143 MVS update

August 1998

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 £325.00 in the UK; $485.00 in the USA and Canada; £331.00 in Europe; £337.00 in Australasia and Japan; and £335.50 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.

Working with VBS datasets

THE PROBLEM

Variable Blocked Spanned datasets are one of the more tricky dataset types to work with, not least because neither ISPF browse nor REXX support them. No doubt this is why VBS datasets are not particularly common, but anyone who has tried working with SMF datasets knows that occasionally they do crop up, and then one is often left largely unsupported by the tools that normally come to hand.

Attempting to browse a VBS dataset from ISPF option 3.4 results in a screen like this:

```
------------------------------------------------------------------------
DSLIST - Datasets Matching TSO0001.MYFILES         Invalid record length
Command ===>                                            Scroll ===> CSR
LRECL and block size are inconsistent for RECFM=V dataset.
Command - Enter "/" to select action          Dsorg  Recfm  Lrecl  Blksz
------------------------------------------------------------------------
         TSO0001.MYFILES.DATAVB                PS    VB     32756  32760
B        TSO0001.MYFILES.DATAVBS               PS    VBS    32756  27998
```

While the error message is undoubtedly true in as far as it goes, it is only telling one part of the story. This is one of those error messages that experienced systems programmers just come to know means something more than it actually says, while juniors can often spend hours trying to figure out what is wrong with the dataset before noticing the all important S tagged after the VB.

REXX programs attempting to read VBS datasets are no more successful, but at least they give truly accurate error messages:

```
+IRX0509E Invalid record format for dataset allocated to file VBSDAT.
          RECFM must be fixed or variable. Spanned records or
          records with track overflow are not supported.
+IRX0670E EXECIO error while trying to GET or PUT a record.
```

This message resulted from running the following program, called REXXVBS, against the same data as in the ISPF browse above:

```
/*---- REXX ----*/
done = 'n'
do while done = 'n'
  "execio 1 diskr vbsdat"
  if rc = Ø then
    do
    parse pull vbsrec
    sid = substr(vbsrec,15,4)
    say 'SID' sid
    call proc_rec
    end
  else
    done = 'y'
end
exit Ø
```

with JCL as follows:

```
//REXXJCL  EXEC PGM=IRXJCL,PARM='REXXVBS'
//SYSEXEC  DD DSN=TSOØØØ1.SYSEXEC,DISP=SHR
//VBSDAT   DD DSN=TSOØØØ1.MYFILES.DATAVBS,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
```

THE SOLUTION

So what to do, if you want to examine the data in the VBS file in a raw state before it is processed by whatever utility is intended for the purpose?

Having found myself in this situation more than once, I decided that a useful utility would be a simple program to rewrite the data into a VB format dataset, which could then be quite happily processed by ISPF and REXX. I was dealing with SMF data, and it was important for my analysis to have a mechanism for selecting only a specified record type, and also to be able to create a dataset of a specified number of records.

The program I wrote is called VBS2VB, and it is shown below. It is specifically tailored to processing SMF datasets because that was my interest at the time, but it works perfectly well on any VBS file that you need to examine. The JCL to run the program is as follows:

```
//*************************************************************
//*  CONVERT VBS FORMAT DATASET TO VB FORMAT.
//*  SYSIN IS SMF REC TYPE REQD OR 'ALL', AND NUMBER OF OUTPUT
//*  RECORDS OR 'ALL'.
//*************************************************************
```

```
//VBS2VB   EXEC PGM=VBS2VB
//STEPLIB  DD DSN=TSO000001.LOADLIB,DISP=SHR
//VBSIN    DD DSN=TSO000001.MYFILES.DATAVBS,DISP=SHR
//*VBSIN   DD DSN=SYS1.MAN1,DISP=SHR
//*VBSIN   DD DSN=IPOSAV.SMFDUMP(0),DISP=SHR
//VBOUT    DD DSN=TSO000001.MYFILES.DATAVB,DISP=(NEW,CATLG),
//            UNIT=3390,VOL=SER=USR001,SPACE=(CYL,(1,1)),
//            DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VB,DSORG=PS)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
0070,000100
//
NNNN,NNNNNN
ALL ,NNNNNN
NNNN,ALL
ALL ,ALL
```

Note that the VBSIN dataset can be an SMF dataset like SYS1.MAN*x*, a dumped SMF dataset which could be on a tape based GDG, or any other VBS file.

The SYSIN parameters control how the program selects records to write to VBOUT, and they must be specified as a 4-byte value for SMF record number to select, a separator byte (a comma), and a 6-byte value for the number of records to be selected.

To specify SMF record type 70, as in the example above, the leading zeros must be included to give a value 0070. Likewise, to specify 100 records in the output file, the value must be written 000100.

Either or both parameters can be replaced by the literal ALL if no selection is desired.

Note that if the VBS dataset is not SMF data, then the first parameter is not meaningful, unless the data just happens to have a field at the same offset that SMF does for record type. But to modify the program to accommodate non-SMF records with a desired selector value elsewhere in the record is very simple, as explained below. VBS2VB produces a report after successful completion, as follows:

```
VBS2VB - CONVERT VBS DATA TO VB FOR REXX PROCESSING/ISPF BROWSE

  VBS RECORDS INPUT :             1201
  VB RECORDS OUTPUT :              100
```

Now the data is in a workable format, and ISPF or a REXX program such as the aforementioned REXXVBS can be used to read and

analyse the data successfully. Note that the VBOUT dataset uses a blocksize of 32760, and this is not the best use of DASD space, considering that, on a 3390 track, some 25KB of space is wasted. The VBOUT dataset will usually require more tracks than the VBSIN dataset is using for this reason.

VBS2VB

```
//ASSEM    EXEC  PGM=ASMA90,
//         PARM='DECK,NOOBJECT,LIST,XREF(FULL),ALIGN'
//SYSLIB   DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.MODGEN,DISP=SHR
//         DD DSN=TSO0001.SOURCE,DISP=SHR
//SYSUT1   DD UNIT=WORK,SPACE=(1700,(400,400))
//SYSUT2   DD UNIT=WORK,SPACE=(1700,(400,400))
//SYSUT3   DD UNIT=WORK,SPACE=(1700,(400,400))
//SYSPUNCH DD DSN=&&LOADSET,
//            UNIT=WORK,DISP=(,PASS),
//            SPACE=(400,(100,100,1)),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
*********************************************************************
**  REFORMAT DATASET FROM VBS TO VB FOR REXX AND ISPF BROWSE       **
*********************************************************************
         TITLE    'REFORMAT VBS TO VB'
         LCLC     &MODULE
&MODULE  SETC     'VBS2VB'
&MODULE  CSECT
&MODULE  AMODE    24
&MODULE  RMODE    24
*
         STM   14,12,12(13)          SAVE CALLER'S REGISTERS
         USING &MODULE,12            ESTABLISH ADDRESSABILITY
         LR    12,15                 SET UP MY BASE
         LR    14,13                 SAVE ADDR(PREVIOUS SAVE AREA)
         LA    13,SAVE               ADDR(MY SAVE AREA)
         ST    13,8(,14)             CHAIN MY SAVE AREA TO PREVIOUS
         ST    14,4(,13)             CHAIN PREVIOUS SAVE AREA TO MINE
         LM    14,1,12(13)           RESTORE REGS 14 ---> 1
*
@START   EQU   *
         XR    4,4                 COUNT INPUT RECS
         XR    5,5                 COUNT OUTPUT RECS
*
@PARMS   LA    2,SYSIN             LOAD ADDRESS OF INPUT DCB
         USING IHADCB,2            SET ADDRESSABILITY TO INPUT DCB
         OPEN  ((2),(INPUT))       OPEN PARAMETER FILE
@READP   GET   SYSIN               READ PARAMETER FILE
```

```
         LR    3,1                 SAVE INPUT RECORD ADDRESS
         MVC   SMFTYP,Ø(3)
         MVC   RECCNT,5(3)
@CLPRM   CLOSE (SYSIN)
*
         CLC   RECCNT,=CL6'ALL   '
         BE    @TYP
         PACK  DBLWRD,RECCNT
         CVB   6,DBLWRD
*
@TYP     CLC   SMFTYP,=CL4'ALL '
         BE    @OPEN
         PACK  DBLWRD,SMFTYP
         CVB   3,DBLWRD
         ST    3,SMFTYP
*
@OPEN    EQU   *
         OPEN  (SYSPRINT,(OUTPUT)) REPORT FILE
         OPEN  (VBOUT,(OUTPUT))    DATA OUTPUT FILE
         LA    2,VBSIN             LOAD ADDRESS OF INPUT DCB
         USING IHADCB,2            SET ADDRESSABILITY TO INPUT DCB
         OPEN  ((2),(INPUT))       OPEN INPUT FILE
*
@READS   EQU   *
         GET   VBSIN               READ INPUT
         LR    3,1                 SAVE INPUT RECORD ADDRESS
         LA    4,1(4)              COUNT INPUT RECS
         USING SMFØØHDR,3          SET ADDRESSABILITY TO SMF RECORD
         CLC   SMFTYP,=CL4'ALL '
         BE    @PROC
         CLC   SMFØØRTY,SMFT1      SEE IF TYPE REQUIRED
         BNE   @READS              NO, BYPASS RECORD
@PROC    PUT   VBOUT,Ø(3)
         LA    5,1(5)              COUNT OUTPUT RECS
         CLC   RECCNT,=CL6'ALL   '
         BE    @READS
         CR    6,5                 REQD OUTPUT RECS YET?
         BH    @READS
*
@CLVBS   CLOSE (VBSIN)             CLOSE FILES
         CLOSE (VBOUT)
*
         MVI   OUTREC,C' '
         MVC   OUTREC+1(132),OUTREC
         PUT   SYSPRINT,OUTCARD
         MVC   OUTREC+2(31),=CL31'VBS2VB - CONVERT VBS DATA TO VB'
         MVC   OUTREC+33(32),=CL32' FOR REXX PROCESSING/ISPF BROWSE'
         PUT   SYSPRINT,OUTCARD
*
         MVI   OUTREC,C' '
         MVC   OUTREC+1(132),OUTREC
         PUT   SYSPRINT,OUTCARD
```

```
*
         MVI   OUTREC,C' '
         MVC   OUTREC+1(132),OUTREC
         MVC   OUTREC+2(21),=CL21'  VBS RECORDS INPUT :'
         CVD   4,PWORK
         MVC   MASK16,EDMASK
         ED    MASK16,PWORK
         MVC   OUTREC+24(16),MASK16
         PUT   SYSPRINT,OUTCARD
*
         MVI   OUTREC,C' '
         MVC   OUTREC+1(132),OUTREC
         MVC   OUTREC+2(21),=CL21'  VB RECORDS OUTPUT :'
         CVD   5,PWORK
         MVC   MASK16,EDMASK
         ED    MASK16,PWORK
         MVC   OUTREC+24(16),MASK16
         PUT   SYSPRINT,OUTCARD
*
         MVI   OUTREC,C' '
         MVC   OUTREC+1(132),OUTREC
         PUT   SYSPRINT,OUTCARD
         CLOSE (SYSPRINT)
*
@FINISH  L     13,SAVE+4           RESTORE CALLERS REGS
         RETURN (14,12),RC=Ø       BACK TO CALLER
*
*********************************************************************
*
SYSIN    DCB   DDNAME=SYSIN,MACRF=GL,DSORG=PS,EODAD=@CLPRM,BFTEK=A
VBSIN    DCB   DDNAME=VBSIN,MACRF=GL,DSORG=PS,EODAD=@CLVBS,BFTEK=A
VBOUT    DCB   DDNAME=VBOUT,MACRF=PM,DSORG=PS,BLKSIZE=3276Ø,RECFM=VB
SYSPRINT DCB   DDNAME=SYSPRINT,DSORG=PS,MACRF=PM,                      X
               LRECL=137,BLKSIZE=137Ø,RECFM=VB
*
*********************************************************************
*
         DS    ØD
SAVE     DS    18F
DBLWRD   DS    D
SMFTYP   DS    ØF
         DS    XL3
SMFT1    DS    XL1
RECCNT   DS    XL6
         DS    ØD
PWORK    DC    PL8'Ø'
MASK16   DS    CL16
EDMASK   DC    XL16'4Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø212Ø'
*
OUTCARD  DC    AL2(137),AL2(Ø)
```

```
OUTREC   DS    CL133
         ORG   OUTREC+133
*
         DCBD  DSORG=QS,DEVD=DA
SMFT00   DSECT
         VBSHDR 00
         END
//*
//LNK      EXEC PGM=IEWL,COND=(7,LT)
//SYSUT1   DD UNIT=WORK,SPACE=(1024,(100,10))
//SYSLIB   DD DSN=TSO0001.LOADLIB,DISP=SHR
//SYSLMOD  DD DSN=TSO0001.LOADLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLIN   DD DSN=&&LOADSET,DISP=(OLD,DELETE)
//         DD *
      NAME VBS2VB(R)
/*
```

The macro VBSHDR is used to map the standard header fields of the SMF record. If the data being processed is not SMF data, but there is still a requirement to select according to a field in the record, then it is here that the simplest modification can achieve the desired result. Simply modifying the macro with filler fields to ensure that the record type field coincides with the specific record layout you are processing will work, although it would obviously be better to change the field names as well to avoid confusion.

VBSHDR

```
         MACRO
         VBSHDR &REC
SMF&REC.HDR DSECT
SMF&REC.LEN DS    BL2      RECORD LENGTH
SMF&REC.SEG DS    BL2      SEGMENT DESCRIPTOR
SMF&REC.FLG DS    BL1      HEADER FLAG BYTE
SMF&REC.RTY DS    BL1      RECORD TYPE
SMF&REC.TME DS    BL4      TOD RECORD WRITTEN
SMF&REC.DTE DS    PL4      DATE RECORD WRITTEN
SMF&REC.SID DS    CL4      SYSTEM ID
SMF&REC.SSI DS    CL4      SUBSYSTEM ID
SMF&REC.STY DS    BL2      SUBTYPE
         MEND
```

Patrick Mullen
MVS Systems Software Consultant (Canada)

A general purpose UCB information routine

INTRODUCTION

Here is a simple subroutine than can be invoked by other Assembler programs, or possibly even a high level language. The intention was to create only one routine to interface with the system to obtain UCB information on the whole range of available device types. The routine obtains the necessary storage required to hold copies of the obtained UCBs, and then passes back a pointer to the storage structure. The calling program can then process the data as needed. Copies of the $ESAPRO, $ESASTG, and $ESAEPI macros are included.

OPERATIONAL ENVIRONMENT

This routine has been used and validated on an MVS 5.2.2 system with DFSMS/MVS 1.3.

