIMUM BLOCK SIZE
         L     R15,DECB2+16        IOB ADDRESS
         SH    R3,14(R15)          CALCULATE SIZE OF BLOCK READ
         SR    R3,R2               REDUCE BY 8Ø BYTES
         LA    R3,Ø(R3,R5)         CALCULATE STOPPING POINT
SAVECMD  MVC   4(76,R6),Ø(R5)      MOVE COMMAND TO STORAGE
         LR    RØ,R2               SET LENGTH FOR GETMAIN
         SR    R2,R2               SET SUBPOOL ZERO
         BAL   R1Ø,CPSTORA         OBTAIN STORAGE FOR NEXT COMMAND
         LA    R1Ø,CMDDCB          POINT TO COMMAND DCB
         LA    R2,8Ø               LOAD RECORD SIZE
         ST    R1,Ø(R6)            SAVE NEXT POINTER IN PREVIOUS
         LR    R6,R1               SAVE NEXT POINTER IN R6
         MVI   Ø(R1),C' '          PREPARE TO CLEAR ACQUIRED STORAGE
         MVC   1(79,R1),Ø(R1)      CLEAR ACQUIRED STORAGE
         XC    Ø(4,R1),Ø(R1)       ZERO CHAIN POINTER FOR END INDICATOR
         BXLE  R5,R2,SAVECMD       GET NEXT RECORD
         B     READNEXT            READ NEXT BLOCK
         EJECT
*************************************************************************
*        ERRORS FROM READCMDS                                           *
*************************************************************************
         SPACE 1
CMDERR1  WTO  'DCIPL2ØI  UNABLE TO OPEN COMMAND DATASET'
CPNOTIFY WTO  'DCIPL22A  NOTIFY SYSTEM PROGRAMMER'
         B     CMDEND1             BRANCH TO END READCMDS
         SPACE 1
         SPACE 1
CMDERR2  WTO  'DCIPL21E  BUILD FAILED FOR COMMAND DIRECTORY'
         B     CPNOTIFY            BRANCH TO END READCMDS
         EJECT
*************************************************************************
*        END OF READCMDS                                                *
*************************************************************************
         SPACE 1
CMDEND1  LR    R1,R6               POINT TO VOIDED COMMAND AREA
         SR    R2,R2               SET SUBPOOL ZERO
         LA    RØ,8Ø               SET LENGTH OF EIGHTY
         BAL   R1Ø,CPSTORF         RELEASE VOIDED COMMAND AREA
         LA    R1Ø,CMDDCB          POINT TO COMMAND DCB
```

```
         SPACE
         LA    RØ,8Ø               SET DEFAULT FOR FREEMAIN
         TM    DCBOFLGS,DCBOFOPN   TEST IF DATASET OPENED SUCCESSFULLY
         BNO   CMDFREE1            IF NOT, FREE DEFAULT
         LH    RØ,DCBBLKSI         SIZE OF GETMAINED AREA
CMDFREE1 LR    R1,R4               POINT TO I/O AREA
         SR    R2,R2               SET SUBPOOL ZERO
         BAL   R1Ø,CPSTORF         RELEASE I/O AREA
         LA    R1Ø,CMDDCB          POINT TO COMMAND DCB
         SPACE
         TM    DCBOFLGS,DCBOFOPN   TEST IF DATASET OPENED SUCCESSFULLY
         BNO   CMDRTRN1            IF NOT, SKIP CLOSE
         CLOSE (CMDDCB)            DONE WITH DATASET
CMDRTRN1 B     DCABORT             CLEAN AND TERMINATE DCIPLES
         SPACE
CMDEND2  LA    RØ,8Ø               SET DEFAULT FOR FREEMAIN
         TM    DCBOFLGS,DCBOFOPN   TEST IF DATASET OPENED SUCCESSFULLY
         BNO   CMDFREE2            IF NOT, FREE DEFAULT
         LH    RØ,DCBBLKSI         SIZE OF GETMAINED AREA
CMDFREE2 LR    R1,R4               POINT TO I/O AREA
         SR    R2,R2               SET SUBPOOL ZERO
         BAL   R1Ø,CPSTORF         RELEASE I/O AREA
         LA    R1Ø,CMDDCB          POINT TO COMMAND DCB
         SPACE
         TM    DCBOFLGS,DCBOFOPN   TEST IF DATASET OPENED SUCCESSFULLY
         BNO   CMDRTRN2            IF NOT, SKIP CLOSE
         CLOSE (CMDDCB)            DONE WITH DATASET
CMDRTRN2 L     R1Ø,CMDBASAV        RETURN BAS ADDRESS
         SPACE 1
         L     15,=A(TO31A+X'8ØØØØØØØ') GO TO 31 FOR DCIPLS LIVES THERE
         BSM   Ø,15
TO31A    DS    ØH
         BR    R1Ø                 BRANCH BACK TO BAS INSTRUCTION
         DROP  R1Ø                 DROP RØ R1Ø
         EJECT
*************************************************************************
*   FREE COMMAND AREAS ACQUIRED IN READCMDS PROCESS ABOVE               *
*************************************************************************
         SPACE 1
CMDFREE  L     R3,CMDA1STC         LOAD ADDRESS OF FIRST COMMAND
         LR    R5,R1Ø              PRESERVE RETURN ADDRESS
CMDFLOOP CLC   Ø(4,R3),=X'ØØØØØØØØ' CHECK FOR LAST COMMAND IN CHAIN
         BE    CMDFRTRN            GO TO FREE LAST AND RETURN
         L     R4,Ø(R3)            SAVE POINTER TO NEXT IN CHAIN
         LA    RØ,8Ø               SET LENGTH OF COMMAND STORAGE AREA
         LR    R1,R3               SET ADDRESS OF COMMAND STORAGE AREA
         SR    R2,R2               SET SUBPOOL OF COMMAND STORAGE AREA
         BAS   R1Ø,CPSTORF         FREE COMMAND STORAGE AREA
         LR    R3,R4               GET NEXT COMMAND TO BE FREED
         B     CMDFLOOP            FREE NEXT COMMAND AREA
         SPACE
CMDFRTRN LA    RØ,8Ø               SET LENGTH OF COMMAND STORAGE AREA
```

```
         LR    R1,R3               SET ADDRESS OF COMMAND STORAGE AREA
         BAS   R1Ø,CPSTORF         FREE THE LAST ONE ACQUIRED
         SPACE 1
         BR    R5                  RETURN
         EJECT
*********************************************************************
*        ENTER CODE TO SEE IF PRODUCT IS ALREADY ACTIVE             *
*********************************************************************
         SPACE 1
CPACTIVE ST    R1Ø,CMDR1SAV        PRESERVE RETURN ADDRESS
         BAS   R1Ø,CMRENQ          SERIALIZE ACCESS TO CSCB CHAIN
         L     R4,CVTPTR           POINT TO CVT
         USING CVT,R4              ESTABLISH CVT ADDRESSABILITY
         L     R5,CVTMSER          DATA AREA OF MSTR SCHD RES DATA AREA
         USING CHAIN,R5            SET ADDRESSABILITY TO CHAIN CSCB
         SPACE 1
CPNEXT   ICM   R5,15,CHPTR         CHAIN POINTER
         BZ    CPEXIT              IF AT END THEN RETURN
         SPACE 1
         CLC   33(8,R3),CHKEY      SEE IF PRODUCT ALREADY ACTIVE
         BNE   CPNEXT              CONTINUE TO LOOP
         MVI   41(R3),C'A'         INDICATE PRODUCT IS ALREADY ACTIVE
         SPACE 1
CPEXIT   BAS   R1Ø,CMRDEQ          REMOVE SERIALIZATION OF CSCB CHAIN
         L     R1Ø,CMDR1SAV        RETRIEVE RETURN ADDRESS
         BR    R1Ø                 EXIT THIS ROUTINE
         DROP  R4,R5
         EJECT
*********************************************************************
*  ENTER CODE TO VERIFY ALL PRODUCTS DEFINED TO DCIPLES             *
*  ARE UP                                                           *
*********************************************************************
         SPACE 1
CPVERUP  CLC   CMDSYSID(4),=C'VSØ5'  WHAT SYSTEM IS IT??
         BE    CPVERPRU            BRANCH TO PRODUCTION
         CLC   CMDSYSID(4),=C'VSØ4'  WHAT SYSTEM IS IT??
         BE    CPVERACU            BRANCH TO ACCENT
         CLC   CMDSYSID(4),=C'VSØ3'  WHAT SYSTEM IS IT??
         BE    CPVERTEU            BRANCH TO TECHNOLOGY
         CLC   CMDSYSID(4),=C'VSØ1'  WHAT SYSTEM IS IT??
         BE    CPVERDEU            BRANCH TO DEVELOPMENT
         CLC   CMDSYSID(4),=C'VSØ2' TEST IF MILLENNIUM SYSTEM
         BE    CPVERYMM            BRANCH TO MILLENNIUM
         BAS   R1Ø,CPNOSYS         ISSUE AN APPROPRITE ERROR MESSAGE
         B     DCABORT             TERMINATE DCIPLES
         SPACE 1
CPVERYMM MVC   CMDPARM(16),CMDDIR29 SAVE PARM FOR MILLENNIUM
         B     CPVERUØ
         SPACE 1
CPVERPRU MVC   CMDPARM(16),CMDDIR1Ø SAVE PARM FOR PRODUCTION
         B     CPVERUØ
         SPACE 1
```