$UCBINFO

```
         TITLE '$UCBINFO - GENERAL UCB SCAN ROUTINE'
*---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+*
* CSECT   : $UCBINFO                                                 *
* MODULE  : $UCBINFO                                                 *
* AUTHOR  : ENTERPRISE DATA TECHNOLOGIES                             *
* DATE    : Ø5-19-98                                                 *
* DESC    : $UCBINFO IS A GENERAL PURPOSE SUBROUTINE THAT CAN BE USED *
*           TO OBTAIN COPIES OF UCBS.  THE OBTAINED INFORMATION IS   *
*           STORED INTO A CHAINED STORAGE STRUCTURE AND PASSED BACK  *
*           TO THE CALLING PROGRAM FOR FURTHER PROCESSING.  THE      *
*           CALLING PROGRAM NEEDS TO SPECIFY THE TYPE OF UCB THAT IS *
*           DESIRED.                                                 *
* MACROS  : $ESAPRO $ESAEPI $ESASTG STORAGE UCBSCAN                   *
* DSECTS  : UCB_STRU                                                 *
* INPUT   : NONE                                                     *
* OUTPUT  : NONE                                                     *
* PLIST   : STANDARD PARAMETER LIST                                  *
*           PLIST+X'ØØ'  ADDRESS OF 4 BYTE AREA CONTAINING UCB TYPE  *
*           PLIST=X'Ø4'  ADDRESS OF 4 BYTE AREA THAT WILL CONTAIN    *
*                        THE POINTER TO THE STORAGE STRUCTURE THAT   *
*                        CONTAINS THE UCBS                           *
* CALLS   : NONE                                                     *
* NOTES   : NONE                                                     *
*---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+*
```

```
         EJECT
$UCBINFO $ESAPRO R12,AM=31,RM=ANY
*
         ST    R1,@PLIST               SAVE PARM POINTER
         L     R2,Ø(R1)                PICK UP ADDRESS FIRST PARM
         L     RØ,G_SIZE               GET THE SIZE OF AREA TO OBTAIN
*
*—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+*
* OBTAIN FIRST STORAGE AREA FOR UCB INFORMATION                        *
*—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+*
*
         STORAGE OBTAIN,                                               +
               LENGTH=(Ø),                                             +
               LOC=(BELOW,ANY),                                        +
               ADDR=(R11)
*
         ST    R11,@F_POINT            SAVE POINTER FOR LATER
         ST    R11,@R_POINT            SAVE POINTER FOR LATER
         USING UCB_STRU,R11            LET ASSEMBLER KNOW THE BASE
         MVC   CHUNK_S,G_SIZE          SAVE SIZE OF CHUNK IN CHUNK
         LA    R11,D_SIZE(,R11)        INCREMENT ADDRESS
         XR    R1Ø,R1Ø                 CLEAR REGISTER 1Ø
*
         CLC   U_ALL,Ø(R2)             Q. ALL DEVICE TYPES
         BE    UCB_LOOP                A. YES
         CLC   U_CHAR,Ø(R2)            Q. CHARACTER READER DEVICES
         BNE   N_CHAR                  A. NO, NEXT CHECK
         OI    D_CLASS,E_SCAN_XDEVCLASS_CHAR
         B     UCB_LOOP                WE ARE READY TO ROLL
N_CHAR   DS    ØH
         CLC   U_COMM,Ø(R2)            Q. CHARACTER READER DEVICES
         BNE   N_COMM                  A. NO, NEXT CHECK
         OI    D_CLASS,E_SCAN_XDEVCLASS_COMM
         B     UCB_LOOP                WE ARE READY TO ROLL
N_COMM   DS    ØH
         CLC   U_CTC,Ø(R2)             Q. CHARACTER READER DEVICES
         BNE   N_CTC                   A. NO, NEXT CHECK
         OI    D_CLASS,E_SCAN_XDEVCLASS_CTC
         B     UCB_LOOP                WE ARE READY TO ROLL
N_CTC    DS    ØH
         CLC   U_DASD,Ø(R2)            Q. CHARACTER READER DEVICES
         BNE   N_DASD                  A. NO, NEXT CHECK
         OI    D_CLASS,E_SCAN_XDEVCLASS_DASD
         B     UCB_LOOP                WE ARE READY TO ROLL
N_DASD   DS    ØH
         CLC   U_DISP,Ø(R2)            Q. CHARACTER READER DEVICES
         BNE   N_DISP                  A. NO, NEXT CHECK
         OI    D_CLASS,E_SCAN_XDEVCLASS_DISP
         B     UCB_LOOP                WE ARE READY TO ROLL
N_DISP   DS    ØH
```

```
         CLC   U_TAPE,Ø(R2)            Q. CHARACTER READER DEVICES
         BNE   N_TAPE                  A. NO, NEXT CHECK
         OI    D_CLASS,E_SCAN_XDEVCLASS_TAPE
         B     UCB_LOOP                WE ARE READY TO ROLL
N_TAPE   DS    ØH
         CLC   U_UREC,Ø(R2)            Q. CHARACTER READER DEVICES
         BNE   N_UREC                  A. NO, NEXT CHECK
         OI    D_CLASS,E_SCAN_XDEVCLASS_UREC
         B     UCB_LOOP                WE ARE READY TO ROLL
*
N_UREC   DS    ØH
*
         MVC   RET_CODE,RCØØ1Ø         SET RETURN TO INDICATE ERROR
         B     EXIT_PGM                EXIT THE PROGRAM
*
*-----------------------------------------------------------------------*
* MAIN LOOP FOR OBTAINING THE UCB INFORMATION                           *
*-----------------------------------------------------------------------*
*
UCB_LOOP DS    ØH
*
         XC    Ø(D_SIZE,R11),Ø(R11)    MAKE SURE THE AREA IS CLEARED
*
         UCBSCAN COPY,                                                  +
               WORKAREA=UCB_WORK,                                       +
               UCBAREA=UCB_AREA,                                        +
               DEVNCHAR=UCB_UNIT,                                       +
               DYNAMIC=YES,                                             +
               RANGE=ALL,                                               +
               DEVCID=D_CLASS,                                          +
               RETCODE=UET_CODE,                                        +
               RSNCODE=USN_CODE,                                        +
               MF=(E,E_SCAN)
*
         CLC   UET_CODE,RCØØØØ         Q. ZERO RETURN CODE
         BNE   EXIT_PGM                A. NO, LOOKS LIKE WE ARE DONE
         LA    R1Ø,1(,R1Ø)             INCREMENT THE COUNTER
         LA    R11,D_SIZE(,R11)        BUMP THE POINTER TO NEXT AREA
         C     R1Ø,F1ØØØ               Q. PROCESSED 1ØØØ ENTRIES
         BNE   UCB_LOOP                A. NO, KEEP GOING
*
         L     R9,@R_POINT             POINT TO THE BEGINNING OF CHUNK
         ST    R1Ø,Ø(R9)               SAVE THE NUMBER OF ENTRIES
         L     RØ,G_SIZE               GET THE SIZE OF AREA TO OBTAIN
*
*-----------------------------------------------------------------------*
* GET ANOTHER CHUNK OF STORAGE AND CHAIN IT UP TO PREVIOUS CHUNK        *
*-----------------------------------------------------------------------*
*
         STORAGE OBTAIN,                                                +
```



```
               LENGTH=(Ø),                                             +
               LOC=(BELOW,ANY),                                        +
               ADDR=(R11)
*
          MVC   CHUNK_S,G_SIZE          SAVE SIZE OF CHUNK IN CHUNK
          ST    R11,@R_POINT            SAVE THE FORWARD POINTER
          ST    R11,4(R9)               SAVE THE FORWARD POINTER
          ST    R9,8(R11)               STORE THE BACKWARD POINTER
          XR    R1Ø,R1Ø                 CLEAR REG 1Ø
          LA    R11,D_SIZE(,R11)        BUMP POINTER
          B     UCB_LOOP                GO PROCESS THE NEXT UCB
          DROP  R11                     NOTIFY THE ASSEMBLER
*
EXIT_PGM DS    ØH
*
          L     R9,@R_POINT             POINT TO THE BEGINNING OF CHUNK
          ST    R1Ø,Ø(R9)               SAVE THE NUMBER OF ENTRIES
          L     R2,@PLIST               GET THE PLIST POINTER
          L     R2,4(R2)                PICK UP ADDRESS OF SECOND PARM
          MVC   Ø(4,R2),@F_POINT        PASS POINTER BACK TO CALLER
*
          $ESAEPI RET_CODE
*
NUM_ENT   EQU   1ØØØ                    NUMBER OF ENTRIES IN A BLOCK
G_SIZE    DC    A((D_SIZE*NUM_ENT)+D_SIZE)
F1ØØØ     DC    A(NUM_ENT)
*
U_ALL     DC    CL4'ALL '               ALL DEVICES
U_CHAR    DC    CL4'CHAR'               CHARACTER READER DEVICES
U_COMM    DC    CL4'COMM'               COMMUNICATIONS DEVICES
U_CTC     DC    CL4'CTC '               CHANNEL TO CHANNEL DEVICES
U_DASD    DC    CL4'DASD'               DASD DEVICES
U_DISP    DC    CL4'DISP'               TAPE DEVICES
U_TAPE    DC    CL4'TAPE'               TAPE DEVICES
U_UREC    DC    CL4'UREC'               UNIT RECORD DEVICES
*
          EJECT
          $ESASTG
@PLIST    DS    F                       PLACE TO SAVE PLIST POINTER
@F_POINT  DS    F                       POINTER TO STORAGE STRUCTURE
@R_POINT  DS    F                       POINTER TO CURRENT CHUNK
RET_CODE  DS    F                       PROGRAM RETURN CODE
UET_CODE  DS    F                       UCBSCAN RETURN CODE
USN_CODE  DS    F                       UCBSCAN REASON CODE
D_CLASS   DS    XL1                     DEVICE TYPE USED BY UCBSCAN
UCB_WORK  DS    1ØØXL1                  WORK AREA
          EJECT
          UCBSCAN PLISTVER=MAX,MF=(L,E_SCAN)
          EJECT
```

```
*--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+*
* STORAGE STRUCTURE WILL USES 52 BYTES PER UCB COPY AND 4-CHARACTER   *
* UCB ADDRESS INFORMATION.  FIRST ENTRY IN THE STRUCTURE IS A SPECIAL *
* LAYOUT, AS DOCUMENTED BELOW.  THE SIZE OF A CHUNK CAN BE SET BY     *
* THE VALUE OF NUM_ENT.  NUM_ENT IS CURRENTLY 1000.  YOU CAN SET THIS *
* TO WHATEVER VALUE YOU DESIRE.                                       *
*                                                                     *
*         +----+                                                      *
*         | 4 BYTES |  NUMBER OF ENTRIES IN A CHUNK                   *
*         +----+                                                      *
*         | 4 BYTES |  POINTER TO NEXT CHUNK                          *
*         +----+                                                      *
*         | 4 BYTES |  POINTER TO PREVIOUS CHUNK                      *
*         +----+                                                      *
*         | 4 BYTES |  SIZE OF THE CHUNK                            *
*         +----+                                                      *
*         | 36 BYTES|  FILLER                                       *
*         +----+                                                      *
*         | 4 BYTES |  4 CHARACTER UCB                              *
*         +----+----------+                                          *
*         | 48 BYTES                        | COPY OF UCB           *
*         +-----------------+                                         *
*         | 4 BYTES |  4 CHARACTER UCB                              *
*         +----+----------+                                          *
*         | 48 BYTES                        | COPY OF UCB           *
*         +-----------------+                                         *
*            .                                                        *
*             .     REPEAT                                          *
*              .      UNTIL                                         *
*               .       STRUCTURE                                   *
*                .        IS                                        *
*                 .         FULL                                    *
*         +----+                                                      *
*         + 4 BYTES |  4 CHARACTER UCB                              *
*         +----+----------+                                          *
*         + 48 BYTES                      | COPY OF UCB           *
*         +-----------------+                                         *
*--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+*
UCB_STRU DSECT                          STRUCTURE DEFINITION
UCB_STOR DS    ØXL52                    SPECIFY SIZE
UCB_##   DS    F                        NUMBER OF ENTRIES IN CHUNK
UCB_FP   DS    A                        POINTER TO NEXT CHUNK
UCB_BP   DS    A                        POINTER TO PREVIOUS CHUNK
CHUNK_S  DS    F                        SIZE OF THE CHUNK
         DS    (52-(*-UCB_STOR))XL1     FILL IT OUT
         ORG   UCB_STOR                 ORG BACK FOR MULTIPLE DEFN.
UCB_UNIT DS    4XL1                     4 CHARACTER UCB
UCB_AREA DS    48XL1                    RETURNED COPY OF THE UCB
D_SIZE   EQU   *-UCB_UNIT               LET ASSEMBLER CALCULATE LENGTH
         END   $UCBINFO
         MACRO
```

$ESAPRO MACRO

```
&LABEL   $ESAPRO &AM=31,&RM=ANY,&MODE=P
.*********************************************************************
.*
.*       THIS MACRO WILL PROVIDE ENTRY LINKAGE AND OPTIONALLY
.*       MULTIPLE BASE REGISTERS.  TO USE THIS MACRO, YOU NEED TO
.*       ALSO USE THE $ESASTG MACRO.  THE $ESASTG DEFINES THE SYMBOL
.*       QLENGTH WHICH OCCURS IN THE CODE THAT &ESAPRO GENERATES.
.*       IF YOU DO NOT CODE ANY OPERANDS, THEN REGISTER 12 WILL BE
.*       USED AS THE BASE.  IF YOU CODE MULTIPLE SYMBOLS, THEN THEY
.*       WILL BE USED AS THE BASE REGISTERS.
.*
.*       EXAMPLES:
.*
.*              SECTNAME $ESAPRO          = REG 12 BASE
.*              SECTNAME $ESAPRO 5        = REG 5 BASE
.*              SECTNAME $ESAPRO R10,R11  = REGS 10 AND 11 ARE BASES
.*
.*********************************************************************
*
         LCLA  &AA,&AB,&AC
*
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
RA       EQU   10
R11      EQU   11
RB       EQU   11
R12      EQU   12
RC       EQU   12
R13      EQU   13
RD       EQU   13
R14      EQU   14
RE       EQU   14
R15      EQU   15
RF       EQU   15
*
FPR0     EQU   0
FPR2     EQU   2
FPR4     EQU   4
FPR6     EQU   6
*
```

```
&LABEL   CSECT
&LABEL   AMODE &AM
&LABEL   RMODE &RM
*
         SYSSTATE ASCENV=&MODE         SET THE ENVIRONMENT
*
         B     $$$$EYEC-*(R15)         BRANCH AROUND EYECATCHER
         DC    AL1(($$$$EYEC-*)-1)     EYECATCHER LENGTH
         DC    CL8'&LABEL'             MODULE ID
         DC    CL3' - '
         DC    CL8'&SYSDATE'           ASSEMBLY DATE
         DC    CL3' - '
         DC    CL8'&SYSTIME'           ASSEMBLY TIME
         DC    CL3'   '                FILLER
*
$$$$F1SA DC    CL4'F1SA'               USED FOR STACK OPERATIONS
$$$$4Ø96 DC    F'4Ø96'                 USED TO ADJUST BASE REGS
*
$$$$EYEC DS    ØH
*
         BAKR  R14,Ø                   SAVE GPRS AND ARS ON THE STACK
         AIF   (N'&SYSLIST EQ Ø).USER12
         LAE   &SYSLIST(1),Ø(R15,Ø)    LOAD OUR BASE REG
         USING &LABEL,&SYSLIST(1)      LET THE ASSEMBLER KNOW
         AGO   .GNBASE
.USER12  ANOP
         MNOTE *,'NO BASE REG SPECIFIED, REGISTER 12 USED'
         LAE   R12,Ø(R15,Ø)            LOAD OUR BASE REG
         USING &LABEL,R12              LET THE ASSEMBLER KNOW
         AGO   .STGOB
.GNBASE  ANOP
         AIF   (N'&SYSLIST LE 1).STGOB
&AA      SETA  2
&AC      SETA  4Ø96
.GNBASE1 ANOP
*
         AIF   (&AA GT N'&SYSLIST).STGOB
&AB      SETA  &AA-1
         LR    &SYSLIST(&AA),&SYSLIST(&AB) GET INITIAL BASE
         A     &SYSLIST(&AA),$$$$4Ø96      ADJUST NEXT BASE
         USING &LABEL+&AC,&SYSLIST(&AA)    LET THE ASSEMBLER KNOW
&AA      SETA  &AA+1
&AC      SETA  &AC+4Ø96
         AGO   .GNBASE1
.STGOB   ANOP
*
         L     RØ,QLENGTH              GET THE DSECT LENGTH
*
         STORAGE OBTAIN,LENGTH=(RØ),LOC=(RES,ANY)
*
```

```
         LR    R15,R1                  GET @(OBTAINED AREA)
         L     R13,QDSECT              GET DISPLACEMENT INTO AREA
         LA    R13,Ø(R13,R15)          GET @(OBTAINED AREA)
         LR    RØ,R13                  SET REG Ø = REG 13
         L     R1,QLENGTH              GET THE LENGTH OF THE AREA
         XR    R15,R15                 CLEAR REG 5
         MVCL  RØ,R14                  INTIALIZE THE AREA
         MVC   4(4,R13),$$$$F1SA       INDICATE STACK USAGE
         USING DSECT,R13               INFORM ASSEMBLER OF BASE
.MEND    ANOP
*
         EREG  R1,R1                   RESTORE REGISTER 1
         MEND
```

$ESAEPI MACRO

```
         MACRO
         $ESAEPI
.*************************************************************************
.*
.*       THIS MACRO WILL PROVIDE EXIT LINKAGE. IT WILL FREE THE
.*       STORAGE AREA THAT WAS ACQUIRED BY THE $ESAPRO MACRO.  YOU
.*       CAN OPTIONALLY PASS IT A RETURN CODE VALUE.  THIS VALUE IS
.*       EITHER THE LABEL OF A FULL WORD IN STORAGE, OR IT IS A REG-
.*       ISTER. AS WITH THE $ESAPRO MACRO, YOU NEED TO USE THE $ESASTG
.*       MACRO.  THE SYMBOL QLENGTH WHICH OCCURS IN THE CODE THAT IS
.*       GENERATED BY THIS MACRO IS DEFINED BY $ESASTG
.*
.*       EXAMPLES:
.*
.*              $ESAEPI          = NO RETURN CODE SPECIFIED
.*              $ESAEPI (R5)     = RETURN CODE IS IN REG 5
.*              $ESAEPI RETCODE  = RETURN CODE IS IN THE FULLWORD AT
.*                                 RETCODE
.*
.*************************************************************************
*
         AIF   (N'&SYSLIST EQ Ø).STGFRE
*
         AIF   ('&SYSLIST(1)'(1,1) EQ '(').REGRC
         L     R2,&SYSLIST(1)          GET RETURN CODE VALUE
         AGO   .STGFRE
.REGRC   ANOP
         LR    R2,&SYSLIST(1,1)        GET RETURN CODE VALUE
.STGFRE  ANOP
*
         L     RØ,QLENGTH              GET THE DSECT LENGTH
*
         STORAGE RELEASE,LENGTH=(RØ),ADDR=(R13)
*
```

```
         AIF   (N'&SYSLIST NE Ø).SETRC
         XR    R15,R15                 CLEAR THE RETURN CODE
         AGO   .MEND
.SETRC   ANOP
         LR    R15,R2                  SET THE RETURN CODE
.MEND    ANOP
         PR                            RETURN TO CALLER
* FOR ADDRESSABILITY PURPOSES
         LTORG
         MEND
```

$ESASTG MACRO

```
         MACRO
         $ESASTG
.*************************************************************************
.*
.*       THIS MACRO IS USED IN CONJUNCTION WITH THE $ESAEPI AND $ESAPRO
.*       MACROS.  IT PROVIDES A Q TYPE ADDRESS CONSTANT. A REGISTER
.*       SAVE AREA ID PROVIDED AS WELL.
.*
.*       EXAMPLES:
.*
.*              $ESASTG
.*      XXX     DC    F          = DEFINE ADDITIONAL STORAGE AREA
.*      YYY     DC    XL255
.*       .       .      .
.*       .       .      .
.*       .       .      .
.*
.*************************************************************************
RC0000   DC    F'Ø'                    USED TO SET RETURN CODES
RC0004   DC    F'4'                    USED TO SET RETURN CODES
RC0008   DC    F'8'                    USED TO SET RETURN CODES
RC000C   DC    F'12'                   USED TO SET RETURN CODES
RC0010   DC    F'16'                   USED TO SET RETURN CODES
QDSECT   DC    Q(DSECT)                DEFINE A QCON
QLENGTH  CXD                           LET ASM CALCULATE THE LENGTH
DSECT    DSECT
         DS    18F                     SET ASIDE REGISTER SAVE AREA
         MEND
```


Reinstalling a product using SMP/E BUILDMCS

INTRODUCTION

MVS Update Issues 98 (December 1994) and 119 (August 1996) contained articles describing different ways to copy SMP/E entries from one CSI to another. This article presents a new SMP/E command (BUILDMCS), which is the IBM solution to that type of problem.

Starting with OS/390 SMP/E Release 2, the new BUILDMCS command provides an automated process for copying selected products from one SMP/E environment to another.

This can be especially useful to extract specific products from an OS/390 ServerPac to desynchronize the installation and migration of the core of MVS and the migration of other strategic products such as NetView or OPC/ESA.

Itcan also be interesting to extract an existing product from an old ServerPac CSI. In our case, it was our main motivation because we discovered that it was no longer possible to order the COBOL II product in an OS/390 Release 2.5 ServerPac. Because we still wanted to be able to apply PTFs to our COBOL II libraries, we had to find a solution to extract SMP/E data from the OS/390 Release 3 CSI to create a new separate CSI environment to maintain our old COBOL II version.

The BUILDMCS command creates MCS and JCLIN needed as input to RECEIVE, APPLY, and ACCEPT processing for reinstallation of products in another SMP/E environment. It extracts, from the existing CSI, all available data, to create required MCS and JCLIN commands streams to reinstall the specified FMIDs.

The rest of this article contains a step-by-step description of how we used the BUILDMCS command to extract COBOL II from our OS/390 Release 3 CSI to reinstall it in a new set of SMP/E zones and in a new set of libraries. This documented example will help readers to use this new facility.

ACCEPT ALL ALREADY APPLIED MAINTENANCE

Before starting the BUILDMCS command, all maintenance applied on the specified FMIDs should first be accepted.

```
//STEPØ1   EXEC PGM=GIMSMP,PARM='PROCESS=WAIT'
//*
//SMPCSI   DD DISP=SHR,DSN=OS39ØRØ3.GLOBAL.CSI
//*
//SMP.SMPCNTL DD *
  SET BDY(COBOLD).
  ACCEPT  FORFMID(
                  HCL14ØØ ,
                  HCQ14ØØ ,
                  JCL14Ø2 ,
                  JCL14Ø3 ,
                  JCL14Ø4 ,
                  JCL14Ø6 ,
                  JCL14Ø7 ,
                  JCQ14Ø2 ,
                 )
          BYPASS(HOLDSYSTEM)
          GROUPEXTEND
          COMPRESS(ALL)
         .
/*
```

EXECUTE SMP/E BUILDMCS COMMAND

The output of the BUILDMCS command is a superseding function SYSMOD for each function related to specified FMIDs.

```
//STEPØ1   EXEC PGM=GIMSMP,PARM='PROCESS=WAIT'
//*
//SMPCSI   DD DISP=SHR,DSN=OS39ØRØ3.GLOBAL.CSI
//*
//SMPPUNCH DD DISP=(,CATLG),DSN=I99Ø557.SMP.BUILDMCS,
//            UNIT=TSO$,
//            SPACE=(TRK,(5,5)),
//            DCB=(RECFM=FBA,LRECL=133,BLKSIZE=133Ø)
//*
//SMP.SMPCNTL DD *
  SET BDY(COBOLD).
  BUILDMCS FORFMID(
                  HCL14ØØ ,
                  HCQ14ØØ ,
                  JCL14Ø2 ,
                  JCL14Ø3 ,
                  JCL14Ø4 ,
```

```
                    JCL1406 ,
                    JCL1407 ,
                    JCQ1402 ,
                   )
          .
/*
```

The content of the resulting SMPPUNCH dataset is shown below:

```
++FUNCTION(HCL1400) FESN(6594001) REWORK(1998106)
/*****************************************************************/
/*****************************************************************/
/*** THIS MCS FOR FUNCTION HCL1400 WAS CREATED BY THE BUILDMCS  ***/
/*** COMMAND ON 1998106                                         ***/
/*****************************************************************/
/*****************************************************************/ .
++VER(Z038) DELETE(HCL1100,HCL1103,HCL1200,HCL1300,HCL1310,HCL1320) SUP(
  AN28427,AN33499,AN37798,AN37800,AN38353,AN38354,AN38495,AN38496,
  AN38556,AN38624,AN38748,AN38750,AN38751,AN40278,AN40347,AN41823,
  AN42731,AN42801,AN42977,AN43816,AN44819,AN46018,AN46122,AN47922,
  AN49897,AN50279,AN50986,AN51110,AN51748,AN52277,AN54408,AN55043,
  AN55636,AN55690,AN57566,AN58207,AN59231,AN59280,AN59624,AN59656,
  AN59933,AN60258,AN60490,AN63411,AN63802,AN64893,AN65079,AN65736,
  AN66057,AN66777,AN67719,AN67919,AN67934,AN68038,AN68728,AN68767,
  AN70201,AN70713,AN70763,AN70874,AN70936,AN71111,AN72097,AN74000,
  AN74360,AN74671,AN75121,AN75195,AN76388,AN76525,AN76610,AN76666,
  AN77418,AN79840,AN86010,AN90431,AN92652,BN51110,UN40103,UN40129,
....
++MAC(IGZBRDGE) DISTLIB(COB2LSRC) FROMDS(DSN(SYS1.COB2LSRC) NUMBER(1))
  SYSLIB(MACLIB).
++MAC(IGZOPD) DISTLIB(COB2LSRC) FROMDS(DSN(SYS1.COB2LSRC) NUMBER(1))
  SYSLIB(MACLIB).
++MAC(IGZOPT) DISTLIB(COB2LSRC) FROMDS(DSN(SYS1.COB2LSRC) NUMBER(1))
  SYSLIB(MACLIB).
++MAC(IGZTUNE) DISTLIB(COB2LSRC) FROMDS(DSN(SYS1.COB2LSRC) NUMBER(1))
  SYSLIB(MACLIB).
....
```

The data elements (++MAC, ++MOD, ++SRC…) MCS, created by the BUILDMCS command, contain a new FROMDS operand that specifies dataset names of SMP/E distribution libraries which will be used during SMP/E receive processing to create a SMPTLIB dataset.

CREATE A NEW SET OF SMP/E ZONES

To reinstall the product, you need to initialize a new set of SMP/E datasets.

```
//CSI    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD  *
      DELETE SMAINT.COBOL2.V140.GLOBAL.CSI
      DELETE SMAINT.COBOL2.V140.SMPLOG
      DELETE SMAINT.COBOL2.V140.SMPLOGA
      DELETE SMAINT.COBOL2.V140.SMPMTS
      DELETE SMAINT.COBOL2.V140.SMPPTS
      DELETE SMAINT.COBOL2.V140.SMPSTS
      DELETE SMAINT.COBOL2.V140.SMPMTS
      DELETE SMAINT.COBOL2.V140.SMPSCDS
      SET MAXCC = 0

      DEFINE CLUSTER                        -
             (NAME(SMAINT.COBOL2.V140.GLOBAL.CSI) -
              FREESPACE(10,5)               -
              KEYS(24 0)                    -
              RECORDSIZE(24 143)            -
              SHAREOPTIONS(2)               -
              REUSE                         -
              VOLUMES(MNT$02)               -
                             )              -
          DATA                              -
             (NAME(SMAINT.COBOL2.V140.GLOBAL.CSI.DATA) -
              CONTROLINTERVALSIZE(4096)     -
              CYLINDERS(8 2)                -
                             )              -
          INDEX                             -
             (NAME(SMAINT.COBOL2.V140.GLOBAL.CSI.INDEX) -
              CYLINDERS(2 1)                -
              IMBED                         -
                             )
/*
//ALLOC  EXEC PGM=IEFBR14
//* ******************************************************************
//*           SMP DATASETS
//* ******************************************************************
//SMPLOG   DD DSN=SMAINT.COBOL2.V140.SMPLOG,DCB=(BLKSIZE=3200,RECFM=U),
//            UNIT=SYSALLDA,VOL=SER=MNT$02,DISP=(NEW,CATLG),
//            SPACE=(3200,(6000,3000),,,ROUND)
//SMPLOGA  DD DSN=SMAINT.COBOL2.V140.SMPLOGA,
//            DCB=(BLKSIZE=3200,RECFM=U),
//            UNIT=SYSALLDA,VOL=SER=MNT$02,DISP=(NEW,CATLG),
//            SPACE=(3200,(3000,1500),,,ROUND)
//SMPMTS   DD DSN=SMAINT.COBOL2.V140.SMPMTS,DISP=(NEW,CATLG),
//            UNIT=SYSALLDA,VOL=SER=MNT$02,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),
//            SPACE=(3120,(60,20,60),,,ROUND)
//SMPPTS   DD DSN=SMAINT.COBOL2.V140.SMPPTS,DISP=(NEW,CATLG),
//            UNIT=SYSALLDA,VOL=SER=MNT$02,
```

```
//            DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=312Ø),
//            SPACE=(312Ø,(4ØØ,2ØØ,3ØØ),,,ROUND)
//SMPSCDS  DD DSN=SMAINT.COBOL2.V14Ø.SMPSCDS,DISP=(NEW,CATLG),
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=312Ø),
//            SPACE=(312Ø,(8ØØØ,4ØØØ,3ØØ),,,ROUND)
//SMPSTS   DD DSN=SMAINT.COBOL2.V14Ø.SMPSTS,DISP=(NEW,CATLG),
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=312Ø),
//            SPACE=(312Ø,(4ØØ,1ØØ,15Ø),,,ROUND)
//*
//* ********************************************************************
//*           INITIALIZE (VSAM REPRO) CSI DATASET
//* ********************************************************************
//*
//REPRO  EXEC PGM=IDCAMS
//SMPCSI   DD DSN=SMAINT.COBOL2.V14Ø.GLOBAL.CSI,DISP=SHR
//SYSPRINT DD SYSOUT=*
//ZPOOL    DD DISP=SHR,DSN=SYS1.MACLIB(GIMZPOOL)
//SYSIN DD  *
  REPRO OUTFILE(SMPCSI) INFILE(ZPOOL)
//*
//* ********************************************************************
//*           INITIALIZE SMP/E FOR COBOL2
//* ********************************************************************
//*
//INITSMP EXEC PGM=GIMSMP
//SMPCSI   DD DSN=SMAINT.COBOL2.V14Ø.GLOBAL.CSI,DISP=SHR
//SMPLOG   DD DSN=SMAINT.COBOL2.V14Ø.SMPLOG,DISP=SHR
//SMPLOGA  DD DSN=SMAINT.COBOL2.V14Ø.SMPLOGA,DISP=SHR
//SMPPTS   DD DSN=SMAINT.COBOL2.V14Ø.SMPPTS,DISP=SHR
//SMPOUT   DD SYSOUT=*
//*
//SMPCNTL  DD *
  SET BDY(GLOBAL).
  UCLIN .
    ADD OPTIONS(OPTIONS)
               NUCID(9)
               DSPREFIX(SMAINT.COBOL2.V14Ø)
               DSSPACE(2ØØ,2ØØ,6ØØ)
               PEMAX(55ØØ)
               ASM(ASM) .
    ADD UTILITY(IEWL)
               PARM(SIZE=(9ØØK,124K),LET,LIST,XREF,NCAL) .
    ADD UTILITY(ASM)
               NAME(ASMA9Ø) .
    ADD GLOBALZONE
            SREL(Z Ø38)
            OPTIONS(OPTIONS) .
    ENDUCL.
```

```
SET BDY(GLOBAL).
UCLIN.
   ADD GZONE
       ZONEINDEX(
          (TARGET,SMAINT.COBOL2.V140.GLOBAL.CSI,TARGET)
                )   .
   ADD GZONE
       ZONEINDEX(
          (DLIB,SMAINT.COBOL2.V140.GLOBAL.CSI,DLIB)
                )   .
ENDUCL.
SET BDY(TARGET).
UCLIN.
ADD TARGETZONE(TARGET)
    OPTIONS(OPTIONS)
    SREL(Z038)
    RELATED(DLIB).
ENDUCL.
SET BDY(DLIB).
UCLIN.
ADD DLIBZONE(DLIB)
    OPTIONS(OPTIONS)
    SREL(Z038)
    RELATED(TARGET).
ENDUCL.

SET BDY(GLOBAL)  .
UCLIN.
   /*************************************************************/
   /* DDDEF FOR GLOBAL ZONE (SMP)                               */
   /*************************************************************/
   ADD DDDEF(SYSPRINT) SYSOUT(*) .
   ADD DDDEF(SMPOUT  ) SYSOUT(*) .
   ADD DDDEF(SMPLOG  ) DA(SMAINT.COBOL2.V140.SMPLOG) MOD .
   ADD DDDEF(SMPLOGA ) DA(SMAINT.COBOL2.V140.SMPLOGA) MOD .
   ADD DDDEF(SMPPTS  ) DA(SMAINT.COBOL2.V140.SMPPTS) SHR .
   ADD DDDEF(SMPTLIB ) DSPREFIX(SMAINT.COBOL2.V140) VOLUME(MNT$02)
       UNIT(SYSDA) .
   ADD DDDEF(SYSUT1  ) NEW DELETE CYL SPACE(5,1) UNIT(SYSDA) .
   ADD DDDEF(SYSUT2  ) NEW DELETE CYL SPACE(5,1) UNIT(SYSDA) .
   ADD DDDEF(SYSUT3  ) NEW DELETE CYL SPACE(5,1) UNIT(SYSDA) .
   ADD DDDEF(SYSUT4  ) NEW DELETE CYL SPACE(5,1) UNIT(SYSDA) .
   ADD DDDEF(SMPWRK1 ) NEW DELETE CYL SPACE(10,5) DIR(250)
       UNIT(SYSDA) .
   ADD DDDEF(SMPWRK2 ) NEW DELETE CYL SPACE(10,5) DIR(250)
       UNIT(SYSDA) .
   ADD DDDEF(SMPWRK3 ) NEW DELETE CYL SPACE(10,5) DIR(250)
       UNIT(SYSDA) .
   ADD DDDEF(SMPWRK4 ) NEW DELETE CYL SPACE(10,5) DIR(250)
       UNIT(SYSDA) .
```

```
   ADD DDDEF(SMPWRK6 ) NEW DELETE CYL SPACE(10,5) DIR(300)
       UNIT(SYSDA) .
ENDUCL.
SET BDY(TARGET).         /* TARGET ZONE INITIALIZATION */
UCLIN.
   /*********************************************************/
   /* DDDEF FOR TARGET ZONE (SMP)                           */
   /*********************************************************/
   ADD DDDEF(SYSPRINT) SYSOUT(*) .
   ADD DDDEF(SMPOUT  ) SYSOUT(*) .
   ADD DDDEF(SMPLOG  ) DA(SMAINT.COBOL2.V140.SMPLOG) MOD .
   ADD DDDEF(SMPLOGA ) DA(SMAINT.COBOL2.V140.SMPLOGA) MOD .
   ADD DDDEF(SMPMTS  ) DA(SMAINT.COBOL2.V140.SMPMTS) SHR .
   ADD DDDEF(SMPPTS  ) DA(SMAINT.COBOL2.V140.SMPPTS) SHR .
   ADD DDDEF(SMPSTS  ) DA(SMAINT.COBOL2.V140.SMPSTS) SHR .
   ADD DDDEF(SMPSCDS ) DA(SMAINT.COBOL2.V140.SMPSCDS) SHR .
   ADD DDDEF(SMPTLIB ) VOLUME(MNT$02) UNIT(SYSDA) .
   ADD DDDEF(SMPWRK1 ) NEW DELETE CYL SPACE(10,5) DIR(250)
       UNIT(SYSDA) .
   ADD DDDEF(SMPWRK2 ) NEW DELETE CYL SPACE(10,5) DIR(250)
       UNIT(SYSDA) .
   ADD DDDEF(SMPWRK3 ) NEW DELETE CYL SPACE(10,5) DIR(250)
       UNIT(SYSDA) .
   ADD DDDEF(SMPWRK4 ) NEW DELETE CYL SPACE(10,5) DIR(250)
       UNIT(SYSDA) .
   ADD DDDEF(SMPWRK6 ) NEW DELETE CYL SPACE(10,5) DIR(300)
       UNIT(SYSDA) .
   ADD DDDEF(SYSUT1  ) NEW DELETE CYL SPACE(5,1) UNIT(SYSDA) .
   ADD DDDEF(SYSUT2  ) NEW DELETE CYL SPACE(5,1) UNIT(SYSDA) .
   ADD DDDEF(SYSUT3  ) NEW DELETE CYL SPACE(5,1) UNIT(SYSDA) .
   ADD DDDEF(SYSUT4  ) NEW DELETE CYL SPACE(5,1) UNIT(SYSDA) .
   ADD DDDEF(MODGEN ) DA(SYS1.MODGEN) SHR .
   ADD DDDEF(AMODGEN) DA(SYS1.AMODGEN) SHR .
   ADD DDDEF(MACLIB ) DA(SYS1.MACLIB) SHR .
   ADD DDDEF(AMACLIB) DA(SYS1.AMACLIB) SHR .
   /*********************************************************/
   /* DDDEF FOR TARGET ZONE                                 */
   /*********************************************************/
   ADD DDDEF(COB2CICS ) DA(SMAINT.COBOL2.V140.COB2CICS ) SHR .
   ADD DDDEF(COB2CLIB ) DA(SMAINT.COBOL2.V140.COB2CLIB ) SHR .
   ADD DDDEF(COB2COBJ ) DA(SMAINT.COBOL2.V140.COB2COBJ ) SHR .
   ADD DDDEF(COB2COMP ) DA(SMAINT.COBOL2.V140.COB2COMP ) SHR .
   ADD DDDEF(COB2CSRC ) DA(SMAINT.COBOL2.V140.COB2CSRC ) SHR .
   ADD DDDEF(COB2DOBJ ) DA(SMAINT.COBOL2.V140.COB2DOBJ ) SHR .
   ADD DDDEF(COB2DSRC ) DA(SMAINT.COBOL2.V140.COB2DSRC ) SHR .
   ADD DDDEF(COB2EOBJ ) DA(SMAINT.COBOL2.V140.COB2EOBJ ) SHR .
   ADD DDDEF(COB2ESRC ) DA(SMAINT.COBOL2.V140.COB2ESRC ) SHR .
   ADD DDDEF(COB2LIB  ) DA(SMAINT.COBOL2.V140.COB2LIB  ) SHR .
   ADD DDDEF(COB2LOBJ ) DA(SMAINT.COBOL2.V140.COB2LOBJ ) SHR .
   ADD DDDEF(COB2LSRC ) DA(SMAINT.COBOL2.V140.COB2LSRC ) SHR .
```

```
   ADD DDDEF(COB2MLIB ) DA(SMAINT.COBOL2.V14Ø.COB2MLIB ) SHR .
   ADD DDDEF(COB2PLIB ) DA(SMAINT.COBOL2.V14Ø.COB2PLIB ) SHR .
   ADD DDDEF(COB29OBJ ) DA(SMAINT.COBOL2.V14Ø.COB29OBJ ) SHR .
   ADD DDDEF(COB29SRC ) DA(SMAINT.COBOL2.V14Ø.COB29SRC ) SHR .
   ADD DDDEF(SAMPLIB  ) DA(SMAINT.COBOL2.V14Ø.SAMPLIB  ) SHR .
   ADD DDDEF(PROCLIB  ) DA(SMAINT.COBOL2.V14Ø.PROCLIB  ) SHR .
   ADD DDDEF(HELP     ) DA(SMAINT.COBOL2.V14Ø.HELP     ) SHR .
   ADD DDDEF(SYSLIB)
       CONCAT(SMPMTS,
              COB2CSRC,
              COB2LSRC,
              AMODGEN,
              MODGEN,
              AMACLIB,
              MACLIB)
       .
ENDUCL.
SET BDY(DLIB).          /* DISTRIBUTION ZONE INITIALIZATION */
UCLIN.
   /*********************************************************/
   /* DDDEF FOR DLIB ZONE (SMP)                             */
   /*********************************************************/
   ADD DDDEF(SYSPRINT) SYSOUT(*) .
   ADD DDDEF(SMPOUT  ) SYSOUT(*) .
   ADD DDDEF(SMPLOG  ) DA(SMAINT.COBOL2.V14Ø.SMPLOG) MOD .
   ADD DDDEF(SMPLOGA ) DA(SMAINT.COBOL2.V14Ø.SMPLOGA) MOD .
   ADD DDDEF(SMPMTS  ) DA(SMAINT.COBOL2.V14Ø.SMPMTS) SHR .
   ADD DDDEF(SMPPTS  ) DA(SMAINT.COBOL2.V14Ø.SMPPTS) SHR .
   ADD DDDEF(SMPSTS  ) DA(SMAINT.COBOL2.V14Ø.SMPSTS) SHR .
   ADD DDDEF(SMPSCDS ) DA(SMAINT.COBOL2.V14Ø.SMPSCDS) SHR .
   ADD DDDEF(SMPTLIB ) VOLUME(MNT$Ø2) UNIT(SYSDA) .
   ADD DDDEF(SMPWRK1 ) NEW DELETE CYL SPACE(1Ø,5) DIR(25Ø)
       UNIT(SYSDA) .
   ADD DDDEF(SMPWRK2 ) NEW DELETE CYL SPACE(1Ø,5) DIR(25Ø)
       UNIT(SYSDA) .
   ADD DDDEF(SMPWRK3 ) NEW DELETE CYL SPACE(1Ø,5) DIR(25Ø)
       UNIT(SYSDA) .
   ADD DDDEF(SMPWRK4 ) NEW DELETE CYL SPACE(1Ø,5) DIR(25Ø)
       UNIT(SYSDA) .
   ADD DDDEF(SMPWRK6 ) NEW DELETE CYL SPACE(1Ø,5) DIR(3ØØ)
       UNIT(SYSDA) .
   ADD DDDEF(SYSUT1  ) NEW DELETE CYL SPACE(5,1) UNIT(SYSDA) .
   ADD DDDEF(SYSUT2  ) NEW DELETE CYL SPACE(5,1) UNIT(SYSDA) .
   ADD DDDEF(SYSUT3  ) NEW DELETE CYL SPACE(5,1) UNIT(SYSDA) .
   ADD DDDEF(SYSUT4  ) NEW DELETE CYL SPACE(5,1) UNIT(SYSDA) .
   ADD DDDEF(MACLIB ) DA(SYS1.MACLIB) SHR .
   ADD DDDEF(AMACLIB) DA(SYS1.AMACLIB) SHR .
   ADD DDDEF(MODGEN ) DA(SYS1.MODGEN) SHR .
   ADD DDDEF(AMODGEN) DA(SYS1.AMODGEN) SHR .
   /*********************************************************/
```

```
        /* DDDEF FOR DLIB ZONE                                        */
        /************************************************************/
        ADD DDDEF(COB2CICS ) DA(SMAINT.COBOL2.V140.COB2CICS ) SHR .
        ADD DDDEF(COB2CLIB ) DA(SMAINT.COBOL2.V140.COB2CLIB ) SHR .
        ADD DDDEF(COB2COBJ ) DA(SMAINT.COBOL2.V140.COB2COBJ ) SHR .
        ADD DDDEF(COB2COMP ) DA(SMAINT.COBOL2.V140.COB2COMP ) SHR .
        ADD DDDEF(COB2CSRC ) DA(SMAINT.COBOL2.V140.COB2CSRC ) SHR .
        ADD DDDEF(COB2DOBJ ) DA(SMAINT.COBOL2.V140.COB2DOBJ ) SHR .
        ADD DDDEF(COB2DSRC ) DA(SMAINT.COBOL2.V140.COB2DSRC ) SHR .
        ADD DDDEF(COB2EOBJ ) DA(SMAINT.COBOL2.V140.COB2EOBJ ) SHR .
        ADD DDDEF(COB2ESRC ) DA(SMAINT.COBOL2.V140.COB2ESRC ) SHR .
        ADD DDDEF(COB2LIB  ) DA(SMAINT.COBOL2.V140.COB2LIB  ) SHR .
        ADD DDDEF(COB2LOBJ ) DA(SMAINT.COBOL2.V140.COB2LOBJ ) SHR .
        ADD DDDEF(COB2LSRC ) DA(SMAINT.COBOL2.V140.COB2LSRC ) SHR .
        ADD DDDEF(COB2MLIB ) DA(SMAINT.COBOL2.V140.COB2MLIB ) SHR .
        ADD DDDEF(COB2PLIB ) DA(SMAINT.COBOL2.V140.COB2PLIB ) SHR .
        ADD DDDEF(COB29OBJ ) DA(SMAINT.COBOL2.V140.COB29OBJ ) SHR .
        ADD DDDEF(COB29SRC ) DA(SMAINT.COBOL2.V140.COB29SRC ) SHR .
        ADD DDDEF(SAMPLIB  ) DA(SMAINT.COBOL2.V140.SAMPLIB  ) SHR .
        ADD DDDEF(PROCLIB  ) DA(SMAINT.COBOL2.V140.PROCLIB  ) SHR .
        ADD DDDEF(HELP     ) DA(SMAINT.COBOL2.V140.HELP     ) SHR .
        ADD DDDEF(SYSLIB)
            CONCAT(SMPMTS,
                   COB2CSRC,
                   COB2LSRC,
                   AMODGEN,
                   MODGEN,
                   AMACLIB,
                   MACLIB)
            .
    ENDUCL.
//*
```