```
CPVERACU MVC   CMDPARM(16),CMDDIRØ7 SAVE PARM FOR ACCENT
         B     CPVERUØ
         SPACE 1
CPVERTEU MVC   CMDPARM(16),CMDDIRØ1 SAVE PARM FOR TECHNOLOGY
         B     CPVERUØ
         SPACE 1
CPVERDEU MVC   CMDPARM(16),CMDDIRØ4 SAVE PARM FOR DEVELOPMENT
         EJECT
CPVERUØ  BAS   R1Ø,READCMDS        BRANCH TO BUILD THIS COMMAND TABLE
         L     R3,CMDA1STC         LOAD ADDRESS OF 1ST COMMAND
         L     R4,CVTPTR           POINT TO CVT
         USING CVT,R4              ESTABLISH CVT ADDRESSABILITY
         L     R5,CVTMSER          DATA AREA OF MSTR SCHD RES DATA AREA
         USING CHAIN,R5            SET ADDRESSABILITY TO CHAIN CSCB
         SPACE 1
CPVERU1  CLC   Ø(4,R3),=X'ØØØØØØØØ' CHECK FOR END OF COMMAND CHAIN
         BE    CPVERU5                 IF SO START TO TERMINATE
         CLI   41(R3),C'P'         IS IT A PRODUCT BEING TESTED
         BNE   CPVERU4             GET NEXT COMMAND
         SPACE 1
         SPACE 1
CPVERU2  ICM   R5,15,CHPTR         CHAIN POINTER
         BZ    CPVERU3             IF AT END THEN DISPLAY FAILURE
         SPACE 1
         CLC   33(8,R3),CHKEY      SEE IF PRODUCT IS ACTIVE
         BNE   CPVERU2             IF NOT, CONTINUE LOOP
         B     CPVERU4              ELSE GET NEXT COMMAND FROM TABLE
         SPACE 1
CPVERU3  MVC   COMMNDWK(43),WTOMSG1 MOVE IN VERIFICATION FAIL FORMAT
         MVC   COMMNDWK+36(8),33(R3) FILL IN NAME OF FAILED APPLICATION
         LA    R1,5Ø+4             LENGTH OF EACH COMMAND PLUS CONSTANT
         STH   R1,WTOMSG           SET LENGTH IN INTERNAL COMMAND
         WTO   MF=(E,WORK),DESC=2,ROUTCDE=8
         MVI   CMDFAILI,C'U'       INDICATE THAT A FAILURE HAS OCCURRED
         SPACE 1
CPVERU4  L     R3,Ø(R3)            LOAD ADDRESS OF NEXT COMMAND
         L     R5,CVTMSER          DATA AREA OF MSTR SCHD RES DATA AREA
         B     CPVERU1
         SPACE 1
CPVERU5  CLI   CMDFAILI,C'U'       CHECK TO SEE IF ANY FAILURES
         BE    CPVERU6             FAILURES PREVIOUSLY DISPLAYED
         WTO   'DCIPLØ6I - VERIFICATION SUCCESSFUL',DESC=2,ROUTCDE=8
CPVERU6  BAS   R1Ø,CMDFREE         FREE COMMAND AREAS ACQUIRED
         B     DCABORT             TERMINATE DCIPLES
         DROP  R4,R5
         EJECT
*************************************************************
*  ENTER CODE TO VERIFY ALL PRODUCTS DEFINED TO DCIPLES    *
*  ARE DOWN                                                *
*************************************************************
         SPACE 1
CPVERDWN CLC   CMDSYSID(4),=C'VSØ5' WHAT SYSTEM IS IT??
```

```
         BE    CPVERPRD            BRANCH TO PRODUCTION
         CLC   CMDSYSID(4),=C'VSØ4' WHAT SYSTEM IS IT??
         BE    CPVERACD            BRANCH TO ACCENT
         CLC   CMDSYSID(4),=C'VSØ3' WHAT SYSTEM IS IT??
         BE    CPVERTED            BRANCH TO TECHNOLOGY
         CLC   CMDSYSID(4),=C'VSØ1' WHAT SYSTEM IS IT??
         BE    CPVERDED            BRANCH TO DEVELOPMENT
         CLC   CMDSYSID(4),=C'VSØ2' TEST IF YMM SYSTEM
         BE    CPVERY2K            BRANCH TO MILLENNIUM
         BAS   R1Ø,CPNOSYS         ISSUE AN APPROPRITE ERROR MESSAGE
         B     DCABORT             TERMINATE DCIPLES
         SPACE
CPNOSYS  WTO   'DCIPLØ4I  UNABLE TO DETERMINE SYSTEM IDENTIFICATION;  VE
               ERIFICATION PROCESS TERMINATED'
         BR    R1Ø                 RETURN TO CALLER
         SPACE 1
CPVERPRD MVC   CMDPARM(16),CMDDIR1Ø SAVE PARM FOR PRODUCTION
         B     CPVERDØ
         SPACE 1
CPVERACD MVC   CMDPARM(16),CMDDIRØ7 SAVE PARM FOR ACCENT
         B     CPVERDØ
         SPACE 1
CPVERY2K MVC   CMDPARM(16),CMDDIR29 SAVE PARM FOR MILLENNIUM
         B     CPVERDØ
         SPACE 1
CPVERTED MVC   CMDPARM(16),CMDDIRØ1 SAVE PARM FOR TECHNOLOGY
         B     CPVERDØ
         SPACE 1
CPVERDED MVC   CMDPARM(16),CMDDIRØ4 SAVE PARM FOR DEVELOPMENT
         EJECT
CPVERDØ  BAS   R1Ø,READCMDS        BRANCH TO BUILD THIS COMMAND TABLE
         L     R3,CMDA1STC         LOAD ADDRESS OF 1ST COMMAND
         L     R4,CVTPTR           POINT TO CVT
         USING CVT,R4              ESTABLISH CVT ADDRESSABILITY
CPVERENQ BAS   R1Ø,CMRENQ          SERIALIZE ACCESS TO THE CSCB CHAIN
CPVERD1  L     R5,CVTMSER          DATA AREA OF MSTR SCHD RES DATA AREA
         USING CHAIN,R5            SET ADDRESSABILITY TO CHAIN CSCB
         SPACE 1
         CLC   Ø(4,R3),=X'ØØØØØØØØ' CHECK FOR END OF COMMAND CHAIN
         BE    CPVERD5                 IF SO START TO TERMINATE
         CLI   42(R3),C'Q'         TEST IF IT IS A PRODUCT BEING TESTED
         BNE   CPVERD4             BRANCH IF NOT TO GET THE NEXT ONE
         SPACE 2
CPVERD2  ICM   R5,15,CHPTR         CHAIN POINTER
         BZ    CPVERD4             IF AT END, GET NEXT COMMAND
         SPACE 1
         CLC   33(8,R3),CHKEY      IF PRODUCT REMAINS ACTIVE, THEN
         BNE   CPVERD2              DISPLAY IT, ELSE TEST NEXT PRODUCT
         SPACE 2
         MVC   COMMNDWK(46),WTOMSG3 MOVE IN VERIFICATION FAIL FORMAT
         MVC   COMMNDWK+38(8),33(R3) FILL IN NAME OF FAILED APPLICATION
         LA    R1,5Ø+4             LENGTH OF EACH COMMAND PLUS CONSTANT
```