ALLOCATE A NEW SET OF COBOL II DATASETS

You then need to allocate new target and distribution libraries.

```
//STEP01   EXEC PGM=IEFBR14
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V140.COB2CICS,
//            UNIT=SYSALLDA,VOL=SER=MNT$02,
//            SPACE=(CYL,(003,01,020)),
//            DCB=(RECFM=U,BLKSIZE=6144,LRECL=0)
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V140.COB2CLIB,
//            UNIT=SYSALLDA,VOL=SER=MNT$02,
//            SPACE=(CYL,(001,01,020)),
//            DCB=(RECFM=FB,BLKSIZE=6160,LRECL=80)
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V140.COB2COBJ,
//            UNIT=SYSALLDA,VOL=SER=MNT$02,
```

```
//            SPACE=(CYL,(Ø1Ø,Ø1,Ø5Ø)),
//            DCB=(RECFM=U,BLKSIZE=6144,LRECL=Ø)
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V14Ø.COB2COMP,
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            SPACE=(CYL,(Ø1Ø,Ø1,Ø5Ø)),
//            DCB=(RECFM=U,BLKSIZE=6144,LRECL=Ø)
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V14Ø.COB2CSRC,
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            SPACE=(CYL,(ØØ5,Ø1,Ø5Ø)),
//            DCB=(RECFM=FB,BLKSIZE=616Ø,LRECL=8Ø)
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V14Ø.COB2DOBJ,
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            SPACE=(CYL,(ØØ1,Ø1,Ø5Ø)),
//            DCB=(RECFM=U,BLKSIZE=6144,LRECL=Ø)
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V14Ø.COB2DSRC,
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            SPACE=(CYL,(ØØ6,Ø1,Ø5Ø)),
//            DCB=(RECFM=FB,BLKSIZE=616Ø,LRECL=8Ø)
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V14Ø.COB2EOBJ,
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            SPACE=(CYL,(ØØ1,Ø1,Ø5Ø)),
//            DCB=(RECFM=U,BLKSIZE=6144,LRECL=Ø)
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V14Ø.COB2LIB,
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            SPACE=(CYL,(Ø12,Ø1,Ø5Ø)),
//            DCB=(RECFM=U,BLKSIZE=6144,LRECL=Ø)
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V14Ø.COB2LOBJ,
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            SPACE=(CYL,(ØØ5,Ø1,Ø5Ø)),
//            DCB=(RECFM=U,BLKSIZE=6144,LRECL=Ø)
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V14Ø.COB2LSRC,
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            SPACE=(CYL,(ØØ3,Ø1,Ø5Ø)),
//            DCB=(RECFM=FB,BLKSIZE=616Ø,LRECL=8Ø)
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V14Ø.COB2MLIB,
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            SPACE=(CYL,(ØØ1,Ø1,Ø5Ø)),
//            DCB=(RECFM=FB,BLKSIZE=616Ø,LRECL=8Ø)
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V14Ø.COB2PLIB,
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            SPACE=(CYL,(Ø1Ø,Ø1,Ø5Ø)),
//            DCB=(RECFM=FB,BLKSIZE=616Ø,LRECL=8Ø)
```

```
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V140.COB290BJ,
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            SPACE=(CYL,(ØØ1,Ø1,Ø5Ø)),
//            DCB=(RECFM=U,BLKSIZE=6144,LRECL=Ø)
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V140.COB29SRC,
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            SPACE=(CYL,(ØØ1,Ø1,Ø5Ø)),
//            DCB=(RECFM=FB,BLKSIZE=616Ø,LRECL=8Ø)
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V140.SAMPLIB,
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            SPACE=(CYL,(ØØ1,Ø1,Ø2Ø)),
//            DCB=(RECFM=FB,BLKSIZE=616Ø,LRECL=8Ø)
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V140.PROCLIB,
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            SPACE=(CYL,(ØØ1,Ø1,Ø2Ø)),
//            DCB=(RECFM=FB,BLKSIZE=616Ø,LRECL=8Ø)
//*
//DD       DD DISP=(,CATLG),DSN=SMAINT.COBOL2.V140.HELP,
//            UNIT=SYSALLDA,VOL=SER=MNT$Ø2,
//            SPACE=(CYL,(ØØ1,Ø1,Ø2Ø)),
//            DCB=(RECFM=FB,BLKSIZE=616Ø,LRECL=8Ø)
```

RECEIVE BUILDMCS MCS AND JCLIN STREAMS IN SMP/E ZONES

At this point, you should receive BUILDMCS output into the new CSI.

```
//STEPØ1   EXEC PGM=GIMSMP,PARM='PROCESS=WAIT',REGION=8M
//*
//SMPCSI   DD DISP=SHR,DSN=SMAINT.COBOL2.V140.GLOBAL.CSI
//*
//SYSPRINT DD SYSOUT=*
//*
//SMPPTFIN DD DISP=SHR,DSN=I990557.SMP.BUILDMCS
//*
//SMPCNTL  DD *
   SET BDY(GLOBAL) .
   RECEIVE SYSMODS
           .
/*
```

The listing shown, in Figures 1a and 1b, shows the result of the RECEIVE processing. During this RECEIVE processing, SMP/E allocates SMPTLIB datasets, as it does during a normal installation.

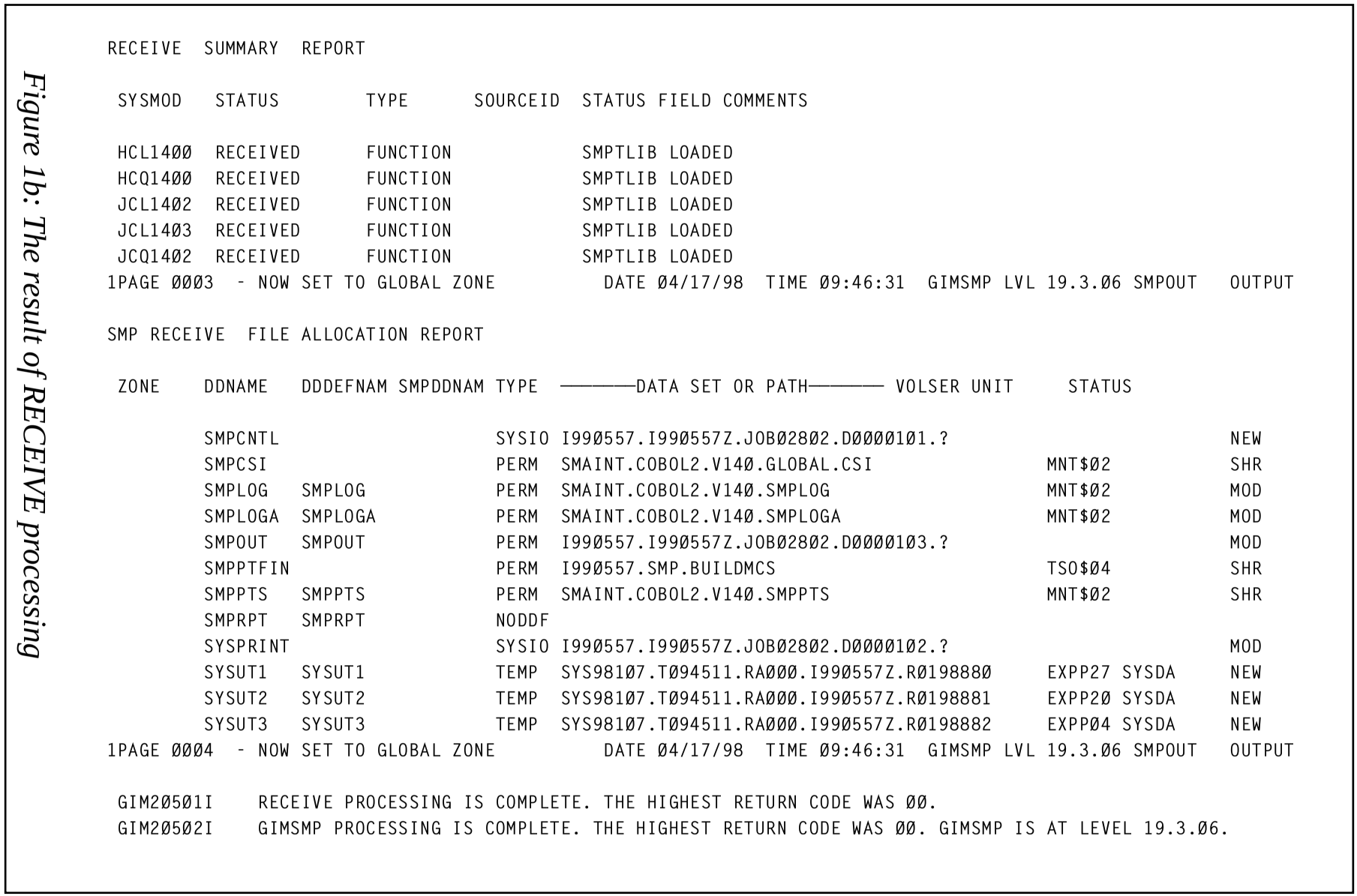
```
1PAGE 0001  - NOW SET TO GLOBAL ZONE          DATE 04/17/98  TIME 09:45:09  GIMSMP LVL 19.3.06 SMPOUT   OUTPUT

GIM42401I   THE FOLLOWING PARAMETERS WERE SPECIFIED ON THE EXEC STATEMENT FOR GIMSMP: 'PROCESS=WAIT'.
   SET BDY(GLOBAL) .
GIM20501I   SET PROCESSING IS COMPLETE. THE HIGHEST RETURN CODE WAS 00.

      RECEIVE SYSMODS
              .

GIM35201I   SMPTLIB SMAINT.COBOL2.V140.HCL1400.F1 WAS ALLOCATED AND CATALOGED ON VOLUME MNT$02.
GIM35201I   SMPTLIB SMAINT.COBOL2.V140.HCL1400.F2 WAS ALLOCATED AND CATALOGED ON VOLUME MNT$02.
GIM39401I   SMPTLIB DATA SETS WERE LOADED FOR SYSMOD HCL1400.
GIM22701I   RECEIVE PROCESSING WAS SUCCESSFUL FOR SYSMOD HCL1400.
GIM35201I   SMPTLIB SMAINT.COBOL2.V140.HCQ1400.F1 WAS ALLOCATED AND CATALOGED ON VOLUME MNT$02.
GIM35201I   SMPTLIB SMAINT.COBOL2.V140.HCQ1400.F2 WAS ALLOCATED AND CATALOGED ON VOLUME MNT$02.
GIM39401I   SMPTLIB DATA SETS WERE LOADED FOR SYSMOD HCQ1400.
GIM22701I   RECEIVE PROCESSING WAS SUCCESSFUL FOR SYSMOD HCQ1400.
GIM35201I   SMPTLIB SMAINT.COBOL2.V140.JCL1402.F1 WAS ALLOCATED AND CATALOGED ON VOLUME MNT$02.
GIM39401I   SMPTLIB DATA SETS WERE LOADED FOR SYSMOD JCL1402.
GIM22701I   RECEIVE PROCESSING WAS SUCCESSFUL FOR SYSMOD JCL1402.
GIM35201I   SMPTLIB SMAINT.COBOL2.V140.JCL1403.F1 WAS ALLOCATED AND CATALOGED ON VOLUME MNT$02.
GIM35201I   SMPTLIB SMAINT.COBOL2.V140.JCL1403.F2 WAS ALLOCATED AND CATALOGED ON VOLUME MNT$02.
GIM39401I   SMPTLIB DATA SETS WERE LOADED FOR SYSMOD JCL1403.
GIM22701I   RECEIVE PROCESSING WAS SUCCESSFUL FOR SYSMOD JCL1403.
GIM35201I   SMPTLIB SMAINT.COBOL2.V140.JCQ1402.F1 WAS ALLOCATED AND CATALOGED ON VOLUME MNT$02.
GIM39401I   SMPTLIB DATA SETS WERE LOADED FOR SYSMOD JCQ1402.
GIM22701I   RECEIVE PROCESSING WAS SUCCESSFUL FOR SYSMOD JCQ1402.
1PAGE 0002  - NOW SET TO GLOBAL ZONE          DATE 04/17/98  TIME 09:46:30  GIMSMP LVL 19.3.06 SMPOUT   OUTPUT
```

Figure 1a: The result of RECEIVE processing



```
RECEIVE  SUMMARY  REPORT

 SYSMOD   STATUS        TYPE      SOURCEID  STATUS FIELD COMMENTS

 HCL1400  RECEIVED      FUNCTION            SMPTLIB LOADED
 HCQ1400  RECEIVED      FUNCTION            SMPTLIB LOADED
 JCL1402  RECEIVED      FUNCTION            SMPTLIB LOADED
 JCL1403  RECEIVED      FUNCTION            SMPTLIB LOADED
 JCQ1402  RECEIVED      FUNCTION            SMPTLIB LOADED
1PAGE 0003  - NOW SET TO GLOBAL ZONE          DATE 04/17/98  TIME 09:46:31  GIMSMP LVL 19.3.06 SMPOUT   OUTPUT

SMP RECEIVE  FILE ALLOCATION REPORT

 ZONE    DDNAME   DDDEFNAM SMPDDNAM TYPE  ————————DATA SET OR PATH———————— VOLSER UNIT     STATUS

         SMPCNTL                   SYSIO I990557.I990557Z.JOB02802.D0000101.?                          NEW
         SMPCSI                    PERM  SMAINT.COBOL2.V140.GLOBAL.CSI               MNT$02           SHR
         SMPLOG   SMPLOG           PERM  SMAINT.COBOL2.V140.SMPLOG                  MNT$02           MOD
         SMPLOGA  SMPLOGA          PERM  SMAINT.COBOL2.V140.SMPLOGA                 MNT$02           MOD
         SMPOUT   SMPOUT           PERM  I990557.I990557Z.JOB02802.D0000103.?                         MOD
         SMPPTFIN                  PERM  I990557.SMP.BUILDMCS                       TSO$04           SHR
         SMPPTS   SMPPTS           PERM  SMAINT.COBOL2.V140.SMPPTS                  MNT$02           SHR
         SMPRPT   SMPRPT           NODDF
         SYSPRINT                  SYSIO I990557.I990557Z.JOB02802.D0000102.?                          MOD
         SYSUT1   SYSUT1           TEMP  SYS98107.T094511.RA000.I990557Z.R0198880   EXPP27 SYSDA     NEW
         SYSUT2   SYSUT2           TEMP  SYS98107.T094511.RA000.I990557Z.R0198881   EXPP20 SYSDA     NEW
         SYSUT3   SYSUT3           TEMP  SYS98107.T094511.RA000.I990557Z.R0198882   EXPP04 SYSDA     NEW
1PAGE 0004  - NOW SET TO GLOBAL ZONE          DATE 04/17/98  TIME 09:46:31  GIMSMP LVL 19.3.06 SMPOUT   OUTPUT

 GIM20501I    RECEIVE PROCESSING IS COMPLETE. THE HIGHEST RETURN CODE WAS 00.
 GIM20502I    GIMSMP PROCESSING IS COMPLETE. THE HIGHEST RETURN CODE WAS 00. GIMSMP IS AT LEVEL 19.3.06.
```

Figure 1b: The result of RECEIVE processing

APPLY COBOL II FMIDS IN THE NEWSMP/E ZONES

You then need to apply the selected FMIDs in the new SMP/E environment.

```
//STEPØ1   EXEC PGM=GIMSMP,PARM='PROCESS=WAIT',REGION=8M
//*
//SMPCSI   DD DISP=SHR,DSN=SMAINT.COBOL2.V14Ø.GLOBAL.CSI
//*
//SYSPRINT DD SYSOUT=*
//*
//SMPCNTL  DD *
   SET BDY(TARGET) .
       APPLY
                 S(
                   HCL14Ø0 ,
                   HCQ14Ø0 ,
                   JCL14Ø2 ,
                   JCL14Ø3 ,
                   JCL14Ø4 ,
                   JCL14Ø6 ,
                   JCL14Ø7 ,
                   JCQ14Ø2 ,
                   )
                 .
/*
```

ACCEPT COBOL II FMIDS IN THE NEWSMP/E ZONES

To finish, you should accept the FMIDs.

```
//STEPØ1   EXEC PGM=GIMSMP,PARM='PROCESS=WAIT',REGION=8M
//*
//SMPCSI   DD DISP=SHR,DSN=SMAINT.COBOL2.V14Ø.GLOBAL.CSI
//*
//SYSPRINT DD SYSOUT=*
//*
//SMPCNTL  DD *
   SET BDY(DLIB) .
       ACCEPT
                 S(
                   HCL14Ø0 ,
                   HCQ14Ø0 ,
                   JCL14Ø2 ,
                   JCL14Ø3 ,
                   JCL14Ø4 ,
                   JCL14Ø6 ,
                   JCL14Ø7 ,
                   JCQ14Ø2 ,
                   )
                 .
/*
```

Finally you get, in your new SMP/E environment, the exact copy of the original product.

USAGE RESTRICTIONS

The BUILDMCS command is not intended to be used with all products. It should not be used with products that have 'intersections' (shared modules or common elements) with other products. This is because SMP/E does not have enough information to correctly create a corresponding superseding SYSMOD function.

RELATED PUBLICATIONS

The publication, *SC28-1805 OS/390 SMP/E Commands* will provide useful additional information regarding the BUILDMCS command.

Patrick Renard
CTRNE (France)

Dispay WTORs in TSO

INTRODUCTION

In *MVS Update Issue 142* (June 1998), in the article *Timed job submission* we considered an edit macro that could be used in place of SUBMIT. The following utility makes use of code from that article. Other prerequisites include; MVS/ESA Version 5, TSO/E, JES2, ISPF Version 3, and above.

THE PROBLEM

Currently our Year 2000 development is being run on an independent MVS system, which to all intents and purposes is the sole domain of the developers. In order to make them as self-reliant as possible, it was necessary to provide them with access to some operator facilities – particularly (as far as this article is concerned) the ability to conveniently issue operator commands, and also to see outstanding WTORs.

THE SOLUTION

As a result I decided to see if I could provide a solution that did not require access to consoles, and did not require any software purchases. The following dialog was the result. All the user has to do is issue the WTOLISTR command from the command line and, assuming there are some outstanding WTORs lying around, a screen similar to the following should appear:

```
                  WTOR Information                          Row 1 to 2 of 2
Command ===>                                                Scroll ===> PAGE

Current number of WTOR's outstanding ===> 2

Enter command REPLY if you wish to either respond to a message
or you wish to enter any other operator command.

Type FLIP if you wish to see any more of the message.

JOBNAME  JOBID    USERID   TIME      MESSAGE
PRODVSV  STC17662 PRODVSV  21.21.46  *12 VSVNMF0170I REPLY "STOP" TO TERMINA
NETVIEW  STC17949 NETVIEW  21.25.09  *08 DSI802A CNM50   REPLY WITH VALID NC
****************************** Bottom of data ******************************
```

Given that messages are often longer than can be fitted on a standard screen, by typing FLIP it is possible to switch to an alternative layout, which provides full access to the message text as follows:

```
                  WTOR Information                          Row 1 to 2 of 2
Command ===>                                                Scroll ===> PAGE

Current number of WTOR's outstanding ===> 2

Enter command REPLY if you wish to either respond to a message
or you wish to enter any other operator command.

Type FLIP if you wish to see any more of the message.

MESSAGE
*12 VSVNMF0170I REPLY "STOP" TO TERMINATE PRODUCT
*08 DSI802A CNM50   REPLY WITH VALID NCCF SYSTEM OPERATOR COMMAND
******************************** Bottom of data ****************************
```

Once switched, this display form will remain in place until another FLIP command is issued, or the dialog is re-initiated. Apart from just providing the display mechanism, it is possible for the user to reply to commands provided the REXX function REXWTO (as documented

in the article *Timed job submission*) is installed in your TSO STEPLIB concatenation.

Installation of this function is straightforward. Simply install the panels, REXX and code in your ISPPLIB, SYSPROC (or SYSEXEC) and STEPLIB as appropriate. Note that the REXX function is not re-entrant and can only be used in TSO. Note also that in order to function this code will require a dynamic APF on and off SVC. Just change the SVCAUTH and SVCDAUTH equates as appropriate. Should you not have these, please use the examples that are also documented in the *Timed job submission* article.

THE MACROS

Before providing the code as such, you will need to make the following macros available to the assembly for WTOLIST.

REXREGS

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

SHOW

```
        MACRO
        SHOW &LABEL,&ASNAME,&ERR=ABENDØØ1,&LEN=,&UNPACK=NO
*********************************************************************
*
* MACRO FORMAT:
*       SHOW &LABEL,&ASNAME,&ERR=,&LEN=,&UNPACK=
* WHERE:
*       &LABEL IS THE NAME OF THE LABEL THAT ADDRESSES THE FIELD FROM
*              WHERE THE DATA TO BE DEFINED IN A REXX VARIABLE IS
*              LOCATED
*       &ASNAME IS THE NAME TO BE ASSIGNED TO THE DATA FOR USE IN REXX
*       &ERR= IS THE LABEL TO BRANCH TO SHOULD AN ERROR OCCUR WHILE
*             CREATING THE REXX VARIABLE. BY DEFAULT IT IS ABENDØØ1
*       &LEN= IF THE DATA AT &LABEL IS NOT DEFINED SUCH THAT THE LENGTH
```

```
*             OF THE DATA IS WHAT YOU WANT, SIMPLY ENTER A NUMBER HERE
*             THAT DEFINES THE LENGTH REQUIRED. CAN ALSO BE USEFUL IF
*             NECESSARY TO DUMP OUT A LARGE AREA.
*       &UNPACK= IF THE DATA IS IN PACKED FORMAT, SET THIS TO YES IF
*             YOU WANT THE AREA UNPACKED FOR YOU. THE DEFAULT IS NO.
*
*********************************************************************
        SHOWSET
LITLOC  LOCTR
&LABCHECK SETC '@_&ASNAME'
         AIF   (D'&LABCHECK).BYPASS
@_&ASNAME DC C'&ASNAME'
.BYPASS ANOP
&SYSECT LOCTR
        LA 1,@_&ASNAME
        ST 1,SHVNAMA
        LA 1,L'@_&ASNAME
        ST 1,SHVNAML
        AIF ('&UNPACK' EQ 'NO').DATAAOK
        UNPK @_UNPACK,&LABEL
        OI @_UNPACK+(L'@_UNPACK-1),X'FØ'
        LA 1,@_UNPACK
        ST 1,SHVVALA
        LA 1,L'@_UNPACK
        AGO .OK
.DATAAOK ANOP
        LA 1,&LABEL
        ST 1,SHVVALA
        AIF (T'&LEN NE 'O').DOLEN
        LA 1,L'&LABEL
        AGO .OK
.DOLEN  ANOP
        LA 1,&LEN
.OK     ANOP
        ST 1,SHVVALL
        LR Ø,1Ø
        LA 1,COMS
        L 15,IRXEXCOM
        BALR 14,15
        LTR 15,15
        BNZ &ERR
        MEND
```

SHOWARAY

```
        MACRO
        SHOWARAY &LABEL,&ASNAME,&ERR=ABENDØØ1,&LEN=,&SUBARRAY=,&DEBIN=
*********************************************************************
*
* MACRO TO CREATE REXX ARRAY VARIABLES
*
```

```
* NOTE RESTRICTION: THIS MACRO IS LIMITED TO CREATING UP TO 9,999,999
*                   ENTRIES FOR EACH ARRAY.
*
* MACRO FORMAT:
*       SHOWARAY &LABEL,&ASNAME,&ERR=,&LEN=,&SUBARRAY=,&DEBIN=
* WHERE:
*       &LABEL IS THE NAME OF THE LABEL THAT ADDRESSES THE FIELD FROM
*              WHERE THE DATA TO BE DEFINED IN A REXX VARIABLE IS
*              LOCATED
*       &ASNAME IS THE NAME TO BE ASSIGNED TO THE DATA FOR USE IN REXX
*       &ERR= IS THE LABEL TO BRANCH TO SHOULD AN ERROR OCCUR WHILE
*             CREATING THE REXX VARIABLE. BY DEFAULT IT IS ABEND001
*       &LEN= IF THE DATA AT &LABEL IS NOT DEFINED SUCH THAT THE LENGTH
*             OF THE DATA IS WHAT YOU WANT, SIMPLY ENTER A NUMBER HERE
*             THAT DEFINES THE LENGTH REQUIRED. CAN ALSO BE USEFUL IF
*             NECESSARY TO DUMP OUT A LARGE AREA.
*       &SUBARRAY= IF A MULTI LEVEL ARRAY IS REQUIRED EG A.1.1 THEN
*                  SET THIS VALUE ACCORDINGLY.
*       &DEBIN= IF THE DATA TO BE CREATED IS BINARY, SETTING THIS TO A
*               VALUE WILL CONVERT THE SPECIFIED NUMBER OF BYTES FROM
*               BINARY TO CHARACTER. THE DEFAULT LENGTH FOR THE
*               OUTPUT DATA IS 4 BYTES. IF THIS IS INSUFFICIENT, THEN
*               SPECIFY A SUITABLE &LEN VALUE TO OVERRIDE IT.
*
*********************************************************************
        AIF (T'&DEBIN EQ 'O').NOLCS
        LCLC &ARRAY(4)
        LCLA &DEFLEN
&DEFLEN SETA 4   * SET THE DEFAULT OUTPUT LENGTH TO 4 (BIN CONVERSION)
&ARRAY(1) SETC '1'
&ARRAY(2) SETC '3'
&ARRAY(3) SETC '7'
&ARRAY(4) SETC '15'
.NOLCS  ANOP
        SHOWSET
LITLOC  LOCTR
&LABCHECK SETC '@_&ASNAME&SUBARRAY'
         AIF   (D'&LABCHECK).BYPASS
         AIF   (T'&SUBARRAY EQ 'O').NORMNAME
&LABCHECK DC C'&ASNAME..&SUBARRAY'
         AGO   .EOFARRAY
.NORMNAME ANOP
&LABCHECK DC C'&ASNAME'
.EOFARRAY ANOP
&LABCHECK._ARRAY DC C'.       '
&LABCHECK._COUNTER DC PL4'0' * COUNTER FIELD FOR THIS ITEM
         AIF   (D'@_UNPACKER).BYPASS
@_UNPACKER DC CL8' ' * UNPACK FIELD FOR COUNTER
          DC CL8' '         * BLANKS FOR FILL DETAILS
@_DWORD  DS CL8 * 8 BYTE WORKAREA FOR BINARY CONVERSION
.BYPASS  ANOP
&SYSECT LOCTR
```

```
        AP &LABCHECK._COUNTER,=P'1' * INCREMENT THE COUNTER THIS PASS
        UNPK @_UNPACKER,&LABCHECK._COUNTER * UNPACK THE VALUE
        OI   @_UNPACKER+7,X'FØ'       * REMOVE THE SIGN
* NOW NEED TO WORK OUT THE LENGTH OF THE COUNTER BIT TO ADD TO ARRAY
        L    R15,&LABCHECK._COUNTER * LOAD THE COUNTER VALUE TO WORK
*                                     OUT THE LENGTH
        SRL  R15,4                 * REMOVE THE SIGN
        XR   R14,R14               * CLEAR R14 FOR A COUNTER
LOOP&SYSNDX DS ØH
        SRA  R15,4                 * MOVE DIGIT BY DIGIT
        LTR  R15,R15
        BZ   COUNT&SYSNDX
        LA   R14,1(,R14)
        B    LOOP&SYSNDX
COUNT&SYSNDX DS ØH
* NOW ADD COUNT FIELD TO NAME
        LA  R15,@_UNPACKER+7        * POINT TO END OF FIELD
        SR  R15,R14                * AND COME BACK TO FIRST DIGIT.
        MVC &LABCHECK._ARRAY+1(7),Ø(R15)
        LA 1,&LABCHECK
        ST 1,SHVNAMA
* NOW CALCULATE NEW LENGTH
        LA 1,L'&LABCHECK
        LA 1,2(R14,R1)
        ST 1,SHVNAML
        AIF (T'&DEBIN EQ 'O').NORMLAB
*
*** NOW ALLOW FOR A BINARY CONVERSION
*
        XR R15,R15
        ICM R15,&ARRAY(&DEBIN),&LABEL * LOAD THE BINARY VALUE
        CVD R15,@_DWORD               * CONVERT TO PACKED
        OI  @_DWORD+7,X'ØF'
        UNPK @_UNPACKER,@_DWORD
*
*** IF THE LEN VALUE IS SUPPLIED THIS OVERRIDES THE DEFAULT OF 4
*
        AIF (T'&LEN EQ 'O').SETDEF  * LENGTH NOT SUPPLIED USE DEFLEN
&DEFLEN SETA &LEN                   * RESET DEFLEN TO SUPPLIED LEN
.SETDEF ANOP
        LA R1,@_UNPACKER+(8-&DEFLEN)
        ST R1,SHVVALA
        LA R1,&DEFLEN
        AGO .OK
.NORMLAB ANOP
        LA 1,&LABEL
        ST 1,SHVVALA
        AIF (T'&LEN NE 'O').DOLEN
        LA 1,L'&LABEL
        AGO .OK
```

```
.DOLEN  ANOP
        LA 1,&LEN
.OK     ANOP
        ST 1,SHVVALL
        LR Ø,1Ø
        LA 1,COMS
        L 15,IRXEXCOM
        BALR 14,15
        LTR 15,15
        BNZ &ERR
        MEND
```

SHOWBASE

```
        MACRO
        SHOWBASE &LABEL,&ERR=ABENDØØ1,&SUBARRAY=
*********************************************************************
*
* MACRO TO CREATE REXX BASE VARIABLES
* SHOULD BE USED IN ASSOCIATION WITH A SHOWARAY MACRO. NOTE THAT A
* SHOWBASE MACRO IS OPTIONAL IF YOU ALREADY KNOW THE NUMBER OF
* VARAIABLES BEING SET. THIS WILL CREATE THE A.Ø ENTRY
*
* MACRO FORMAT:
*       SHOWBASE &LABEL,&ERR=,&SUBARRAY=
* WHERE:
*       &LABEL IS THE NAME OF THE REXX ARRAY LABEL WHICH HAS BEEN
*              CREATED. THIS WILL CREATE THAT LABEL.Ø ENTRY
*       &ERR= IS THE LABEL TO BRANCH TO SHOULD AN ERROR OCCUR WHILE
*             CREATING THE REXX VARIABLE. BY DEFAULT IT IS ABENDØØ1
*       &SUBARRAY= IF SUBARRAYS HAVE BEEN USED THIS WILL INSERT THE
*                  APPROPRIATE VALUE EG A.1.Ø
*
*********************************************************************
        SHOWSET
        AIF (T'&SUBARRAY EQ 'O').NORMNAME
&ASNAME   SETC '&LABEL..&SUBARRAY..Ø'
        AGO .CHECKER
.NORMNAME ANOP
&ASNAME   SETC '&LABEL..Ø'
.CHECKER  ANOP
&LABCHECK SETC '@_&LABEL&SUBARRAY._COUNTER'
        AIF (D'&LABCHECK).ITSOK
MNOTE   NO ARRAY ELEMENTS DEFINED.
        MEXIT
.ITSOK  ANOP
LITLOC  LOCTR
@_A&SYSNDX DC C'&ASNAME'
        AIF (D'@_UNPACKER).BYPASS
@_UNPACKER DC CL8' '
```

```
.BYPASS ANOP
&SYSECT LOCTR
        LA 1,@_A&SYSNDX
        ST 1,SHVNAMA
        LA 1,L'@_A&SYSNDX
        ST 1,SHVNAML
        LA  1,L'&LABCHECK
        BCTR 1,Ø
        LA  15,&LABCHECK
        LA  1,Ø(15,1)
        OI  Ø(1),X'ØF'
        UNPK @_UNPACKER,&LABCHECK
        LA 1,@_UNPACKER
        ST 1,SHVVALA
        LA 1,L'@_UNPACKER
        ST 1,SHVVALL
        LR Ø,1Ø
        LA 1,COMS
        L 15,IRXEXCOM
        BALR 14,15
        LTR 15,15
        BNZ &ERR
        MEND
```

SHOWSET

```
        MACRO
        SHOWSET
        AIF (D'SHOW_START).NONEED
        B  BY_SHOW_START
SHOW_START DS ØH
        ST R1Ø,COMRET
        LA 6,COMSHVB
        USING SHVBLOCK,R6
        XC COMSHVB(SHVBLEN),COMSHVB
        XC SHVNEXT,SHVNEXT
        MVI SHVCODE,C'S'
        BR 14
BY_SHOW_START DS ØH
LITLOC  LOCTR
@_UNPACK DC CL16' '
&SYSECT LOCTR
.NONEED ANOP
        BAL 14,SHOW_START
        MEND
```

WTOLIST

```
*********************************************************************
*
* WTOLIST: A REXX FUNCTION TO LIST OUT WTOR DETAILS
*
* USAGE: CALL WTOLIST
*
* NOTE:  WTOLIST WILL RETURN THE FOLLOWING INFORMATION:
*        JOB_NAME.X ............ JOB NAME OF WTOR ISSUER
*        JOB_ID.X .............. JOB ID OF WTOR ISSUER
*        USER_ID.X ............. RACF USER ID OF JOB SUBMITTER
*        REPLY_ID.X ............ REPLY NUMBER FOR MESSAGE
*        XMEM_DATA ............. COULD BE YES OR NO. IF YES THEN
*                                THE ACTUAL WTOR MESSAGE WILL ALSO
*                                BE RETRIEVED
*        WTOR_MESSAGE.X ........ ASSOCIATED MESSAGE DATA.
*        RC .................... COULD BE ZERO OR 4. IF 4 THEN NO
*                                DATA AT ALL IS RETURNED.
*
* NOTE ALSO THAT THIS PROGRAM HAS A CUT-OUT OF 1ØØ MAXIMUM WTOR
* ENTRIES JUST IN CASE.
*
*********************************************************************
WTOLIST TITLE 'REXX FUNCTION TO RETRIEVE WTOR INFORMATION'
WTOLIST  AMODE 31
WTOLIST  RMODE ANY
WTOLIST  CSECT
SVCAUTH  EQU 235     <=== set to your apf on SVC
SVCDAUTH EQU 236     <=== set to your apf off SVC
         REXREGS
         PRINT GEN
         BAKR 14,Ø
         LR   12,15
         USING WTOLIST,12
         PRINT GEN
         LR    R1Ø,RØ                 *R1Ø -> A(ENVIRONMENT BLOCK)
         USING ENVBLOCK,R1Ø
         L     R9,ENVBLOCK_IRXEXTE    *R9 -> A(EXTERNAL EP TABLE)
                     USING IRXEXTE,R9
*
* GET A WORK AREA FOR REXX OUTPUT
* MAP WITH R2 ... NEED TO DO THIS BEFORE ANY ROUTING TO POSSIBLE
* REXX VARIABLE OUTPUT (EG ROUTINE ABENDØØ1)
*
         STORAGE OBTAIN,LENGTH=AREALEN,ADDR=(2),LOC=BELOW
*
         USING WORKAREA,2
*
* PREPARE THE REXX AREA FOR USE
*
         XC  COMS(COMSLEN),COMS * SET TO LOW VALUES
         LA  15,COMID
         ST  15,COMS
```

```
         LA  15,COMDUMMY
         ST  15,COMS+4
         ST  15,COMS+8
         LA  15,COMSHVB
         ST  15,COMS+12
         LA  15,COMRET
         ST  15,COMS+16
         OI  COMS+16,X'8Ø'
         MVC COMID,=C'IRXEXCOM'
*
*** LOCATE THE CVT
*
         XR   3,3
         USING PSA,3
         L    3,FLCCVT
         USING CVT,3
*
* NOW FIND THE UCM
*
         L    3,CVTCUCB
         USING UCM,3
*
* NOW COMMENCE CHAINING THROUGH THE ORE'S
* FIRST CHECK IF THERE ARE ANY THERE (RPYQ=Ø MEANS NONE).
*
         CLC   UCMRPYQ,=F'Ø'
         BE    SETRC4
*
*** ASSUMING THERE ARE MESSAGE, LETS GO HAVE A NOSEY IN THE CONSOLE
*** ADDRESS SPACE AND SEE WHAT THEY ARE
*
         L     R4,UCMASCB    * GET THE CONSOLE ADDRESS SPACE ASCB
         USING ASCB,R4       * AND MAP IT
         L     R4,ASCBASSB   * NOW GET THE ASSB FOR THE TOKEN
         USING ASSB,R4
*
         L     R3,UCMRPYQ
         USING OREF,3
*
* WILL NEED TO BE AUTHORIZED TO RETRIEVE DATA
*
         SVC SVCAUTH
*
         MODESET KEY=ZERO,MODE=SUP
*
         ALESERV ADD,ALET=ASALET,STOKEN=ASSBSTKN,CHKEAX=NO
*
         LTR 15,15 * DID THE LINK TO CONSOLE WORK?
         BZ  SETYES *   YES TO SO AN OK FLAG FOR RETRIEVING DATA.
*
```

```
         MVI YESFLAG,C'N' * OTHERWISE SET NO.
         B   PREP_LOOP
*
SETYES   DS ØH
         MVI YESFLAG,C'Y' * SET YES
         LAM  RØ,R15,FULL_WORDS * AND ENSURE ACCESS REGISTER PREPARED
         LAM  R8,R8,ASALET      * AND USE 8 FOR ACCESS
*
PREP_LOOP  DS ØH
         LA    R4,1ØØ   * SET A BREAKOUT COUNTER
*
LOOP     DS   ØH
         LOCASCB ASID=OREASID  * FIND THE ASCB FOR THE ASID
*
         LTR 15,15
         BNZ SKIPIT
         LR  7,1        * SWAP R1 TO R7 FOR MAPPING PURPOSES
         USING ASCB,R7
         IAZXJSAB READ,ASCB=(7),JOBID=JOBIDIT,JOBNAME=THISJOB,         X
               USERID=THISUSER
*
         LTR 15,15      * NO DETAILS FOUND?
         BNZ SKIPIT
*
         SHOWARAY THISJOB,JOB_NAME
         SHOWARAY JOBIDIT,JOB_ID
         SHOWARAY THISUSER,USER_ID
         SHOWARAY OREID,REPLY_ID
*
         CLI YESFLAG,C'Y'      * IS CROSS MEMORY POSSIBLE?
         BNE SKIPIT
*
***      CROSS MEMORY IS POSSIBLE, SO LETS GET THE INFO
*
         XC WTORINFO,WTORINFO   * CLEAR OUT THE RETURN WORK AREA
*
         L  R8,ORERWQE          * NOW ADDRESS THE COSOLE WTOR
*
         SAC 512                * ACTIVATE CROSS MEMORY
         MVC WTORINFO,32(8)     * THE ACTUAL WTOR IS 32 BYTES IN
         SAC Ø
*
         SHOWARAY WTORINFO,WTOR_MESSAGE
*
SKIPIT   DS ØH
         CLC  ORELKP,=F'Ø' * ALL DONE
         BE   SETBASE      * YES SO END JOB CODE Ø
*
         L    3,ORELKP
         BCT 4,LOOP
```

```
*
*** IF ALL THE ENTRIES HAVE BEEN DISPLAYED, USE JOBNAME AS A BASE
*
SETBASE  DS ØH
*
         SHOWBASE JOB_NAME
*
*** SET A COMPLETION RC OF Ø
*
         SHOW RCØ,RC
*
         CLI YESFLAG,C'Y'
         BNE NOXMEM
*
         SHOW YES,XMEM_DATA
*
         ALESERV DELETE,ALET=ASALET
*
         B CLEAN_UP
*
NOXMEM   DS ØH
*
         SHOW NO,XMEM_DATA
*
CLEAN_UP DS ØH
*
         MODESET KEY=NZERO,MODE=PROB
         SVC SVCDAUTH
*
RETURN   DS ØH
*
         STORAGE RELEASE,LENGTH=AREALEN,ADDR=(2)
         PR
*
SETRC4   DS ØH
*
*** SET A COMPLETION RC OF 4 NO ORES EXIST.
*
         SHOW RC4,RC
         B RETURN
*
ABENDØØ1 DS    ØH
         ABEND 1
*
*********************************************************************
***      WORKING STORAGE ETC                                      ***
*********************************************************************
*
         TITLE  'WORKING STORAGE / DSECTS'
         LTORG
RCØ      DC C'Ø'
RC4      DC C'4'
```

```
YES      DC C'YES'
NO       DC C'NO'
FULL_WORDS DC 16F'Ø'
*
WORKAREA DSECT
*
*      IRXEXCOM PARAMETER AREA
*
         DS  ØD
COMS     DS  5AL4
COMID    DS  CL8
COMDUMMY DS  AL4              * NOT USED
COMSHVB  DS  (SHVBLEN)X       * IRXEXCOM SHVBLOCK (LENGTH FROM DSECT)
COMRET   DS  AL4              * IRXECOM RC
         DS  ØD
COMSLEN  EQU *-COMS
JOBIDIT  DS D
THISJOB  DS D
THISUSER DS D
ASALET   DS F
WTORINFO DS CL4Ø
YESFLAG  DS C
AREALEN  EQU *-WORKAREA
         IHAPSA
         CVT DSECT=YES
         IAZJSAB
         IHAORE
         IEECUCM
         IHAASCB
         IHAASSB
         IHAASVT
         IARRCE
         IRXEFPL
         IRXARGTB
         IRXEVALB
         IRXENVB
         IRXEXTE
         IRXSHVB
         END
```