```
         STH   R1,WTOMSG           SET LENGTH IN INTERNAL COMMAND
         BAS   R1Ø,CMRDEQ          RELEASE LOCK ON THE CSCB CHAIN
         WTO   MF=(E,WORK),DESC=2,ROUTCDE=8
         BAS   R1Ø,CMRENQ          SERIALIZE ACCESS TO THE CSCB CHAIN
         MVI   CMDFAILI,C'D'       INDICATE THAT A FAILURE HAS OCCURRED
         SPACE 1
CPVERD4  L     R3,Ø(R3)            LOAD ADDRESS OF NEXT COMMAND
         B     CPVERD1
         SPACE 1
CPVERD5  BAS   R1Ø,CMRDEQ          RELEASE LOCK ON THE CSCB CHAIN
         CLI   CMDFAILI,C'D'       CHECK TO SEE IF ANY FAILURES
         BE    CPVERD6             FAILURES PREVIOUSLY DISPLAYED
         WTO   'DCIPLØ6I - VERIFICATION SUCCESSFUL',DESC=2,ROUTCDE=8
CPVERD6  BAS   R1Ø,CMDFREE         FREE COMMAND AREAS ACQUIRED
         B     DCABORT             TERMINATE DCIPLES
         SPACE 1
         DROP  R4,R5               FORGET CVT AND CSCB
         EJECT
*************************************************************
*  ENTER CODE TO DISPLAY CHKEY OF ALL PRODUCTS —FOR      *
*  SYSTEM PROGRAMMER USE.. K S,DEL=R                       *
*  AN EXPOSURE EXITS HERE - THE CSCB CHAIN MAY CHANGE...   *
*************************************************************
         SPACE 1
CPCHKEY  L     R4,CVTPTR           POINT TO CVT
         USING CVT,R4              ESTABLISH CVT ADDRESSABILITY
         L     R5,CVTMSER          DATA AREA OF MSTR SCHD RES DATA AREA
         USING CHAIN,R5            SET ADDRESSABILITY TO CHAIN CSCB
         SPACE 1
CPCHKØ1  ICM   R5,15,CHPTR         CHAIN POINTER
         BZ    DCABORT             IF AT END THEN TERMINAT
         MVC   COMMNDWK(42),WTOMSG2 MOVE IN CHKEY FORMAT
         MVC   COMMNDWK+35(8),CHKEY  FILL IN NAME OF CHKEY
         LA    R1,5Ø+4             LENGTH OF EACH COMMAND PLUS CONSTANT
         STH   R1,WTOMSG           SET LENGTH IN INTERNAL COMMAND
         WTO   MF=(E,WORK),DESC=6,ROUTCDE=8
         B     CPCHKØ1             GET NEXT COMMAND
         SPACE 1
         DROP  R4,R5
         EJECT
*********************************************************************
*   RESPOND APPROPRIATELY TO OUTSTANDING WTORS.                     *
*********************************************************************
         SPACE 1
PATDOORE BAS   R1Ø,PATREST         TARRY, MAYBE TSO HAS ACTIVE USERS
         SPACE 1
         ESAR  R1                  OBTAIN SECONDARY ADDRESS SPACE ID
         ST    R1,PATASID          STOW IT
         SPACE 1
         L     R3,CVTPTR           ADDRESS OF CVT
         USING CVT,R3              ESTABLISH CVT ADDRESSABILITY
         SPACE 1
```

```
         L     R5,CVTASVT         FETCH ADDRESS OF ASVT
         USING ASVT,R5
         L     R4,ASVTMAXU         MAXIMUM NUMBER OF ADDRESS SPACES
         SPACE 1
PATLOC   TM    ASVTENTY,ASVTAVAL   TEST IF ENTRY IS AVAILABLE
         BO    PATGRUVE            BRANCH IF SO
         SPACE 1
         L     R6,ASVTENTY         RETRIEVE ADDRESS OF ASCB
         USING ASCB,R6             ESTABLISH ASCB ADDRESSABILITY
         SPACE 1
         ICM   R1,15,ASCBJBNI      POINTER TO INITIATED JOBNAME
         BZ    COTJBNI
         SPACE 1
         CLC   Ø(8,R1),=CL8'CONSOLE' TEST IF CONSOLE ADDRESS SPACE
         BNE   PATGRUVE
         B     PATGOTIT
         SPACE 1
COTJBNI  EQU   *
         SPACE 1
         ICM   R1,15,ASCBJBNS      POINTER TO START/MOUNT/LOGON TASK
         BZ    PATGRUVE            FORMAT IT
         SPACE 1
         CLC   Ø(8,R1),=CL8'CONSOLE' TEST IF CONSOLE ADDRESS SPACE
         BE    PATGOTIT            BRANCH IF SO
         SPACE 1
PATGRUVE LA    R5,4(R5)            NEXT ENTRY
         BCT   R4,PATLOC           LOOP POWER
         B     DCABORT             BACK TO DUST
         EJECT
PATGOTIT LH    R2,ASCBASID         OBTAIN ASID OF CONSOLE ADR SPACE
         DROP  R5,R6               FORGET ASVT, ASCB
         SPACE 1
         LA    R1,1                SET AUTHORIZATION
         AXSET AX=(R1)               INDEX TO ONE
         SSAR  R2                  USE DATA IN ADDRESS SPACE OF CONSOLE
         SAC   512                 UNIVERSAL ACCESS MODE
         SPACE 1
         LA    R1,21               SET LENGTH OF REPLY TO FIFTEEN
         STH   R1,WTOMSG             FOR OPERATOR REPLY
         SPACE 1
         L     R1,PATA31           POINT TO 31-BIT ADDRESS MODE
         BSM   RØ,R1               ENTER 31-BIT AMODE
         SPACE 1
PAT31    L     R4,CVTCUCB          ADDRESS OF UCM BASE
         USING UCM,R4              SET UCM ADDRESSABILITY
         LAM   R4,R6,PATONE        INITIALIZE ACCESS REGISTERS
         ICM   R5,15,UCMRPYQ       ADDRESS OF FIRST ORE
         BZ    PATENDE             BRANCH IF NONE
         USING OREF,R5             SET ORE ADDRESSABILITY
         SPACE 1
PATCONT  L     R6,ORERWQE          ADDRESS OF ASSOCIATED REAL WQE
         USING WQE,R6              SET WQE ADDRESSABILITY
```

```
         SPACE 1
         LH    R1,WQETXTLN         ACTUAL LENGTH OF TEXT OF MESSAGE
         CH    R1,DCIP12           TEST IF LENGTH MEETS MINIMUMS
         BL    PATNXORE            BRANCH IF NOT LARGE ENOUGH
         EJECT
         L     R3,PATVTAB          TABLE OF WTOR'S MESSAGES
         SPACE 1
PATGETWT CLC   4(8,R3),WQETXT+4    WAS RESPONSE PROVIDED FOR THIS WTOR?
         BNE   PATMISST            BRANCH IF NOT
         SPACE
         CLC   WQESYSNM(4),SYSID   TEST IF IT'S FOR THIS SYSTEM
         BNE   PATMISST            BRANCH IF NOT
         SPACE
         CLC   17(6,R3),=C'LINK 2' CHK FOR /RST OF LINK ON PROD IMS1
         BNE   CONTXXX             BRANCH IF NOT
         CLC   WQETXT+25(4),=C'IM1S' CHK TO SEE IF WE GOT IMS1 WTOR
         BNE   PATMISST            FORGET IT AND GO TO NEXT WTOR
CONTXXX  MVC   COMMNDWK+2(15),12(R3)  SET RESPONSE INTO COMMAND AREA
         MVC   COMMNDWK(2),WQETXT+1 SET RESPONSE INTO COMMAND AREA
         SPACE 1
         L     R1,PATASID          STOW IT
         SSAR  R1                  SET TO SECONDARY TO CURRENT
         SAC   Ø                   ACCESS DATA ONLY WITHIN THIS ASID
         SPACE 1
         BAS   R1Ø,PATSVC34        RESPOND TO WTOR
         SAC   512                 OBTAIN UNIVERSAL ACCESS TO DATA
         SSAR  R2                  USE DATA IN ADDRESS SPACE ON CONSOLE
         SPACE 1
         B     PATNXORE            POINT TO NEXT WTOR
         SPACE 1
PATMISST CLC   Ø(4,R3),=X'ØØØØØØØØ' CHECK FOR LAST COMMAND IN CHAIN
         BE    PATNXORE            CHECK FOR MORE
         L     R3,Ø(R3)            LOAD ADDRESS OF NEXT COMMAND
         B     PATGETWT            LOOP POWER
         SPACE 1
PATNXORE ICM   R5,15,ORELKP        OBTAIN ADDRESS OF NEXT ORE
         BNZ   PATCONT             IF ONE IS AVAILABLE, PROCESS IT
         SPACE 1
PATENDE  EQU   *
         SAC   Ø                   RETURN CONTROL OF ADR SPC TO PRIMARY
         L     R1,PATASID          RETRIEVE ORIGINAL SECONDARY ASID
         SSAR  R1                  SET SECONDARY TO CURRENT
         SPACE 1
         DROP  R3,R6               FORGET CVT
         BR    R8                  RETURN
         TITLE ' TERMINATION PROCESSING.'
*************************************************************************
*     CLEANUP, THEN TERMINATE.                                          *
*************************************************************************
DCABORT  LA    R1,PAT24            SET HI-ORDER BIT TO A ZERO
         BSM   RØ,R1               REENTER 24-BIT ADDRESSING MODE
         SPACE 1
```

```
PAT24    WTO   'DCIPL57I  COMMAND SUBSYSTEM TERMINATING'
         CNOP  Ø,4
         BAL   1,IHBØØ83A          BRANCH AROUND MESSAGE
         DC    AL2(43)             TEXT LENGTH
         DC    B'ØØØØØØØØØØØØØØØØ' MCSFLAGS
         DC    C'DCIPL57I  COMMAND SUBSYSTEM TERMINATING
                                   MESSAGE TEXT
         DS    OH
         SVC   35                  ISSUE SVC 35
         DELETE EP=DCIPLSFR        REMOVE SUBSYS FUNCT RTN FROM SYSTEM
         SPACE 2
         L     RØ,WORKSP           SIZE OF WTO AREA
         LR    R1,R9               ADDRESS OF WTO AREA
         LA    R2,252              SUBPOOL OF WTO AREA
         BAS   R1Ø,CPSTORF         RELEASE WTO AREA
         TITLE  ' DE-ACTIVATE COMMAND SUBSYSTEM.'
         L     R1,CVTPTR           OBTAIN POINTER TO THE CVT
         USING CVTMAP,R1           ESTABLISH CVT ADDRESSABILITY
         SPACE 1
         L     R1,CVTJESCT         FETCH POINTER TO THE JESCT
         USING JESCT,R1            ESTABLISH JESCT ADDRESSABILITY
         SPACE 1
         ICM   R6,15,JESSSCT       FETCH POINTER TO THE SSCVT
         USING SSCT,R6             ESTABLISH SSCVT ADDRESSABILITY
         SPACE 1
GETSSCVT BZ    CANESTAE            IF END OF SSCVT, THEN BRANCH.
         CLC   SSCTSNAM,PATNAME    IF CORRECT SUBSYSTEM NAME
         BE    GOTSSCVT             THEN PROCESS
         ICM   R6,15,SSCTSCTA        ELSE LOAD POINTER TO NEXT SSCVT
         B     GETSSCVT               AND CONTINUE TO SCAN FOR SUBSYS
         SPACE 1
GOTSSCVT ICM   R2,15,SSCTSSVT      IF NO SSVT THEN
         BZ    CANESTAE             DO NOT FREE SSVT AND COMMAND TABLE.
         XC    SSCTSSVT,SSCTSSVT   DE-ACTIVATE SUBSYSTEM.
         DROP  R1
         ICM   R3,15,SSVTANKR-SSVT(R2)   IF NO COMMAND TABLE THEN
         BZ    FREESSVT                   DO NOT UNALLOCATE IT.
         SPACE
         LR    R6,R2               PRESERVE POINTER TO SSVT
         L     RØ,TABLSP           LOAD TABLE LENGTH AND
         LR    R1,R3                ADDRESS OF TABLE AND
         LA    R2,245                SUBPOOL OF TABLE AREA AND
         BAS   R1Ø,CPSTORF         THEN FREE IT
         SPACE 1
FREESSVT L     RØ,SSVTSP           LOAD LENGTH OF SSVT AND
         LR    R1,R6                ITS ADDRESS AND
         LA    R2,245                ITS SUBPOOL AND
         BAS   R1Ø,CPSTORF         THEN FREE IT
         SPACE 1
CANESTAE ESTAE Ø                   CANCEL ERROR RECOVERY
         SPACE 1
         MODESET KEY=NZERO,MODE=PROB RETURN TO BEING A PLEBIAN
```

```
         EJECT
CPRETURN LR    R1,R13              FREEMAIN ADDRESS
         L     R13,4(,R1)          CALLING REGISTER SAVE AREA
         XC    8(4,R13),8(R13)     UNCHAIN SAVE AREAS
         SPACE 1
         LA    RØ,72               SIZE OF SAVE AREA
         SR    R2,R2               SET SUBPOOL
         BAS   R1Ø,CPSTORF         FREE VIRTUAL SAVE AREA
         SPACE 1
         RETURN (14,12),RC=Ø       TERMINATE NORMALLY
         DROP R6
         TITLE 'DCIPLS    COMMON CODE'
PAT2LONG WTO   'DCIPL1ØA  BROADCAST MESSAGE WAS TOO LONG: TRY AGAIN'
         B     DCNOWAIT            AWAIT RETRY
         SPACE 1
PATSVC34 SR    RØ,RØ               CLEAR CONSOLE IDENTIFICATION
         STH   RØ,WTOMSG+2         CLEAR RIGHT HALF
         LA    R1,WTOMSG           POINT TO OPERATOR COMMAND
         SVC   34                  BROADCAST WARNING
         SPACE 1
         BR    R1Ø                 DEPART
         SPACE 2
*********************************************************************
*        SLEEP                                                      *
*********************************************************************
         SPACE
PATREST  STIMER WAIT,BINTVL=CLAMTIME SLEEP...
         SPACE 1
         BR    R1Ø                 ALIVE - AT LAST≥
         SPACE 2
PATMOVE  MVC   COMMNDWK+2Ø(*-*),Ø(R1) ***  E X E C U T E  O N L Y  ***
         EJECT
*********************************************************************
*        ACQUIRE VIRTUAL STORAGE IN THE SUBPOOL SPECIFIED           *
*********************************************************************
         SPACE
CPSTORA  STORAGE OBTAIN,LENGTH=(Ø),SP=(2) ACQUIRE SUBPOOL STORAGE
         SPACE 1
         BR    R1Ø                 RETURN TO CALLER
         SPACE 1
*********************************************************************
*        FREE VIRTUAL STORAGE FROM THE SUBPOOL SPECIFIED            *
*********************************************************************
         SPACE
CPSTORF  STORAGE RELEASE,LENGTH=(Ø),ADDR=(1),SP=(2) FREE SUBPOOL STORGE
         SPACE 1
         BR    R1Ø                 RETURN TO CALLER
         EJECT
*********************************************************************
*        SERIALIZE USE OF THE COMMAND-SCHEDULING-CONTROL-BLOCK CHAIN *
*********************************************************************
         SPACE 1
```

```
*        ENQ   (CMR1,CMR2,E,3,SYSTEM),RET=NONE,RNL=NO,SMC=STEP
CMRENQ   ENQ   (CMR1,CMR2,E,3,SYSTEM),RET=NONE,RNL=NO SMC=STEP
         BR    R1Ø                 RETURN TO CALLER
         SPACE 1
*        DEQ   (CMR1,CMR2,3,SYSTEM),RET=NONE,RNL=NO,SMC=STEP
CMRDEQ   DEQ   (CMR1,CMR2,3,SYSTEM),RET=NONE,RNL=NO SMC=STEP
         BR    R1Ø                 RUECKSPRUNG
         SPACE 1
CMR1     DC    CL8'SYSIEFSD'
CMR2     DC    CL3'Q1Ø'
         TITLE  ' CONTANTS AND WORKAREAS.'
*********************************************************************
*        THE FOLLOWING COMMANDS ARE USED TO TERMINATE MISCELLANEOUS  *
*        PROBLEM APPLICATIONS WHEN PAP COMMAND IS ISSUED             *
*********************************************************************
         SPACE 1
MISCTAB  DS    ØF
HSMA     DC    CL2Ø'F HSMA,STOP'
HSMB     DC    CL2Ø'F HSMB,STOP'
HSMC     DC    CL2Ø'F HSMC,STOP'
HSMD     DC    CL2Ø'F HSMD,STOP'
HSMYMM   DC    CL2Ø'F HSMY2K,STOP'
SILOS    DC    CL2Ø'P SILO'
PPGPLLA  DC    CL2Ø'P LLA '
PPGPVLF  DC    CL3'VLF'
PPGPTM   DC    CL14'P TM '
         SPACE 1
*********************************************************************
*        THIS IS THE COMMAND TO TERMINATE TELECOMMUNICATIONS         *
*********************************************************************
         SPACE 1
PATCANCL DC    H'16',H'Ø',CL12'Z NET,CANCEL'
         EJECT
ENTNUM   EQU   3                   NUMBER OF CELLS AVAILABLE
         SPACE 1
ENTLEN   EQU   1Ø8                 SIZE OF A CELL IN COMMAND TABLE
         SPACE 1
TABPRE   EQU   12                  LENGTH OF COMMAND TABLE'S PREFIX
         SPACE 1
TABLEN   EQU   ENTNUM*ENTLEN+TABPRE SIZE OF COMMAND TABLE
         SPACE 1
TABLSP   DC    A(TABLEN)
         SPACE 1
WORKSP   DC    A(WORKLEN)
         SPACE 1
SSVTSP   DC    A(SSVTLEN)
         SPACE 1
SSCTSP   DC    A(SSCTSIZE)
         SPACE 1
PATSSCVT DC    C'SSCT'
         SPACE 1
PATNAME  DC    CL8'DCIPLES '
```