WTOLISTR

The following is issued to invoke the WTOR display:

```
/* REXX */
/* */
/* Prepare a table for display purposes */
default_panel=WTOLSTP1 /* set default display format */
ADDRESS ISPEXEC
looper:
'TBCREATE WTO NAMES(jobname jobid userid time message),
```

```
NOWRITE REPLACE'
/* */
/* Call Assembler support routine to obtain relevant information */
/* about the WTOR situation                                      */
/* */
CALL WTOLIST
/* */
/* Now loop around to list all the WTO items in table form */
/* */
IF RC=4 THEN DO
   zedsmsg='No outstanding WTORS'
   zedlmsg='There are no WTOR"s to display'
   'SETMSG MSG(ISRZ00Ø1)'
   EXIT
   END
numwtor=STRIP(job_name.Ø,'L','Ø')
DO x=1 TO job_name.Ø
jobname=job_name.x
jobid=job_id.x
userid=user_id.x
IF xmem_data='YES' THEN DO /* if the message has also been retrieved */
   message=wtor_message.x
   time=wtor_time.x
   END
ELSE message='Cross memory failure'
'TBADD WTO'
END
'TBTOP WTO'
'TBDISPL WTO PANEL('default_panel')'
IF reply='END' then EXIT
UPPER zcmd
IF zcmd='REPLY' THEN CALL response_panel
IF zcmd='FLIP' THEN DO
  IF default_panel='WTOLSTP1' THEN default_panel='WTOLSTP2'
  ELSE default_panel='WTOLSTP1'
  END
SIGNAL looper
response_panel:
zwinttl='Enter command to be issued'
'ADDPOP ROW(1) COLUMN(1)'
'DISPLAY PANEL(WTOREP1)'
'REMPOP'
IF reply='END' THEN RETURN
CALL REXWTO literal
RETURN
```

WTOLISTP1

The following is the initial panel displayed:

```
)Attr Default(%+_)
   ! type(output) intens(high) caps(on ) just(left )
   @ type(output) intens(low ) caps(off) just(asis )
)Body  Expand(//)
/ /%WTOR Information for lpar / /
%Command ===>_zcmd                                / /%Scroll ===>_amt +
+
+Current number of WTORs outstanding ===>!numwtor+
+
+Enter command%REPLY+ if you wish to either respond to a message
+or you wish to enter any other operator command.
+
+Type%FLIP+if you wish to see more of the message.
+
JOBNAME  JOBID    USERID    TIME      MESSAGE
)Model
!z       !z       !z        !z        !z
)Init
  .HELP = WTOLSTH1
  .ZVARS = '(jobname jobid userid time message)'
  &amt = PAGE
)PROC
&REPLY = .RESP
)End
```

WTOLSTP2

This is the panel displayed upon the first use of the FLIP command:

```
)Attr Default(%+_)
   ! type(output) intens(high) caps(on ) just(left )
   @ type(output) intens(low ) caps(off) just(asis )
)Body  Expand(//)
/ /%WTOR Information for lpar / /
%Command ===>_zcmd                                / /%Scroll ===>_amt +
+
+Current number of WTOR's outstanding ===>!numwtor+
+
+Enter command%REPLY+ if you wish to either respond to a message
+or you wish to enter any other operator command.
+
+Type%FLIP+if you wish to more information about the message.
+
 MESSAGE
)Model
!z
)Init
```

```
  .HELP = WTOLSTH2
  .ZVARS = '(message)'
  &amt = PAGE
)PROC
&REPLY = .RESP
)End
```

WTOLSTH1

This is the HELP panel for the primary panel:

```
)BODY
'──────── Help Panel For WTOR list ─────────────────
+
+This panel displays all the outstanding WTOR's on the system.
+
+The display consists of 5 columns.
+Column 1, JOBNAME is the name of the job issuing the WTOR.
+Column 2, JOBID is the JES2 job number for the issuer.
+Column 3, USERID is the RACF userid for the issuer.
+Column 4, TIME is the time the WTOR was issued.
+Column 5, MESSAGE is the actual WTOR. Note only part of the message is
+          shown because of screen sizes. Please type%FLIP+to use the
+          alternative screen display to see more of the message.
+
+If you wish to respond to any of the messages, or if you just wish to
+enter any other operator command, then simply type REPLY on the command
+line and a pop-up panel will be shown to allow a command to be entered.
+Note though that you will need the necessary RACF authority to actually
+be able to get the command to work.
)PROC
.help=isp00004
)END
```

WTOLSTH2

This is the HELP panel for the alternate FLIP panel:

```
)BODY
'──────── Help Panel For WTOR list ─────────────────
+
+This panel displays all the outstanding WTOR's on the system.
+
+The display lists out all the outstanding WTORS on the system.
+
+To see who issued the messages and when, type%FLIP+for the alternative
+screen layout.
+
```

```
+If you wish to respond to any of the messages, or if you just wish to
+enter any other operator command, then simply type REPLY on the command
+line and a pop-up panel will be shown to allow a command to be entered.
+Note though that you will need the necessary RACF authority to actually
+be able to get the command to work.
)PROC
.help=isp00004
)END
```

WTOREP1

This is the panel for issuing commands:

```
)Attr Default(%+_)
)Body Window(74,1)
%>_literal
)init
.help=wtoreph
)proc
&reply=.resp
VER (&literal,NB)
)End
```

WTOREPH

The following is the HELP panel for command issuing:

```
)BODY
'──────── Help Panel For REPLY command ──────────
+
+Please enter a command to be issued to the system.
+
+If it is a reply to a message as shown on the WTOR screen, then enter
+the number on the message as shown followed by the reply.
+
+Otherwise if you just wish to enter a command simply type the command
+to be issued.

 Press PF3 if you do not wish to issue a command.
+
+
+Note that you will need the necessary authority in RACF OPERCMDS for
+the command to actually be issued.
)PROC
.help=isp00004
)END
```

A REXX utility to delete PDS members – part 2

This month we complete our look at the REXX utility to delete PDS members with a simple command without going through ISPF option 3.1.

```
    /*————————Note————————————*/
    /* If zlcdate is not in yy/mm/dd format        */
    /*    uncomment and modify lines suitably      */
    /*————————Note————————————*/
    /* Parse Var zlcdate mm '/' dd '/' yy          */
    /* zlcdate=yy'/'mm'/'dd                        */
    /*————————Note————————————*/
    Parse Var xSaveCrt xPart1 'CRE' xPart2
    xSaveCrt=xPart1||zlcdate||xPart2
    End
   Else sThisMem='I'
 End
If Pos('CHA',xSaveCrt)>0 Then Do
  nPos=Pos('CHA',xSaveCrt)+3
  xComprtr=Substr(xSaveCrt,nPos,1)
  If zlmdate<>'' & Pos(xComprtr,'\=<>^')>0 Then Do
    /*————————Note————————————*/
    /* If zlmdate is not in yy/mm/dd format        */
    /*    uncomment and modify lines suitably      */
    /*————————Note————————————*/
    /* Parse Var zlmdate mm '/' dd '/' yy          */
    /* zlmdate=yy'/'mm'/'dd                        */
    /*————————Note————————————*/
    Parse Var xSaveCrt xPart1 'CHA' xPart2
    xSaveCrt=xPart1||zlmdate||xPart2
    End
   Else sThisMem='I'
 End
If Pos('~',xSaveCrt)>0 Then Do
  Parse Var xSaveCrt xPart1 '~' xPart2
  xSaveCrt=xPart1||'CHA'||xPart2
  End
If Pos('\',xSaveCrt)>0 Then Do
  Parse Var xSaveCrt xPart1 '\' xPart2
  xSaveCrt=xPart1||'CRE'||xPart2
  End
/*If sThisMem='N' Then Do*/
/*  Interpret 'cTrue='Translate(xSaveCrt,'!','/')*/
/*  If cTrue Then sThisMem='Y'*/
/*  End*/
/* Else sThisMem='N'*/
Interpret 'cTrue='Translate(xSaveCrt,'!','/')
Select
```

```
 When \Datatype(cTrue,'W') Then Do
  Say '*Error* Invalid Parameter passed ' xParams
  Say '        Do not enclose Criteria under quotes'
  sPassCriteria='N'
  sTime2Leave='Y'
  sFatalErr='Y'
  End
 When cTrue Then sPassCriteria='Y'
 Otherwise sPassCriteria='N'
 End
Return

OneMemOnly:
sWrongUser='N'
Select
 When xMemPat<>xMemName Then Do
  sDeletes='N'
  sTime2Leave='Y'
  End
 When sOthers='Y' Then Do
  sThisMem='Y'
  End
 When sOthers='N' & zluser=Userid() Then Do
  sThisMem='Y'
  End
 Otherwise Do
  sWrongUser='Y'
  sDeletes='N'
  sTime2Leave='Y'
  End
 End
Return

RiteAudit:
If nOutRecs=Ø Then Call AllocOut
'EXECIO 1 DISKW xOutDsn  (STEM xAudit.'
If Rc<>Ø Then Do
  Say '*Error* Write Failure On ' xFileOut 'RC='rc
  Say '        Program Aborted'
  xSamFir=MSG('OFF')
  'FREE DD(xOutDsn)'
  xAskNb=MSG(xSamFir)
  Exit 16
  End
nOutRecs=nOutRecs+1
Return

AllocOut:
xOutName="'"xOutName"'"
xAvail=SYSDSN(xOutName)
If xAvail='OK' Then Do
```

```
  xSamFir=MSG('OFF')
  'DELETE' xOutName
  xAskNb=MSG(xSamFir)
  If Rc<>Ø Then Do
    Say '*Error*' xOutName 'Could Not Be Deleted - Return Code = ' Rc
    Say 'Program Aborted'
    Exit
    End
  End
nLrecl=132
xDfltDisp='NEW UNIT(SYSDA) LRECL('nLrecl')',
          'SPACE(2Ø) DSORG(PS) RECFM(F,B) TRACKS  RELEASE'
xSamFir=MSG('OFF')
'FREE DD(xOutDsn)'
xAskNb=MSG(xSamFir)
'ALLOCATE DSN('xOutName') DD(xOutDsn)' xDfltDisp
If Rc<>Ø Then Do
  Say '*Error* Unable To Alloc' xOutName
  Say 'Return Code Is ' Rc
  Exit 16
  End
Return

EojInfo:
Select
 When sFatalErr='Y' Then Nop
 When sDeletes='N' & sWrongUser='Y' Then Do
  Say '*Note*'xMemPat 'Was Created/Modified By' zluser
  Say '        Program Could Not Delete This Member'
  End
 When sDeletes='N' & nOthrMems>Ø Then Do
  If xMemPat='' Then xMemPat='*'
  Say '*Warning* No Members Found In Pds' xFromPds 'That Could Match' xMemPat
  Say '          And Created/Modified By' Userid()
  Say '          (However There Were' nOthrMems 'Members Created By Others'
  Say '          Program Could Not Delete Any Members'
  End
 When sDeletes='N' & xMemPat='' Then Do
  Say '*Warning* No Members Found In Pds' xFromPds
  Say '          That Could Match' xCriteria
  Say '          ( Has the criteria been coded correctly? )'
  Say '          Program Could Not Delete Any Members'
  End
 When sDeletes='N' Then Do
  Say '*Warning* No Members Found In PDS' xFromPds 'That Could Match' xMemPat
  Say '          Program Could Not Delete Any Members'
  End
 When sWatch<>'Y' Then Do
  Say '*Note* List Of Members Deleted is in' xOutName
  Say '       (Total Members Deleted='nDelMems')'
  End
 Otherwise Nop
```

```
 End
Say '*Note* Job terminated normally'
Return

InitVarbls:
nDelMems=Ø
nLimit=Ø
nOthrMems=Ø
nOutRecs=Ø
nStars=Ø
sDeletes='N'
sFatalErr='N'
sJust1Mem='N'
sPassCriteria=''
sTime2Leave='N'
xAudit.Ø=1
xPrfx=''
xSufx=''
Return

/* ——Note————Note————————Note——— */
/* The Defaults are defined here. Modify them to suit your          */
/* installations standards.                                         */
/* ——Note————Note————————Note——— */

BuildDflts:
xOutName=Userid()'.DELETED.MEMBERS.LIST'     /* Output Dataset Name  */
sWatch='N'                              /* Y will display delete mems  */
sOthers='N'                             /* Y = select All Members      */
                                        /* N = select My Members ONLY  */
Return

SplitParams:
xCriteria=''
xFromPds=''
Select
 When Words(xParams)=1 Then Do
  If Verify(xParams,'=<>','M')>Ø Then xCriteria=xParams
   Else xFromPds=xParams
  End
 When Words(xParams)=2 Then Do
  If Verify(Word(xParams,1),'=<>','M')=Ø Then Parse Var xParams xFromPds
xCriteria
   Else Parse Var xParams xCriteria xFromPds
  End
 Otherwise Do
  Say '*Error*' xParams 'Contains Invalid Parameters'
  Say '       Program Aborted'
  Exit
  End
 End
Return
```

```
How2Use:
'CLRSCRN'
Parse Source . . xExecName .
Queue 'Use ' xExecName 'To Delete Members From a Partition Dataset (PDS)'
Queue '—————————————————————————'
Queue 'Format of the Command is (Assuming that the EXEC is in one of your'
Queue '                          SYSEXEC Libraries)'
Queue '<TSO>' xExecName 'PdsName<Pattern> <Criteria>'
Queue 'If you are not in ISPF option 6, then suffix the command with TSO'
Queue 'For PDS name, follow TSO rules. Place the PDS name in quotes if'
Queue 'it is not starting with your Userid.'
Queue 'Type just the PDS name to delete members created/modified by' Userid()
'OR'
Queue 'Choose a single member OR'
Queue 'Choose a pattern for partial selection - Use * as a wild character'
Queue 'Note:- Only one wild character allowed for partial selection'
Queue 'Example: Your selection can be  SA*ENA OR NB* OR *ASK'
Queue 'You can further limit your selection by using Criteria'
Queue 'Criteria can be any combination of'
Queue '  ID , CRE(ated Date in YY/MM/DD) And/Or CHA(nged Date in YY/MM/DD'
Queue 'Please ensure that the statistics have been saved in YY/MM/DD format'
Queue 'If not you may have to modify the zlcdate and zlmdate suitably'
Queue 'Example Of Criteria:- ID=MOYEENKH&CHA<97/Ø1/Ø1'
Queue '                      ID<>HASLEFT'
Queue '                      CRE<97/Ø1/Ø1'
Queue "                      ID='' or ID<>''"
Queue '<>=\^ are the only comparators that are permitted in Criteria.'
Queue 'Use "&" to AND and "|" to OR the arguments'
Queue 'To use the criteria you must pass it when you are calling' xExecName
Queue 'If you have not used criteria, then the program will ask you'
Queue 'whether to delete members created/modified by others'
Do Until Queued()=Ø
 Parse Pull xHelp
 Say Copies(' ',5) xHelp
 End
Return
```

Moyeen Ahmed Khan
Systems Programmer (Canada)

Year 2000 aid: change ‘DATE’= parameters

INTRODUCTION

This program, YEAR2KCD, is similar to YEAR2KC which can be found in *MVS Update* Issues 136 and 137. The major differences are:

- YEAR2KCD does not change all dates to a ten-digit format, but modifies the existing date.
- YEAR2KCD assumes that dates, including century, are in the form ‘mm/dd/ccyy’.
- YEAR2KCD does not contain an ERROR file since all updates are in place.

It is intended to be used while conversion to ten-digit years is in progress. Except for the program name and, if desired, removal of the ERROR dataset definition, the JCL remains the same.