```
         SPACE 1
PATA31   DC    A(PAT31+X'80000000')
         SPACE 1
CLAMTIMX DC    F'100'              100 = ONE SECOND
         SPACE 1
CLAMTIME DC    F'600'              100 = ONE SECOND
         SPACE 1
BLANK    DC    CL1' '
         SPACE 3
PPGERMSG DC    CL43'DCIPL21E '
         ORG   PPGERMSG+9
         DC    C'RETURN CODE = '
PPGRETC  DC    CL3' '
         DC    C'REASON CODE = '
PPGREAC  DC    CL3' '
         ORG
         SPACE
WTOMSG1  DC    CL43'DCIPL11I - THIS PRODUCT IS NOT UP - XXXXXXXX'
         SPACE 1
WTOMSG2  DC    CL42'DCIPL12I - SYSTEM PROGRAMMER CHKEY XXXXXXXX'
         SPACE 1
WTOMSG3  DC    CL46'DCIPL13I - THIS PRODUCT IS NOT DOWN - XXXXXXXX'
         SPACE 1
ASIDPAT  DC    X'F0F0F0F0F0'
         SPACE 1
HEXTRAN  DC    C'0123456789ABCDEF'  TRANSLATE TABLE
         SPACE 1
PATONE   DC    F'1'
         DC    F'1'
         DC    F'1'
         SPACE 1
DCIP12   DC    H'12'
         SPACE 2
         DS    0F
PPHCASID DS    F
PPHONE   DC    F'1'
PPHZERO  DC    F'0'
PPHJNAME DC    CL8'NET'
ATCCONFT EQU   X'440'
CONVTHAA EQU   X'94'
RDTFORW  EQU   X'70'
RDTPLEN  EQU   X'68'
RPRNAME  EQU   0
RPRCURS1 EQU   X'3C'
RDTPRIRN EQU   X'08'
FSMCATAC EQU   X'05'
         SPACE 2
PATDCB   DCB   DSORG=PO,MACRF=R,EODAD=PATOPER,RECFM=FB,DDNAME=CURLIST
         SPACE 1
CMDDCB   DCB   DSORG=PO,MACRF=R,EODAD=CMDEND2,RECFM=FB,DDNAME=COMMAND
         SPACE 1
CMDECB   DC    F'0'
```

```
PATECB   DC    F'Ø'
         SPACE 1
ATCSTAT1 EQU   X'488'
         SPACE 1
ATCACTIV EQU   X'4Ø'
         EJECT
*************************************************************************
*        THE MEMBER NAMED CURLIST THAT RESIDES IN THE DATASET NAMED     *
*        NCP25.AGØ3ZGEN.NCPA CONTAINS THE VALUE FOR THE LIST OPTION     *
*        ON A VTAM START COMMAND; IT MUST BE A1 OR A4.                  *
*************************************************************************
         SPACE 1
A1       DC    CL3'A1'
A4       DC    CL3'A4'
         SPACE 2
         CNOP  2,4                  ALIGN LIST
LISTADDR DC    X'ØØØ1',AL2(12),CL8'CURLIST',4C' '
         SPACE 1
BLOCKADR EQU   LISTADDR+12
         SPACE 3
PATSNET  DC    H'22',H'Ø',CL18'S NET,,,(LIST=PG)'
         SPACE 1
PATSLEN  EQU   *-PATSNET
         SPACE 1
PATSARG  EQU   PATSLEN-4
         EJECT
*************************************************************************
*        TABLE OF MEMBERS IN NCP25.AGØ3ZGEN.NCPA THAT CONTAIN           *
*        OPERATOR COMMANDS, IDENTIFIERS OF TASKS THAT ARE TO BE         *
*        CONTROLLED, AND RESPONSES TO OUTSTANDING WTOR'S.               *
*************************************************************************
         SPACE 1
         CNOP  2,4                  ALIGN LIST
CMDDIRØ1 DC    X'ØØØ1',AL2(12),CL8'TECUPNET',4C' '
         SPACE 1
CMDADRØ1 EQU   CMDDIRØ1+12
         SPACE 1
         CNOP  2,4                  ALIGN LIST
CMDDIRØ2 DC    X'ØØØ1',AL2(12),CL8'TECDWNET',4C' '
         SPACE 1
CMDADRØ2 EQU   CMDDIRØ2+12
         SPACE 1
         CNOP  2,4                  ALIGN LIST
CMDDIRØ4 DC    X'ØØØ1',AL2(12),CL8'DEVUPNET',4C' '
         SPACE 1
CMDADRØ4 EQU   CMDDIRØ4+12
         SPACE 1
         CNOP  2,4                  ALIGN LIST
CMDDIRØ5 DC    X'ØØØ1',AL2(12),CL8'DEVDWNET',4C' '
         SPACE 1
CMDADRØ5 EQU   CMDDIRØ5+12
         SPACE 1
```

```
         CNOP  2,4                   ALIGN LIST
CMDDIRØ7 DC    X'ØØØ1',AL2(12),CL8'ACCUPNET',4C' '
         SPACE 1
CMDADRØ7 EQU   CMDDIRØ7+12
         SPACE 1
         CNOP  2,4                   ALIGN LIST
CMDDIRØ8 DC    X'ØØØ1',AL2(12),CL8'ACCDWNET',4C' '
         SPACE 1
CMDADRØ8 EQU   CMDDIRØ8+12
         SPACE 1
         CNOP  2,4                   ALIGN LIST
CMDDIR1Ø DC    X'ØØØ1',AL2(12),CL8'PROUPNET',4C' '
         SPACE 1
CMDADR1Ø EQU   CMDDIR1Ø+12
         SPACE 1
         CNOP  2,4                   ALIGN LIST
CMDDIR11 DC    X'ØØØ1',AL2(12),CL8'PRODWNET',4C' '
         SPACE 1
CMDADR11 EQU   CMDDIR11+12
         SPACE 1
         CNOP  2,4                   ALIGN LIST
CMDDIR13 DC    X'ØØØ1',AL2(12),CL8'DCIPWARN',4C' '
         SPACE 1
CMDADR13 EQU   CMDDIR13+12
         EJECT
         CNOP  2,4                   ALIGN LIST
CMDDIR14 DC    X'ØØØ1',AL2(12),CL8'PAPRESOR',4C' '
         SPACE 1
CMDADR14 EQU   CMDDIR14+12
         SPACE 1
         CNOP  2,4                   ALIGN LIST
CMDDIR15 DC    X'ØØØ1',AL2(12),CL8'PAPMODIF',4C' '
         SPACE 1
CMDADR15 EQU   CMDDIR15+12
         SPACE 1
         CNOP  2,4                   ALIGN LIST
CMDDIR16 DC    X'ØØØ1',AL2(12),CL8'PAPPCANC',4C' '
         SPACE 1
CMDADR16 EQU   CMDDIR16+12
         SPACE 1
         CNOP  2,4                   ALIGN LIST
CMDDIR17 DC    X'ØØØ1',AL2(12),CL8'PAPWTOR2',4C' '
         SPACE 1
CMDADR17 EQU   CMDDIR17+12
         SPACE 1
         CNOP  2,4                   ALIGN LIST
CMDDIR18 DC    X'ØØØ1',AL2(12),CL8'PAPWTOR1',4C' '
         SPACE 1
CMDADR18 EQU   CMDDIR18+12
         SPACE 1
         CNOP  2,4                   ALIGN LIST
CMDDIR19 DC    X'ØØØ1',AL2(12),CL8'NETWTOR1',4C' '
```

```
         SPACE 1
CMDADR19 EQU   CMDDIR19+12
         SPACE 1
         CNOP  2,4                  ALIGN LIST
CMDDIR2Ø DC    X'ØØØ1',AL2(12),CL8'NETWTOR2',4C' '
         SPACE 1
CMDADR2Ø EQU   CMDDIR2Ø+12
         SPACE 1
CMDDIR21 DC    X'ØØØ1',AL2(12),CL8'NETWTOR3',4C' '
         SPACE 1
CMDADR21 EQU   CMDDIR21+12
         SPACE 1
CMDDIR22 DC    X'ØØØ1',AL2(12),CL8'CONNEDEV',4C' '
         SPACE 1
CMDADR22 EQU   CMDDIR22+12
         SPACE 1
CMDDIR23 DC    X'ØØØ1',AL2(12),CL8'CONNEACC',4C' '
         SPACE 1
CMDADR23 EQU   CMDDIR23+12
         SPACE 1
CMDDIR24 DC    X'ØØØ1',AL2(12),CL8'CONNEPRO',4C' '
         SPACE 1
CMDADR24 EQU   CMDDIR24+12
         EJECT
CMDDIR25 DC    X'ØØØ1',AL2(12),CL8'CONWTORS',4C' '
         SPACE 1
CMDADR25 EQU   CMDDIR25+12
         SPACE 1
CMDDIR26 DC    X'ØØØ1',AL2(12),CL8'CONWTORA',4C' '
         SPACE 1
CMDADR26 EQU   CMDDIR26+12
         SPACE 1
CMDDIR27 DC    X'ØØØ1',AL2(12),CL8'CONWTORD',4C' '
         SPACE 1
CMDADR27 EQU   CMDDIR27+12
         SPACE 1
CMDDIR28 DC    X'ØØØ1',AL2(12),CL8'CONWTORP',4C' '
         SPACE 1
CMDADR28 EQU   CMDDIR28+12
         SPACE 1
CMDDIR29 DC    X'ØØØ1',AL2(12),CL8'YMMUPNET',4C' '
         SPACE 1
CMDADR29 EQU   CMDDIR29+12
         SPACE 1
CMDDIR3Ø DC    X'ØØØ1',AL2(12),CL8'YMMDWNET',4C' '
         SPACE 1
CMDADR3Ø EQU   CMDDIR3Ø+12
         SPACE 1
CMDDIR31 DC    X'ØØØ1',AL2(12),CL8'NETWTOR2',4C' '
         SPACE 1
CMDADR31 EQU   CMDDIR31+12
```

```
         SPACE 1
CMDDIR32 DC    X'0001',AL2(12),CL8'CONNEYMM',4C' '
         SPACE 1
CMDADR32 EQU   CMDDIR32+12
         SPACE 1
CMDDIR33 DC    X'0001',AL2(12),CL8'CONWTORY',4C' '
         SPACE 1
CMDADR33 EQU   CMDDIR33+12
         EJECT
         LTORG
         TITLE  ' ERROR RECOVERY TO DE-ACTIVATE SUBSYSTEM (OPTIONAL), AX
               ND RESET COMMAND TABLE LOCK WORD.'
*********************************************************************
*        GENERALIZED PROGRAM RECOVERY ROUTINE                       *
*                                                                   *
*        PERFORMS THE ERROR RECOVERY PROCESSING FOR ANY ROUTINE.    *
*        THIS ROUTINE IS INVOKED VIA THE ESTAE ERROR RECOVERY       *
*        MECHANISM IN THE EVENT OF AN ABEND CONDITION IN A ROUTINE. *
*        THE PURPOSE OF THIS RECOVERY ROUTINE IS TO IDENTIFY THE OFF- *
*        SET OF AN ABEND WITHIN A CSECT THAT ISSUED THE CORRESPONDING *
*        ESTAE AND, WHEN REQUESTED, ATTEMPT TO RECOVER AT A         *
*        RETRY ADDRESS IN PARAMETERS PASSED TO THIS EXIT IN THE SDWA. *
*        IF NO PARAMETERS WERE SPECIFIED WHEN A RECOVERY ENVIRONMENT *
*        WAS ESTABLISHED, PERCOLATION CONTINUES WITHOUT AN ATTEMPT   *
*        TO RECOVER FROM ABEND CONDITIONS.                          *
*                                                                   *
*        REG       ENTRY VALUE                                      *
*                                                                   *
*        R0        CODE EXPLAINING TYPE I/O PROCESSING PERFORMED    *
*                                                                   *
*                  0    - I/O QUIESCED AND RESTORABLE               *
*                                                                   *
*                  4    - I/O HALTED AND NOT RESTORABLE             *
*                                                                   *
*                  8    - NO I/O AT TIME OF ABEND                   *
*                                                                   *
*                  12   - SDWA STORAGE UNAVAILABLE                  *
*                         R1  -  ABEND COMPLETION CODE              *
*                         R2  -  ADDR OF ESTAE PARM LIST OR 0       *
*                                                                   *
*                  16   - NO I/O PROCESSING PERFORMED               *
*                                                                   *
*                                                                   *
*        R1        POINTER TO SDWA                                  *
*                  FIRST WORD POINTS TO ESTAE PARM LIST             *
*                                                                   *
*        R13       POINTER TO 72-BYTE SAVE AREA ADDRESS             *
*                                                                   *
*        R14       RETURN ADDRESS                                   *
*                                                                   *
*        R15       ENTRY ADDRESS                                    *
*********************************************************************
```

```
         EJECT
         DROP  R12,R7,R11          FORGET ORIGINAL BASE REGISTERS
         DS    ØF                  ALIGN RECOVERY RTN ON INTEGRAL BNDRY
         USING PSA,RØ              ESTABLISH PSA ADDRESSABILITY
         USING *,R15
         SPACE 1
PATEXIT  DS    ØH                  ESTAE ERROR RECOVERY
         SPACE 1
         SAC   Ø                   ENSURE OPERATING IN PRIMARY MODE
         L     R2,CVTPTR           FETCH POINTER TO ESA'S CVT
         USING CVTMAP,R2           ESTABLISH CVT ADDRESSABILITY
         SPACE 1
         L     R2,CVTJESCT         FETCH POINTER TO THE JESCT
         USING JESCT,R2            ESTABLISH JESCT ADDRESSABILITY
         SPACE 1
         ICM   R6,15,JESSSCT       FETCH POINTER TO THE SSCVT
         USING SSCT,R6             ESTABLISH SSCVT ADDRESSABILITY
         SPACE 1
PATERXIT BZ    PATSABLE            AT END, DEPART
         CLC   SSCTSNAM,=C'DCIPLES' IF CORRECT SUBSYSTEM NAME,
         BE    PATERRTN              THEN PROCESS IT
         ICM   R6,15,SSCTSCTA         ELSE OBTAIN POINTER TO NEXT SSCVT
         B     PATERXIT                AND CONTINUE TO SEARCH
         SPACE 1
PATERRTN ICM   R4,15,SSCTSSVT      OBTAIN ADDRESS OF SUBSYSTEM'S SSVT
         BZ    PATSABLE            BRANCH IF NONEXISTENT
         XC    SSCTSSVT,SSCTSSVT   DE-ACTIVATE THE SUBSYSTEM(OPTIONAL)
         DROP  R2
         XC    SSVTASCB-SSVT(L'SSVTASCB,R4),SSVTASCB-SSVT(R4)
         ICM   R4,15,SSVTANKR-SSVT(R4) TEST IF TABLE AREA OBTAINED
         BZ    PATSABLE            BRANCH IF NOT
         XC    8(4,R4),8(R4)       RESET LOCK-WORD IN COMMAND TABLE
         DROP  R6                  FORGET SSCVT
         EJECT
PATSABLE C     RØ,=F'12'           TEST IF AN SDWA IS AVAILABLE
         BE    Ø(R14)              IF NOT, RETURN TO CONTINUE WITH DUMP
         SPACE 1
         ST    R14,12(R13)         SAVE RETURN ADDRESS
         L     R14,Ø(R1)           RETRY ADDRESS
         TM    Ø(R14),X'8Ø'        TEST IF RE-ENTRY
         L     R14,12(R13)         RESTORE RETURN ADDRESS
         BO    PATSTAE             BRANCH IF NOT
         SVC   3                   BACK TO DUST
         SPACE 1
PATSTAE  BAKR  R14,RØ              SAVE ENVIRONMENT AT ENTRY
         DROP  R15                 FORGET TEMPORARY BASE
         SPACE 1
         LR    R12,R15             PRIME BASE REGISTER
         USING PATEXIT,R12         ESTABLISH  ADDRESSABILITY
         SPACE 1
         USING SDWA,R5             ESTABLISH SDWA ADDRESSABILITY
         LR    R5,R1               ADDRESS OF SDWA
```