PROGRAM SOURCE

```
         TITLE 'PROGRAM MODIFIES DATE= PARMS IN A  PARTITONED DATASET'
*************************************************************************
* THIS PROGRAM READS ALL MEMBERS OF THE SPECIFIED PARTIONED DATASET,  *
* SEARCHING FOR JCL THAT SPECIFIES DATE PARAMETERS.  WHEN SUCH A      *
* PARAMETER IS FOUND IT IS OVERLAID WITH THE VALUE SPECIFIED BY THE   *
* PARAMETER FIELD INCLUDED ON THE ENVOKING EXEC CARD.                 *
*                                                                     *
* THE DATE IS LEFT IN THE FORMAT FOUND WHICH IS ASSUMED TO BE EITHER  *
* MM/DD/YY OR MM/DD/CCYY.                                             *
*************************************************************************
         SPACE 2
         MACRO
&NAME    MOVEPAT &X
         MVC   &X+1(L'&X-1),PATTERN+L'PATTERN-L'&X+1
         MEND
         DC    C'YEAR2KCD—ASSEMBLED &SYSDATE/&SYSTIME'
         PRINT NOGEN
BEGIN    SAVE  (14,12)             SAVE REGISTERS
         BALR  12,Ø                LOAD BASE REGISTER
         USING *,12                ESTABLISH ADDRESSABILITY
         ST    13,SAVEAREA+4       SAVE ENTRY REGISTER
         LA    13,SAVEAREA         POINT TO SAVE AREA
```

```
         L     6,Ø(1)              SAVE ADDRESS OF PARAMETER
         OPEN  (PDS,UPDAT,PRINTER,OUTPUT,PDSDIR) OPEN FILES
         LH    8,Ø(6)              SET LENGTH
         LA    6,1(6)              POINT TO BYTE PRECEEDING INFO FIELD
         XR    7,7                 CLEAR INITIAL LENGTH
         TIME
         ST    1,TODAY             SAVE DATE (ØØYYDDDC)
         SRA   1,16                EXTRACT CURRENT YEAR
         SLL   1,4                 ALLOW ROOM FOR SIGN
         ST    1,YEARS+4           STORE YYØ
         OI    YEARS+7,X'C'        SET SIGN
         AP    YEARS,=P'19ØØ'      SET CENTURY
         CVB   1,YEARS             CONVERT TO BINARY
         STC   1,LEAPFLAG          SAVE FOR LEAP YEAR TEST
         BCTR  1,Ø                 LAST YEAR
         M     Ø,=F'365.25'        1ØØ*YEARS*DAYS/YEAR
         D     Ø,1ØØ               DAYS FROM 1/Ø/ØØ TO 12/31/(YR-1)
         CVD   1,SAVEDAYS          CONVERT TO DECIMAL
         ZAP   DAYS,TODAY+2(2)     SAVE DAYS FROM BEGINNING OF YEAR
         AP    SAVEDAYS,DAYS       DAYS FROM 1/Ø/ØØ TO TODAY
         ZAP   MONTHS,=P'1'        INITIALIZE MONTH
         LM    15,1,=A(JANUARY,L'JANUARY,DECEMBER) DAYS/MONTH TABLE
         TM    LEAPFLAG,3          LEAP YEAR?
         BNZ   DDDLOOP              NO
         AP    FEBRUARY,=P'1'      ADJUST
DDDLOOP  CP    DAYS,Ø(L'JANUARY,15) CURRENT MONTH?
         BNH   DDDFOUND             YES
         AP    MONTHS,=P'1'        INCREMENT MONTH
         SP    DAYS,Ø(L'JANUARY,15) DECREMENT DAYS PER CURRENT MONTH
         BXLE  15,Ø,DDDLOOP        CONTINUE
DDDFOUND UNPK  DATE(2),MONTHS      UNPACK MONTH
         UNPK  DATE+3(2),DAYS      UNPACK DAY
         UNPK  DATE+6(4),YEARS     UNPACK YEAR
         MVI   DATE+2,C'/'         SEPARATE MONTH AND DAY
         MVI   DATE+5,C'/'         SEPARATE DAY AND YEAR
         OI    DATE+1,C'Ø'         FORCE MONTH NUMERIC
         OI    DATE+4,C'Ø'         FORCE DAY NUMERIC
         OI    DATE+9,C'Ø'         FORCE YEAR NUMERIC
         MVC   NEWDATE,DATE        INITIALIZE DATE
*        B     OVERRODE
         LTR   2,8                 LOAD PARM LENGTH
         BNP   OVERRODE            BRANCH IF NOT POSITIVE
         LA    Ø,L'PARMHEAD        LOAD MAXIMUM LENGTH
         CR    Ø,2                 MAXIMUM EXCEEDED?
         BNL   PARMLOK              NO
         LR    2,Ø                 LOAD MAXIMUM LENGTH
PARMLOK  EX    2,MOVEPARM          MOVE PARAMETERS TO LINE
         LA    1,PARMHEAD(8)       POINT PAST END OF PARAMETERS
         MVI   Ø(1),C')'           TERMINATE
         BAL   11,PRINT            PRINT PARAMETER LISTING
         MVI   OUTAREA,C'Ø'        SET TO DOUBLE SPACE NEXT LINE
```

```
         SPACE
PARMLOOP LA    4,PARMEND           POINT TO NULL RETURN
         BAL   14,KHNSCAN          GET PARAMETER
         BAL   11,TEST             FOR TESTING
NOTSUBPM LA    4,PARMERR           POINT TO NULL RETURN
         CLC   =C'DATE=',Ø(6)      'DATE' OPTION?
         BE    SETDATE              YES
         CLC   =C'PRNT=',Ø(6)      'PRINT' OPTION?
         BE    SETPRINT             YES
         CLC   =C'MEMB=',Ø(6)      'MEMBER' OPTION?
         BE    SETNAME              YES
         CLC   =C'CTRL=',Ø(6)      'CONTROL' OPTION?
         BNE   PARMERR              NO
         MVI   SIGN,X'F'           INITIALIZE SIGN (NOT REALLY NECESRY)
         BAL   14,KHNSCAN          GO GET CONTROL VALUE
         CLI   Ø(6),C'X'           OVERRIDE?
         BE    OVERRIDE             YES
         BAL   11,TEST             FOR TESTING
         BAL   14,NUMTEST          VERIFY THAT IT'S NUMERIC
         ZAP   CONTROL,PACKWORK    SET VALUE
         B     PARMLOOP            CONTINUE PARAMETER SCAN
OVERRIDE MVI   CONTROL,X'FF'       SET OVERRIDE
         B     PARMLOOP            CONTINUE PARAMETER SCAN
         SPACE
SETDATE  BAL   14,KHNSCAN          SCAN FOR MM/DD/YY
         BAL   11,TEST             FOR TESTING
         CLC   =C'TODAY',Ø(6)      CURRENT SYSTEM DATE?
         BE    SETTODAY             YES
         CH    7,=H'9'             IS FIELD 1Ø BYTES LONG?
         BNE   PARMERR              NO
         ZAP   TODAY,SAVEDAYS      INITIALIZE DATE
         CLC   DATE,Ø(6)           SAME AS TODAY?
         BE    PARMLOOP             YES
         CLI   2(6),C'/'           SEPARATOR BETWEEN MONTH AND DAY?
         BNE   PARMERR              NO
         CLI   5(6),C'/'           SEPARATOR BETWEEN DAY AND YEAR?
         BNE   PARMERR              NO
         MVC   MMDDCCYY,=X'ØØ1Ø3Ø4Ø6Ø7Ø8Ø9' SET GATHER PATTERN
         TR    MMDDCCYY,Ø(6)       GATHER MMDDCCYY
         MVC   TRTAB,TRTAB-1       MAKE TABLE NON ZERO
         XC    TRTAB+C'Ø'(1Ø),TRTAB+C'Ø' TURN OFF NUMERIC PORTION
         TRT   MMDDCCYY,TRTAB      IS MMDDCCYY NUMERIC?
         BNZ   PARMERR             NO
         CLC   =C'ØØ',MMDDCCYY     IS MONTH OKAY?
         BNL   PARMERR              NO
         CLC   =C'12',MMDDCCYY
         BL    PARMERR              NO
         CLC   =C'ØØ',MMDDCCYY+2   IS DAY OKAY?
         BNL   PARMERR              NO
         CLC   =C'19',MMDDCCYY+4   IS CENTURY OK?
         BH    PARMERR              NO
```

```
         CLC   =C'2Ø',MMDDCCYY+4
         BL    PARMERR              NO
         PACK  MONTHS,MMDDCCYY(2)   PACK MONTH
         PACK  DAYS,MMDDCCYY+2(2)    DAY
         PACK  YEARS,MMDDCCYY+4(4)  YEAR
         ZAP   DOUBLE,YEARS         MOVE YEAR TO DOUBLE WORD
         CVB   Ø,DOUBLE             LOAD INTO REGISTER
         STC   Ø,LEAPFLAG           SAVE BINARY LOW ORDER BYTE
         ZAP   FEBRUARY,=P'28'      ASSEME NOT LOOP YEAR
         TM    LEAPFLAG,3           LEAP YEAR?
         BNZ   NOTLEAP               NO
         ZAP   FEBRUARY,=P'29'      SET FOR LEAP YEAR
NOTLEAP  ZAP   DOUBLE,MONTHS        MOVE MONTH TO DOUBLE WORD
         CVB   1,DOUBLE             LOAD INTO REGISTER
         LR    15,1                 SAVE FOR BELOW
         MH    1,=AL2(L'JANUARY)    * TABLE WIDTH
         LA    1,JANUARY-L'JANUARY(1) INDEX TABLE
         CP    Ø(L'JANUARY,1),DAYS IS DATE TOO LARGE?
         BL    PARMERR              YES
         MVC   NEWDATE,Ø(6)         SAVE DATE FOR OVERLAY (SEE CKEKDATE)
         LR    1,Ø                  SAVE YEAR
         BCTR  1,Ø                  LAST YEAR
         M     Ø,=F'365.25'         1ØØ*YEARS*DAYS/YEAR
         D     Ø,1ØØ                DAYS FROM 1/Ø/ØØ TO 12/31/(YR-1)
         CVD   1,DOUBLE             CONVERT TO DECIMAL
         ZAP   TODAY,DOUBLE         SAVE DAYS
         AP    TODAY,DAYS           ADD DAYS FROM ENTRY
         LA    1,JANUARY-L'JANUARY  POINT TO ZERO DAYS
DATELOOP AP    TODAY,Ø(L'JANUARY,1) ADD DAYS IN MONTH
         LA    1,L'JANUARY(1)       POINT TO NEXT MONTH
         BCT   15,DATELOOP          ACCUMULATE DAYS IN PREVIOUS MONTHS
         B     PARMLOOP             GO GET NEXT PARAMETER
         SPACE
SETTODAY MVC   NEWDATE,DATE         GET CURRENT DATE
         B     PARMLOOP             GO GET NEXT PARAMETER
         SPACE
SETPRINT BAL   14,KHNSCAN           GET PRINT OPTION
         BAL   11,TEST              FOR TESTING
         CLC   =C'DIAG',Ø(6)        DIAGNOSE OPTION?
         BE    DIAGNOSE              YES
         CLC   =C'BEFORE',Ø(6)      BEFORE OPTION?
         BE    BEFORE                YES
         CLC   =C'AFTER',Ø(6)       AFTER OPTION?
         BE    AFTER                 YES
         CLC   =C'LIST',Ø(6)
         BNE   NOTSUBPM              NO
         OI    OPTIONS,LISTBIT      TURN ON OPTION
PRINT4   CLI   4(6),C','            ADDITIONAL PRINT OPTION?
         BE    SETPRINT              YES
         B     PARMLOOP              NO
DIAGNOSE OI    OPTIONS,DIAGBIT      TURN ON OPTION
```

```
         B     PRINT4              GO CHECK FOR ADDITIONAL PRNT OPTS
BEFORE   OI    OPTIONS,BFOREBIT    TURN ON OPTION
         CLI   6(6),C','           ADDITIONAL PRINT OPTION?
         BE    SETPRINT             YES
         B     PARMLOOP             NO
AFTER    OI    OPTIONS,AFTERBIT    TURN ON OPTION
         CLI   5(6),C','           ADDITIONAL PRINT OPTION?
         BE    SETPRINT             YES
         B     PARMLOOP             NO
         SPACE
SETNAME  BAL   14,KHNSCAN          GET PRINT OPTION
         BAL   11,TEST             FOR TESTING
         MVC   MEMBER,=8C' '       INITIALIZE NAME PADDING
         EX    7,MOVENAME          MOVE MEMBER NAME
         OI    OPTIONS,MEMBRBIT    SET OPTION BIT
         B     PARMLOOP             NO
         SPACE
PARMEND  CLI   CONTROL,X'FF'       OVERRODE?
         BE    OVERRODE             YES
         SP    TODAY,CONTROL       ADD CONTROL DATE TO ENTERED DATE
         CP    TODAY,SAVEDAYS      SAME AS TODAY'S DATE?
         BNE   DATEERR              NO
OVERRODE BAL   11,HEADPAGE         PRINT PAGE HEADING
         SPACE 3
READDIR  GET   PDSDIR,DIRBLOCK     READ DIRECTORY BLOCK
         LA    6,DIRBLOCK+2        LOAD ADDRESS OF FIRST ENTRY
         LH    5,DIRBLOCK          LOAD NUMBER OF BYTES USED
         STH   5,DIRSPACE          SAVE
         B     FIRSTDIR            GO PROCESS FIRST RECORD OF BLOCK
         SPACE
DIRLOOP  L     6,DIRENTRY          LOAD ADDRESS OF CURRENT LOCATION
         LH    5,DIRSPACE          LOAD REMAINING SPACE IN BLOCK
         IC    1,11(6)             LOAD 'C'
         N     1,=F'31'            GET USER AREA HALFWORDS (5 LOW BITS)
         LA    1,12(1,1)           BYTES + MEMBER NAME, 'TTR', AND 'C'
         SR    5,1                 DEDUCT CURRENT ENTRY LENGTH
         AR    6,1                 POINT TO NEXT ENTRY
         SPACE
FIRSTDIR CLI   Ø(6),X'FF'          LAST DIRECTORY ENTRY?
         BE    DIREND               YES
         CH    5,=H'11'            ROOM FOR ADDITIONAL ENTRY?
         BL    READDIR              NO
         ST    6,DIRENTRY          SAVE CURRENT POINTER
         STH   5,DIRSPACE          SAVE REMAINING SPACE
         MVC   TTRN,8(6)           SAVE RELATIVE DASD ADDRESS
         MVI   TTRN+3,Ø            CLEAR 'N'
         CLI   TTRN+2,Ø            VALID ADDRESS?
         BZ    NOMEMBER             NO (SHOULDN'T HAPPEN)
         ZAP   CARDS,=P'Ø'         INITIALIZE COUNT
         AP    MEMBERS,=P'1'       COUNT DIRECTORY ENTRY
         NI    SWITCHES,X'FF'-UPDATBIT CLEAR UPDATE SWITCH
```

```
*        CP    MEMBERS,=P'1Ø'      FOR TESTING
*        BH    DIREND              FOR TESTING
         POINT PDS,TTRN            POINT TO NOTE LIST RECORD
         SPACE
READPDS  LA    2,DECBA             POINT TO DECB
         READ  (2),SF,PDS,MF=E     READ BLOCK FROM MEMBER
         CHECK (2)                 AWAIT ECB POSTING
         LA    9,PDS               GET ADDRESS OF PDS DCB
         USING IHADCB,9            ESTABLISH ADDRESSABILITY
         LH    8,DCBLRECL          LOAD RECORD LENGTH
         LH    9,DCBBLKSI          LOAD MAXIMUM BLOCK SIZE
         DROP  9                   DROP ADDRESSABILITY
         L     1,DECBA+16          LOAD RECORD POINTER WORD
         SH    9,14(1)             SUBTRACT REMAINING COUNT
         L     7,=A(BLOCK)         GET ADDRESS OF BLOCK
         AR    9,7                 POINT TO END OF BLOCK
         BCTR  9,Ø                 POINT TO LAST BYTE OF BLOCK
         MVC   SAVENAME,Ø(6)       SAVE MEMBER NAME
         TM    OPTIONS,MEMBRBIT    MEMBER OPTION?
         BZ    OUTLOOP              NO
         CLC   MEMBER,Ø(6)         NAME FOUND?
         BNE   DIRLOOP              NO
         SPACE
OUTLOOP  MVC   MEMBNAME,Ø(6)       MOVE MEMBER NAME
         PUSH  PRINT               SAVE PRINT OPTIONS
         PRINT NOGEN
         MOVEPAT MEMBERNO          MOVE EDIT PATTERN
         POP   PRINT               RESTORE PRINT OPTIONS
         ED    MEMBERNO,MEMBERS+1  FORMAT MEMBER NUMBER
         MVC   INAREA,Ø(7)         MOVE RECORD
         AP    CARDS,=P'1'         COUNT CARD IMAGE
         MOVEPAT CARDNO            MOVE EDIT PATTERN
         ED    CARDNO,CARDS        FORMAT CARD NUMBER
         TM    OPTIONS,LISTBIT+DIAGBIT LIST OR DIAGNOSE?
         BZ    NOLIST               NO
         TM    OPTIONS,DIAGBIT     DIAGNOSE?
         BZ    NOLIST               NO
         BAL   11,TESTX            PRINT DIAGNOSTIC LINE
         B     NOLIST              BYPASS VANILLA LISTING
         BAL   11,PRINT            PRINT CARD IMAGE
NOLIST   STM   5,9,SAVE5TO9        SAVE REGISTERS
         CLC   =C'//',Ø(7)         JCL CARD?
         BNE   NOTJCL               NO
         CLI   2(7),C'*'           COMMENTS CARD?
         BE    NOTJCL               YES
         TM    SWITCHES,CONTBIT    CONTINUATION CARD EXPECTED?
         BO    CONTINUD             YES
         NI    SWITCHES,UPDATBIT   TURN OFF ALL EXCEPT UPDATE SWITCH
CONTINUD LR    6,7                 POINT TO BEGINNING OF CARD IMAGE
         BCTR  6,Ø                 POINT TO PREVIOUS BYTE
```

```
         LA    4,NOTJCL            SET NULL RETURN
         XR    7,7                 CLEAR FIELD LENGTH
         LA    8,72                SET CARD IMAGE FIELD LENGTH
*        BAL   11,TEST             FOR TESTING
         BAL   14,KHNSCAN          SKIP PAST '//NAME'
         TM    OPTIONS,DIAGBIT     DIAGNOSE OPTION?
         BZ    NODIAG1              NO
         BAL   11,TEST             FOR TESTING
NODIAG1  TM    SWITCHES,CONTBIT    IS THIS A CONTINUATION CARD?
         BO    SKIPTYPE             YES
         BAL   14,KHNSCAN          SEARCH FOR JCL TYPE OPERATOR
         TM    OPTIONS,DIAGBIT     DIAGNOSE OPTION?
         BZ    NODIAG2              NO
         BAL   11,TEST             FOR TESTING
NODIAG2  CLC   =C'EXEC',Ø(6)       EXECUTE CARD?
         BNE   SKIPTYPE             NO
         BAL   14,KHNSCAN          SCAN FOR POTENTIAL PROGRAM NAME
         TM    OPTIONS,DIAGBIT     DIAGNOSE OPTION?
         BZ    NODIAG3              NO
         BAL   11,TEST             FOR TESTING
NODIAG3  NI    SWITCHES,X'FF'-CONTBIT TURN OFF CONTINUATION BIT
         CLC   =C'PGM=',Ø(6)       ANY PROGRAM?
         BNE   NEXTSTEP             NO
         SPACE
         OI    SWITCHES,PGMBIT     TURN ON PGM SWITCH
         B     NEXTSTEP            GO CHECK FOR END OF STATEMENT
         SPACE
SKIPTYPE NI    SWITCHES,X'FF'-CONTBIT TURN OFF CONTINUATION BIT
         SPACE
CONTINUE BAL   14,KHNSCAN          SCAN FOR JCL GROUP
         TM    SWITCHES,DATEBIT    MM/DD/YY EXPTECTED?
         BO    CHEKDATE             YES
         TM    OPTIONS,DIAGBIT     DIAGNOSE OPTION?
         BZ    NODIAG4              NO
         BAL   11,TEST             FOR TESTING
NODIAG4  TM    SWITCHES,PGMBIT     PROGRAM?
         BZ    NEXTSTEP             NO
         CLC   =C'PARM=',Ø(6)      'PARM' GROUP?
         BNE   NOTPARM              NO
         OI    SWITCHES,PARMBIT    TURN ON PARM BIT
         B     NEXTSTEP            GO GET PARMETER FIELDS
         SPACE
NOTPARM  TM    SWITCHES,PARMBIT    IN 'PARM' GROUP?
         BZ    NEXTSTEP             NO
         TM    SWITCHES,QUOTEBIT   STILL IN PARM='...'?
         BO    PARMYET              YES
         CP    NESTS,=P'Ø'         STILL IN PARM=(...)?
         BNZ   PARMYET              YES
         NI    SWITCHES,X'FF'-PARMBIT TURN OFF PARM BIT
         SPACE
```

```
NEXTSTEP LA    1,1(6,7)            POINT PAST END OF FIELD
         TM    SWITCHES,QUOTEBIT   WITHIN QUOTATION?
         BO    CONTINUE             YES
         CLI   Ø(1),C' '
         BE    NOTJCL               YES
         B     CONTINUE            GO GET NEXT PARAMETER
         SPACE
PARMYET  CLC   =C'DATE=',Ø(6)      DATE SUB PARAMETER?
         BNE   NEXTSTEP             NO
         OI    SWITCHES,DATEBIT    TURN ON DATE BIT
*
         BAL   14,KHNSCAN          GET DATE
*
         TM    OPTIONS,DIAGBIT     DIAGNOSE?
         BZ    CHEKDATE             NO
         BAL   11,TEST             PRINT DIAGNOSTIC
*