```
         SPACE 1
         MVC   PATCNAME,SDWANAME   SDWA
         MVC   PATSDWA,Ø(R1)       SDWA
         SPACE 1
         L     RØ,PATESIZE         SIZE OF WORK AREA
         STORAGE OBTAIN,LENGTH=(Ø) OBTAIN WORK AREA
         LR    R2,R1               POINT TO IT
         EJECT
         USING PATDSECT,R2         ESTABLISH BASE
         MVC   PATGPRS(4*16),SDWAGRSV REGISTERS AT TIME OF ABEND
         L     R9,SDWASR13         POINT TO SAVE AREA AT TIME OF ERROR
         MVC   PATEWTO(PATWTOL),PATWTO INITIALIZE WORK AREA
         SPACE
         UNPK  PATDLPSW(9),SDWAEC1(5)  LEFT HALF OF EC PSW
         MVI   PATDLPSW+8,C' '
         TR    PATDLPSW,PAWzANS-24Ø
         SPACE
         UNPK  PATDRPSW(9),SDWANXT1(5) RIGHT HALF OF EC PSW
         MVI   PATDRPSW+8,C' '
         TR    PATDRPSW,PATRANS-24Ø
         SPACE 1
         L     RØ,SDWANXT1         TERMINATION ADDRESS
         N     RØ,PAT7FFF          CLEAR 31-AMODE BIT
         ST    RØ,PATGILL          SAVE INSTRUCTION ADDRESS
         L     R4,PSATOLD          CURRENT TCB ADDRESS
         BAS   R8,PATMLOC          LOCATE FAILING MODULE
         SPACE 3
         L     R4,PSATOLD          CURRENT TCB ADDRESS
         USING TCB,R4              ESTABLISH TCB ADDRESSABILITY
         L     R8,TCBRBP           POINT TO AN RB
         DROP  R4                  FORGET TCB
         USING RBBASIC,R8          ESTABLISH RB ADDRESSABILITY
PATNXTRB N     R8,PATØFFF          CLEAR DE TRASH
         LR    R11,R8              PRESERVE ADDRESS OF RB
         L     R8,RBLINK           FETCH ADDRESS OF NEXT RB
         CR    R4,R8               TEST IF MISSING RB
         BNE   PATNXTRB            BRANCH IF NOT
         LR    R8,R11              SET ADDRESS OF RB
         SPACE 1
         L     RØ,RBOPSWA          FETCH RIGHT HALF OF PSW
         N     RØ,PAT7FFF          CLEAR 31-AMODE BIT
         ST    RØ,PATGILL          USE IT AS FAILING INSTRUCTION ADDRES
         BAS   R8,PATMLOC          LOCATE FAILING MODULE
         SPACE 1
         LR    R8,R11              RESTORE ADDRESS OF RB
         L     R3,RBCDE            POINT TO CONTENTS DIRECTORY ENTRY
         N     R3,PATØFFF          CLEAR OPTIONS
         B     PHCDNAME            ENTER COMMON CODE
         DROP  R8                  FORGET RB
         EJECT
*********************************************************************
*        LOCATE MODULE CONTAINING FAILING INSTRUCTION               *
```

```
*********************************************************************
         SPACE 1
PATMLOC  L     R4,PSATOLD          CURRENT TCB ADDRESS
         USING TCB,R4              ESTABLISH TCB ADDRESSABILITY
         SPACE 1
         L     R3,TCBJPQ           ADDR OF LAST CDE IN JOB PACK AREA Q
         SPACE 1
         MVC   PATDPGM,=CL8'UNKNOWN' SET CONSTANT IN WTO AREA
         MVC   PATDEP,=CL8'UNKNOWN' SET CONSTANT IN WTO AREA
         MVC   PATDOFF,=CL8'UNKNOWN' SET CONSTANT IN WTO AREA
         USING CDENTRY,R3          ESTABLISH CDE ADDRESSABILITY
         TM    CDATTR,CDMIN        TEST IF MINOR CDE
         BO    PGNXTCDE            BRANCH IF NOT
         SPACE 1
PATGETXL L     R7,CDXLMJP          ADDR OF EXTENT LIST OF THIS MODULE
         USING XTLST,R7            ESTABLISH XTLST ADDRESSABILITY
         CLC   PATGILL,XTLMSBAD    COULD BLOCK CONTAIN FAILING INST?
         BL    PGNXTCDE            BRANCH IF NOT
         SPACE 1
         L     RØ,PATGILL          TERMINATION ADDRESS
         S     RØ,XTLMSBAD         COMPUTE OFFSET INTO BLOCK
         CLM   RØ,7,XTLMSBLN       TEST IF OFFSET WITHIN BLOCK
         BL    PATOK               BRANCH IF SO
         SPACE 1
PGNXTCDE ICM   R3,15,CDCHAIN       NEXT CDE ON CHAIN
         BE    Ø(R8)               AT END RETURN
         TM    CDATTR,CDMIN        TEST IF MINOR CDE
         BNO   PATGETXL            BRANCH IF NOT
         B     PGNXTCDE            ELSE TRY TRY AGAIN
         SPACE 1
PATOK    ST    RØ,PATGILL          SAVE OFFSET
         MVI   PATGILL,Ø           CLEAR TRASH
         UNPK  PATDOFF(9),PATGILL(5)
         MVI   PATDOFF+8,C' '
         TR    PATDOFF,PATRANS-24Ø
PHCDNAME MVC   PATDPGM,CDNAME      MODULE ACTIVE AT TIME OF ABEND
         SPACE 1
         L     R15,CDENTPT         FETCH ADDRESS OF ENTRY POINT
         CLC   Ø(3,R15),=XL3'47FØFØ' TEST IF MODULE CONFORMS
         BNE   PATRADIC            BRANCH IF A RADICAL
         CLI   4(R15),Ø            TEST IF LENGTH IS ZERO
         BE    PATRADIC            BRANCH IF SO
         MVC   PATDEP,=CL8' '      BLANK AREA
         SR    R14,R14             CLEAR WORK REGISTER
         IC    R14,4(R15)          OBTAIN LENGTH OF ENTRY POINT NAME
         CH    R14,=H'9'           TEST IF NAME EXCEEDS EIGHT BYTES
         BL    *+8                 BRANCH IF NOT
         LA    R14,8               LIMIT LENGTH TO EIGHT BYTES
         BCTR  R14,RØ              REDUCE BY ONE FOR MOVE
         EX    R14,PATMVEP         MOVE NAME OF ENTRY POINT TO WTO AREA
         EJECT
PATRADIC CLI   SDWACMPC+2,X'ØØ'    TEST IF USER ABEND
```

```
         BNE   PATDOUSR            BRANCH IF SO
         TM    SDWACMPC+1,X'ØF'    TEST IF USER ABEND
         BNZ   PATDOUSR            BRANCH IF SO
         UNPK  PATDCODE+1(5),SDWACMPC(3) ASSUME SYSTEM ABEND
         MVI   PATDCODE+4,C' '
         MVI   PATDCODE+5,C' '
         TR    PATDCODE+1(3),PATRANS-24Ø
         B     PATXWTO
         SPACE 1
PATDOUSR L     R1,SDWACMPF         FETCH USER ABEND CODE
         N     R1,PATØØ3F          CLEAR HI-ORDER TRASH
         CVD   R1,PPHTWICE         THENCE TO PACKED DECIMAL
         UNPK  PATDCODE,PPHTWICE+5(3) ASSUME USER ABEND
         MVI   PATDCODE,C'U'       SHOW USER ABEND
         OI    PATDCODE+4,24Ø      FILL FINAL CHARACTER FOR TRANSLATION
         MVI   PATDCODE+5,C' '     REMOVE GARBAGE
         TR    PATDCODE+1(4),PATRANS-24Ø EVERYTHING TO UPER CASE EBCDIC
         SPACE 1
PATXWTO  WTO   MF=(E,PATEWTO)
         EJECT
*********************************************************************
*        FORMAT GENERAL PURPOSE REGISTERS AT ENTRY TO ABEND         *
*********************************************************************
         SPACE 1
         MVI   PATEWTO+4,C' '      REMOVE TRASH
         MVC   PATEWTO+5(PATWTOL-5),PATEWTO+4 FROM WORK AREA
         LA    R3,PATGPRC-9        POINT TO CONSTANTS - 9
         LA    R7,SDWAGRSV         POINT TO REGISTERS AT ENTRY TO ABEND
         LA    R8,4                SET LOOP COUNT
PATPGPRS LA    R3,9(R3)            NEXT CONSTANT
         MVC   PATEWTO+4(9),Ø(R3)  CONSTANT TO WTO AREA
         LA    R4,4                SET LOOP COUNT
         LA    R6,PATEWTO+14       POINT TO REGISTER AREA
PATUNPKG UNPK  Ø(9,R6),Ø(5,R7)     CONVERT REGISTER CONTENTS
         TR    Ø(8,R6),PATRANS-24Ø  TO EBCDIC
         MVI   8(R6),C' '          CLEAR TRASH FROM WTO
         LA    R6,1Ø(R6)           NEXT OUTPUT LOCATION
         LA    R7,4(R7)            NEXT GENERAL PURPOSE REGISTER
         BCT   R4,PATUNPKG         LOOP POWER
         WTO   MF=(E,PATEWTO)      DISPLAY FORMATTED DATA
         BCT   R8,PATPGPRS         DISPLAY GENERAL PURPOSE REGISTERS
         SPACE 1
         L     RØ,PATESIZE         SIZE OF AREA TO BE RELEASED
         SPACE 1
         STORAGE RELEASE,ADDR=(2),LENGTH=(Ø)
         SPACE 1
         ICM   R6,15,SDWAPARM      ADDRESS OF ESTAE PARAMETER LIST
         BE    PATABEND            IF NONE, CONTINUE WITH TERMINATION
         NI    Ø(R6),255-X'8Ø'     CLEAR RETRY INDICATOR
         ICM   R6,15,Ø(R6)         RETRY ADDRESS
         BE    PATABEND            IF NONE, CONTINUE WITH TERMINATION
         EJECT
```

```
         SETRP WKAREA=(R5),RETADDR=(R6),RC=4  RETRY ONCE
         SPACE 1
         SR    R15,R15             SET RETURN CODE
         PR    R14                 RETURN TO DUST
         SPACE 2
PATABEND SETRP WKAREA=(R5),RC=Ø    ABEND
         SPACE 1
         SR    R15,R15             SET RETURN CODE
         PR    R14                 RETURN TO DUST
         EJECT
*********************************************************************
*        CONSTANTS AND EQUATES                                      *
*********************************************************************
         SPACE 1
PPHTWICE DS    D
PAT7FFF  DC    X'7FFFFFFF'
PATØFFF  DC    X'ØØFFFFFF'
PATØØ3F  DC    X'ØØØØØFFF'
PATMVEP  MVC   PATDEP(*-*),5(R15)  **** EXECUTE ONLY ****
PATSDWA  DC    XL256'ØØ'
         DC  CL8'JOHNSLUV'
PATCNAME DC  CL8' '
         SPACE 1
PATSIZE  DC    A(18*4)
PATESIZE DC    A(PATDSIZE)
PATWTO   WTO  ' ABEND STAP  IN PGM LLEWWOP  AT EP HP       OFFSET PATTYL
               H   PSW - SNIKWAH   ENNA        ',MF=L
PATWTOL  EQU  *-PATWTO
PATRANS  DC    C'Ø123456789ABCDEF'
         SPACE 3
PATLIST  DC    X'8Ø',AL3(Ø) <===== PLACE ADDRESS OF RETRY ROUTINE HERE
         SPACE
PATGPRC  DC    CL9'GPR  Ø-3'
         DC    CL9'GPR  4-7'
         DC    CL9'GPR  8-11'
         DC    CL9'GPR 12-15'
         SPACE 2
         LTORG
         EJECT
PATDSECT DSECT
PATGILL  DS    2F
PATGPRS  DS    16F
PATEWTO  DS    (PATWTOL)C
         ORG   PATEWTO+4
         DC    C' ABEND '
PATDCODE DC    CL5'UØØØØ'
         DC    C' IN PGM '
PATDPGM  DC    CL8' '
         DC    C' AT EP '
PATDEP   DC    CL8' '
         DC    C' OFFSET '
PATDOFF  DC    CL8' '
```

```
         DC    C' PSW - '
PATDLPSW DC    CL8' '
         DC    C'  '
PATDRPSW DC    CL8' '
         DC    C' '
         ORG
PATDSIZE EQU   ((((*-PATGILL)+7)/8)*8)
         TITLE 'FORMAT OF AN ENTRY IN THE COMMAND TABLE'
WORK     DSECT ,                  WTO BUFFER.
WTOMSG   DS    CL4                LENGTH AND MCS FLAGS.
COMMNDWK DS    CL(ENTLEN)         MESSAGE TEXT.
         ORG   COMMNDWK
         DS    F                  +Ø
CMDCONID DS    F                  +4  CONSOLE IDENTIFIER
CMDTEXT  DS    CL8                +8  TEXT OF COMMAND
         DS    CL34               +16 MULTI-PURPOSE AREA
CMDSYSID DS    CL4                +5Ø SYSTEM IDENTIFIER
CMDFAILI DS    CL1                +54 FAILURE INDICATOR
         DS    CL9                +55 FILLER
CMDA1STC DS    CL4                +64 ADDRESS OF FIRST COMMAND
CMDPARM  DS    ØCL16              +68 PARAMETER FOR BLDL
         SPACE
CMDBLDLN DS    H                  +68 NUMBER OF ENTRIES IN BLDL LIST
CMDBLDLL DS    H                  +7Ø LENGTH OF AN ENTRY IN BLDL LIST
CMDBLDLM DS    CL8                +72 NAME OF MEMBER IN PDS
CMDBLDLT DS    CL4                +8Ø TTRØ OF MEMBER IN PDS
         SPACE
CMDR1SAV DS    CL4                +84 SAVE AREA FOR R1
CMDR2SAV DS    CL4                +88 SAVE AREA FOR R2
CMDBASAV DS    CL4                +92 SAVE AREA FOR BAS
         ORG
         DS    CL4                DESCRIPTOR AND ROUTING CODES
FLAG     DS    CL4                MESSAGE TYPE
SYSID    DS    CL4
         DS    ØD
         SPACE 1
WTORECB  DS    F
CLAMHOLD DS    F
PPGCOUNT DS    F
PPGWORK  DS    XL1ØØ
PPGUCB   DS    XL48
WORKLENH EQU   *-WORK
         SPACE 1
```

Editor’s note: this article will be continued in the next issue.

NEON Systems has launched its Shadow Enterprise Transactions family aimed at increasing integrity, flexibility, and utility of e-commerce applications on S/390 systems. It consists of elements already supported in the Shadow products, including support for a variety of two-phase commit protocols found in the XA, Microsoft, and BEA execution environments, with support for IBM's OS/390 Recoverable Resource Management Services (RRMS). Now, CICS Transaction Server and virtually all OS/390 data and transactional sources can participate in complex transactions with mainframe-level integrity.

A Shadow product on OS/390 implements Shadow Enterprise Transactions through integration with RRMS, enabling transactional access to resources on and outside OS/390. Integration with RRMS allows applications to access multiple protected resources residing on OS/390 or on other platforms in a single enterprise transaction, in which all changes to all resources will be completed or rolled back.

For further information contact:
Neon Systems Inc, 14141 Southwest Freeway, Suite 6200, Sugar Land, TX 77478, USA.
Tel: (281) 491 4200
Fax: (281) 242 3880 or
Neon Systems UK Ltd, Third Floor, Sovereign House, 26-30 London Road, Twickenham, Middlesex, TW1 3RW, UK.
Tel: (0181) 607 9911
Fax: (0181) 607 9933.
http://www.neonsys.com

* * *

Tivoli has released Version 3.6.1 of its framework for OS/390, along with related Security Management, Software Distribution, Inventory, Distributed Monitoring, and User Administration tools. New capabilities include distribution of software from an OS/390 to the entire enterprise, inventory of hardware and software information across the enterprise, and storage of systems management data in an OS/390 relational database.

For further information contact:
Tivoli Systems, 9442 Capital of Texas Highway, North Austin, TX 78759, USA.
Tel: 512 436 8000
Fax; 512 794 0623
Tivoli Systems, Sefton Park, Bells Hills, Buckinghamshire, SL2 4HD, UK.
Tel: 01753 896 896
Fax: 01753 896 899
http://www.tivoli.com

* * *

Software AG has rolled out Version 6.3 of its Adaplex+ tool for mainframe-based Adabas databases under MVS/ESA. For the first time, it supports Parallel Sysplex-based continuous availability of OLTP applications, regardless of which computer in the Sysplex network fails.

For further information contact:
Software AG, 11190 Sunrise Valley Drive, Reston, VA 2201, USA.
Tel: (703) 860 5050
Fax: (703) 391 6975
Software AG (UK) Ltd, Charter Court, 74-78 Victoria Street, St Albans, AL1 3XH, UK.
Tel: 01727 84445
Fax: 01727 840092
http://www.sofwareag.com

* * *