CHEKDATE NI    SWITCHES,X'FF'-DATEBIT TURN OFF DATE BIT
*        CH    7,=H'8'             SPECIAL MM/DD/YY/? STUPID COBOL!
*        BNE   NOTCOBOL             NO
*        CLI   8(6),C'/'
*        BNE   NEXTSTEP             NO
*        B     COBOL                YES
*OTCOBOL CH    7,=H'7'             IS IT OF THE FORM 'MM/DD/YY'?
*        BNE   NEXTSTEP             NO
COBOL    CLI   2(6),C'/'
         BNE   NEXTSTEP             NO
         CLI   5(6),C'/'
         BNE   NEXTSTEP             NO
         TM    OPTIONS,LISTBIT     LIST OPTION?
         BO    NOTBFORE             YES (IE DONE)
         TM    OPTIONS,DIAGBIT+BFOREBIT DIAGNOSE OR BEFORE OPTIONS?
         BNM   NOTBFORE             BOTH (IE DONE) OR NEITHER
         TM    OPTIONS,BFOREBIT    BEFORE OPTION?
         BZ    NOTBFORE             NO
         L     1,SAVE5TO9+8        LOAD ADDRESS OF CARD IMAGE
         MVC   INAREA,Ø(1)         MOVE TO PRINT LINE
         BAL   11,PRINT            PRINT BEFORE REPLACING DATE
*
NOTBFORE CLI   8(6),C'/'           COBOL TYPE MM/DD/YY/?
         BNH   NBNCENT              YES
*
         TM    8(6),C'Ø'           NUMERIC IN 9TH BYTE?
         BNO   NBNCENT              NO
*
         CH    7,=H'9'             CONTAINS CENTURY?
         BL    NBNCENT              NO
         BE    NBCENT               YES
*
         CLI   1Ø(6),C'/'          COBOL TYPE MM/DD/CCYY/?
```

```
         BE    NBCENT               YES
*
NBNCENT  MVC   Ø(6,6),NEWDATE      OVERLAY MM/DD/
         MVC   6(2,6),NEWDATE+8    OVERLAY YY
         B     NBCOUNT             GO COUNT DATE OCCURANCE
*
NBCENT   MVC   Ø(1Ø,6),NEWDATE     OVERLAY DATE (MM/DD/CCYY)
*
NBCOUNT  AP    DATES,=P'1'         INCREMENT COUNT FOR THIS BLOCK
         OI    SWITCHES,UPDATBIT   INDICATE UPDATE OCCURANCE
         TM    OPTIONS,AFTERBIT    AFTER OPTION?
         BZ    NEXTSTEP             NO
         L     1,SAVE5TO9+8        LOAD ADDRESS OF CARD IMAGE
         MVC   INAREA,Ø(1)         MOVE TO PRINT LINE
         BAL   11,PRINT            PRINT LINE
         B     NEXTSTEP            GO PROCESS NEXT FIELD
         SPACE
NOTJCL   LM    5,9,SAVE5TO9        RESTORE REGISTERS
         TM    SWITCHES,UPDATBIT   ANY UPDATES?
         BZ    NOUPDATE             NO
UPDATES  DS    ØC                  LATER UPDATE RECORD
         LA    2,DECBA             POINT TO DECB
         WRITE (2),SF,PDS,MF=E     READ BLOCK FROM MEMBER
         CHECK (2)                 AWAIT ECB POSTING
         SPACE
NOUPDATE BXLE  7,8,OUTLOOP         CONTINUE UNTIL BLOCK COMPLETED
         B     READPDS             GO GET NEXT BLOCK
         SPACE
         B     READPDS             CONTINUE READ
         SPACE
NEXTMEMB TM    SWITCHES,UPDATBIT   ANY UPDATES?
         BO    DIRLOOP              YES
         MVC   OUTAREA,OUTAREA-1   CLEAR CARD IMAGE
         MOVEPAT MEMBERNO          MOVE EDIT PATTERN
         ED    MEMBERNO,MEMBERS+1  FORMAT MEMBER NUMBER
         MVC   MEMBNAME(25),SAVENAME INDICATE NO CHANGES
         BAL   11,PRINT            PRINT “NO CHANGE” LINE
         SPACE
         B     DIRLOOP             GO GET NEXT MEMBER
         SPACE 2
DIREND   DS    ØH                  END OF DIRECTRY
         CLOSE (PDS,,PRINTER,,PDSDIR) CLOSE FILES
RETURN   L     13,SAVEAREA+4       RESTORE ENTRY REGISTER
         RETURN (14,12),RC=Ø       RESTORE REGISTERS/RETURN
SAVEAREA DC    18F'Ø'
NOMEMBER MVC   OUTAREA+2(8),Ø(6)   SET MEMBER NAME
         MVC   OUTAREA+11(9),=C'NOT FOUND' SET ERROR MESSAGE
         BAL   11,PRINT            PRINT ERROR LINE
         B     DIRLOOP             GO PROCESS REMAINDER OF LIST
         SPACE
MOVENAME MVC   MEMBER(*-*),Ø(6)
```

```
MOVEPARM MVC   PARMHEAD(*-*),1(6)
MOVETEST MVC   INAREA(*-*),Ø(6)
TEST     EX    7,MOVETEST          MOVE CHUNK
TESTX    UNPK  TESTOPTS(3),OPTIONS(2) UNPACK NYBLS OF BIT SWTCHS
         UNPK  TESTSWTS(3),SWITCHES(2) UNPACK NYBLS OF BIT SWTCHS
         ST    7,DOUBLE            SAVE LENGTH
         UNPK  TESTLEN(5),DOUBLE(5) UNPACK NYBLS OF LENGTH
         ST    11,DOUBLE           SAVE RETURN ADDRESS
         UNPK  TESTLOC(5),DOUBLE(5) UNPACK NYBLS OF ADDRESS
         NC    TESTOPTS(15),=15X'F' TURN OFF ZONE BITS
         TR    TESTOPTS(15),=C'Ø123456789ABCDEF' CONVERT TO HEX DSPLY
         MVI   TESTSWTS+2,C' '     SET SEPARATOR
         MVI   TESTOPTS+2,C' '     SET SEPARATOR
         MVI   TESTLEN+4,C' '       "
         MVI   TESTLOC+4,C' '       "
         SPACE
PRINT    L     1,=A(PRINTER)       LOAD ADDRESS OF PRINTER DCB
         PUT   (1),OUTAREA         PRINT LINE
         MVI   OUTAREA,C' '        SET TO SINGLE SPACE
         MVC   OUTAREA+1(L'OUTAREA-1),OUTAREA CLEAR TO BLANKS
         BCTR  1Ø,11               RETURN IF PAGE IS NOT FULL
         SPACE
HEADPAGE AP    PAGES,=P'1'         INCREMENT PAGE HEADING
         MOVEPAT PAGENO            SET EDIT PATTERN
         ED    PAGENO,PAGES        FORMAT PAGE NUMBER
         L     1,=A(PRINTER)       LOAD ADDRESS OF PRINTER DCB
         PUT   (1),HEADING         PRINT PAGE HEADING
         MVI   OUTAREA,C'Ø'        SET TO DOUBLE SPACE
         LA    1Ø,56               LINES/PAGE
         BR    11                  RETURN
         SPACE 2
DATEERR  MVC   ERRMESSG+8(12),=C'DATE OR CTRL'
         B     RITEERR             GO PRINT MESSAGE
         SPACE
PARMERR  MVC   ERRMESSG+8(16),=C'PARAMETER OPTION'
         SPACE
RITEERR  BAL   11,HEADPAGE         PRINT PAGE HEADING
         MVI   OUTAREA,C'Ø'        SET TO DOUBLE SPACE
         BAL   11,TEST             PRINT CONDITIONS
         MVC   OUTAREA+1(24),ERRMESSG SET ERROR MESSAGE SUFFIX
         BAL   11,PRINT            PRINT ERROR MESSAGE
*        DC    H'Ø'
         B     RETURN              EXIT
ERRMESSG DC    CL24'INVALID'
         SPACE 2
BOMB     LR    8,Ø                 SAVE LOCATION OF ECB
         LR    9,1                 SAVE REG 1
         SYNADAF ACSMETH=BPAM      CREATE ERROR MESSAGE
         MVC   OUTAREA+1(6Ø),68(1) MOVE MESSAGE
         BAL   11,PRINT            PRINT SYNAD MESSAGE
         MVC   OUTAREA+1(34),=C'UAD,DT,DDNAME  ,OPER  ,ERROR DESCR'
```

```
MVC   OUTAREA+39(23),=C',BBBBCCCCHHHHRR,ACSMETH'
BAL   11,PRINT            PRINT SYNAD IDS
SYNADRLS
STCM  9,8,SYNADR1+1       SAVE LOW NYBL OF HIGH BYTE
SRL   9,4                 SHIFT NYBL
STCM  9,8,SYNADR1         SAVE HIGH NYBL OF HIGH BYTE
NC    SYNADR1,=15X'F'     FORCE ZONES OFF
TR    SYNADR1,=C'Ø123456789ABCDEF' CONVERT TO DISPLAY HEX
UNPK  SYNADECB(L'SYNADECB+1),Ø(L'SYNADECB/2+1,15) UNPACK DECB
NC    SYNADECB,=15X'F'    FORCE ZONES OFF
TR    SYNADECB,=C'Ø123456789ABCDEF' CONVERT TO DISPLAY HEX
MVI   SYNADECB+L'SYNADECB,C' ' OVERLAY BYTE
L     15,16(8)            LOAD ADDRESS OF IOB
UNPK  IOBW1(9),Ø(5,15)    UNPACK IOB BYTES Ø-3
NC    IOBW1,=15X'F'       FORCE ZONES OFF
TR    IOBW1,=C'Ø123456789ABCDEF' CONVERT TO DISPLAY HEX
MVI   IOBW1+8,C' '        OVERLAY TRAILING BYTE
UNPK  IOBW2(9),4(5,15)    UNPACK IOB BYTES Ø-3
NC    IOBW2,=15X'F'       FORCE ZONES OFF
TR    IOBW2,=C'Ø123456789ABCDEF' CONVERT TO DISPLAY HEX
MVI   IOBW2+8,C' '        OVERLAY TRAILING BYTE
UNPK  IOBW3(9),8(5,15)    UNPACK IOB BYTES Ø-3
NC    IOBW3,=15X'F'       FORCE ZONES OFF
TR    IOBW3,=C'Ø123456789ABCDEF' CONVERT TO DISPLAY HEX
MVI   IOBW3+8,C' '        OVERLAY TRAILING BYTE
UNPK  IOBW4(9),12(5,15)   UNPACK IOB BYTES Ø-3
NC    IOBW4,=15X'F'       FORCE ZONES OFF
TR    IOBW4,=C'Ø123456789ABCDEF' CONVERT TO DISPLAY HEX
MVI   IOBW4+8,C','        OVERLAY TRAILING BYTE
MVC   OUTAREA+1(SYNADERL),SYNADER
UNPK  SENSE(5),2(3,15)    UNPACK SENSE BYTES
NC    SENSE,=15X'F'       FORCE ZONES OFF
TR    SENSE,=C'Ø123456789ABCDEF' CONVERT TO DISPLAY HEX
MVI   SENSE+4,C','        OVERLAY TRAILING BYTE
UNPK  CMDAD(7),9(4,15)    UNPACK COMMAND ADDRESS
NC    CMDAD,=15X'F'       FORCE ZONES OFF
TR    CMDAD,=C'Ø123456789ABCDEF' CONVERT TO DISPLAY HEX
MVI   CMDAD+6,C','        OVERLAY TRAILING BYTE
UNPK  STATUS(5),12(3,15)  UNPACK STATUS BYTES
NC    STATUS,=15X'F'      FORCE ZONES OFF
TR    STATUS,=C'Ø123456789ABCDEF' CONVERT TO DISPLAY HEX
MVI   STATUS+4,C','       OVERLAY TRAILING BYTE
UNPK  COUNT(5),14(3,15)   UNPACK STATUS BYTES
NC    COUNT,=15X'F'       FORCE ZONES OFF
TR    COUNT,=C'Ø123456789ABCDEF' CONVERT TO DISPLAY HEX
MVI   COUNT+4,C' '        OVERLAY TRAILING BYTE
MVC   OUTAREA+1(SYNADERL),SYNADER
BAL   11,PRINT            PRINT ERROR MESSAGE
DC    H'Ø'                FORCE DUMP
SPACE 3
```

```
*************************************************************************
*  THIS IS AN INTERNAL SUBROUTINE TO SCAN CICS TERMINAL INPUT/OUTPUT  *
*  AREAS (TIOA) INPUT FROM UNFORMATTED SCREENS.  RETURNED FIELDS ARE  *
*  ARE NON-BLANK CHARACTER STRINGS THAT ARE CONCATENATED BY AT LEAST  *
*  ONE BLANK.                                                         *
*-------------------------------------*
*  TO REDUCE INSTRUCTION PATH LENGTH IT NEITHER SAVES                 *
*  REGISTERS NOR USES CONVENTIONAL CALLING SEQUENCE.                  *
*-------------------------------------*
*  USAGE:                                                             *
*                                                                     *
*    1)  TO SCAN FOR FIELD SEPARATED BY ' ', ',', ''', '(' OR ')'     *
*                                                                     *
*        LA    4,NULL              LOAD ADDRESS OF EOB RETURN         *
*        BAL   14,KHNSCAN          SCAN FOR NEXT INPUT FIELD          *
*                                                                     *
*    2)  TO VALIDATE NUMERIC FIELDS:                                  *
*                                                                     *
*        LA    4,ERROR             LOAD ADDRESS OF NON-NUMERIC RETURN *
*        BAL   14,NUMTEST          CHECK FIELD FOR NUMERIC DATA       *
*-------------------------------------*
*  REGISTER USAGE:                                                    *
*                                                                     *
*    1)  FOR KHNSCAN, CONTENTS OF REGISTER 1 IS USED AS               *
*        A WORK REGISTER AND IS NOT RESTORED.                         *
*    2)  ON ENTRY TO KHNSCAN AND NUMTEST, THE FOLLOWING ASSUMPTIONS   *
*        ARE MADE:  REGISTER 6 CONTAINS THE ADDRESS OF THE CURRENT    *
*        FIELD; REGISTER 7, THE LENGTH - 1 OF THAT FIELD; REGISTER 8, *
*        THE REMAINING LENGTH OF THE TIOA.                            *
*    3)  ON RETURN, KHN1SCAN AND KHNSCAN, REGISTERS 6-8 ARE SET TO    *
*        THOSE VALUES DEFINED IN "2)".                                *
*    4)  FOR NUMERIC FIELDS, NUMTEST PACKS THE FIELD INTO 'PACKWORK'. *
*        ELSE, THIS FIELD IS INITIALIZED TO ZERO.                     *
*************************************************************************
         SPACE
KHNSCAN  MVC   TRTAB,TRTAB-1       SET TABLE TO NON ZERO
         MVI   TRTAB+C' ',Ø        CLEAR BLANK POSITION
         XR    1,1                 CLEAR REGISTER (HIGH ORDER BYTE)
         LA    6,1(6,7)            POINT PAST LAST FIELD
PRESCAN  CLI   Ø(6),C'='           EQUAL SIGN?
         BE    SPECIAL              YES
         CLI   Ø(6),C'+'           PLUS SIGN?
         BNE   NOTPLUS              NO
         MVI   SIGN,X'C'           SET SIGN
         B     SPECIAL             GO ADJUST POSITION AND LENGTH
NOTPLUS  CLI   Ø(6),C'-'           PLUS SIGN?
         BNE   NOTMINUS             NO
         MVI   SIGN,X'D'           SET SIGN
         B     SPECIAL             GO ADJUST POSITION AND LENGTH
NOTMINUS CLI   Ø(6),C'('           OPEN PARENTHSES?
         BNE   NOTLEFT              NO
```

```
         AP    NESTS,=P'1'         INCREMENT NESTING COUNT
         B     SPECIAL             GO ADJUST POSITION AND LENGTH
NOTLEFT  CLI   Ø(6),C')'           RIGHT PARENTHESIS?
         BNE   NOTRIGHT             NO
         SP    NESTS,=P'1'         DECREMENT NESTING COUNT
         B     SPECIAL             GO ADJUST POSITION AND LENGTH
NOTRIGHT CLI   Ø(6),C''''          WAS FIELD FOLLOWED BY A QUOTE?
         BNE   NOTQUOTE             NO
         XI    SWITCHES,QUOTEBIT   FLIP QUOTE BIT
         B     SPECIAL             GO ADJUST POSITION AND LENGTH
NOTQUOTE CLI   Ø(6),C','           IS CURRENT POSITION A COMMA?
         BNE   NONSPCL              NO
         CLI   1(6),C' '           DESIGNATES CONTINUATION?
         BNE   SPECIAL              NO
         TM    SWITCHES,QUOTEBIT   INSIDE A QUOTATION?
         BO    SPECIAL              YES
         OI    SWITCHES,CONTBIT     NO, IE A CONTINUATION INDICATION
         BR    4                   GIVE NULL RETURN (BYPASS ANY COMMNTS
         SPACE
SPECIAL  LA    6,1(6)              SKIP PAST SPECIAL CHARACTER
         BCTR  8,Ø                 DECREMENT LENGTH
         LTR   8,8                 END OF CARD?
         BMR   4                    YES
         B     PRESCAN             GO PROCESS NEXT CHARACTER
NONSPCL  EX    8,TRT               SEARCH FOR FIRST NON BLANK
         BCR   8,4                 EXIT IF NOT FOUND
         LR    7,1                 ADDRESS OF FIRST NON-BLANK
         SR    7,6                 DEDUCT ADDRESS OF LAST POSITION
         SR    8,7                 SUBTRACT LENGTH FROM TOTAL LENGTH
         BCR   4,4                 EXIT IF NEGATIVE
         LR    6,1                 POINT TO FIRST NON BLANK
         CLI   Ø(6),C''''          QUOTATION AT BEGINNING?
         BE    PRESCAN              YES, RECIRCULATE
         CLI   Ø(6),C'('           OPEN PAREN AT BEGINNING?
         BE    PRESCAN              YES, RECIRCULATE
         CLI   Ø(6),C','           NULL FIELD AT BEGINNING?
         BE    PRESCAN              YES, RECIRCULATE
         CLI   Ø(6),C'+'           UNARY PLUS SIGN AT BEGINNING?
         BE    PRESCAN              YES, RECIRCULATE
         CLI   Ø(6),C'-'           UNARY MINUS SIGN AT BEGINNING?
         BE    PRESCAN              YES, RECIRCULATE
         XC    TRTAB,TRTAB         SET TABLE TO ZEROS
         MVI   TRTAB+C' ',C' '     TURN ON BLANK POSITION
         MVI   TRTAB+C',',C','     TURN ON COMMA POSITION
         MVI   TRTAB+C'''',C''''   TURN ON C'''' POSITION
         MVI   TRTAB+C'(',C'('     TURN ON C'(' POSITION
         MVI   TRTAB+C')',C')'     TURN ON C')' POSITION
         MVI   TRTAB+C'=',C'='     TURN ON C'=' POSITION
         LR    15,8                SAVE CURRENT LENGTH
         LR    Ø,6                 SAVE CURRENT LOCATION
LASTSCAN EX    8,TRT               SEARCH FOR FIRST BLANK
```

```
         BZ    NOHITS              IF NO OBJECTS FOUND
         SPACE
         TM    SWITCHES,QUOTEBIT   WITHIN QUOTATION?
         BZ    SCANHIT              NO
         CLC   =C'''''',Ø(1)       IMBEDDED QUOTES?
         BNE   SCANHIT              NO
         LA    1,2(1)              STEP OVER IMBEDDED QUOTES
         AR    8,6                 ADJUSTED LENGTH=PREVIOUS LENGTH
         SR    8,1                   +(PREVIOUS-CURRENT)LOCATION
         LR    6,1                 RESET CURRENT POSITION
         BP    LASTSCAN            IF POSITIVE LENGTH, CONTINUE SCAN
         LR    6,Ø                 RESTORE ORIGINAL LOCATION
         LR    8,15                RESTORE ORIGINAL LENGTH
         B     SCANHIT             GO PROCESS
NOHITS   LR    6,Ø                 RESTORE ORIGINAL LOCATION
         LR    8,15                RESTORE ORIGINAL LENGTH
         LA    1,Ø(6,8)            POINT TO END OF INPUT
SCANHIT  LR    7,1                 LOAD ADDRESS OF BLANK
         SR    7,6                 SUBTRACT ADDRESS OF FIRST NON-BLANK
         BCR   13,4                NULL IF NOT POSITIVE
         SR    8,7                 DEDUCT FROM TOTAL LENGTH
         BCTR  7,14                RETURN
         BR    14                  RETURN
         SPACE 2
TRTAB    DS    CL256               MUST IMMEDIATELY FOLLOW NON-ZERO
         DC    1ØX'Ø'
TRT      TRT   Ø(*-*,6),TRTAB
TESTNUM  TRT   Ø(*-*,6),TRTAB+16
         SPACE 5
NUMTEST  MVC   TRTAB,TRTAB-1       FILL WITH NON ZEROS
         EX    7,TESTNUM           IS FIELD NUMERIC?
         BCR   7,4                   NO
         EX    7,PACK              PACK FIELD
         NI    PACKWORK+L'PACKWORK-1,X'FØ' MASK SIGN BITS OFF
         OC    PACKWORK+L'PACKWORK-1(1),SIGN TURN SIGN BITS ON
         BR    14                  RETURN
PACK     PACK    PACKWORK,Ø(*-*,6)
PACKWORK DS    CL16
         SPACE 2
DOUBLE   DC    D'Ø'
HALF     DC    H'Ø'
DIRSPACE DS    H
DIRENTRY DS    A
SAVE5TO9 DS    5F
MEMBERS  DC    PL3'Ø'
DATES    DC    PL3'Ø'
CARDS    DC    PL3'Ø'
NESTS    DC    PL2'Ø'
PATTERN  DC    X'2Ø2Ø2Ø2Ø2Ø212Ø'
MEMBER   DC    XL8'Ø'
```

```
NEWDATE  DC    CL1Ø' '
OPTIONS  DC    X'Ø'
CHNGBIT  EQU   X'4Ø'
DIAGBIT  EQU   X'2Ø'
LISTBIT  EQU   X'1Ø'
BFOREBIT EQU   X'Ø8'
AFTERBIT EQU   X'Ø4'
MEMBRBIT EQU   X'Ø2'
SWITCHES DC    X'Ø'
QUOTEBIT EQU   X'8Ø'
COMMABIT EQU   X'4Ø'
CONTBIT  EQU   X'2Ø'
PARMBIT  EQU   X'1Ø'
UPDATBIT EQU   X'Ø8'
PGMBIT   EQU   X'Ø4'
DATEBIT  EQU   X'Ø1'
CONTROL  DC    PL2'Ø'
LEAPFLAG DC    X'Ø'
SIGN     DC    X'C'
DAYS     DS    PL2
MONTHS   DS    PL2
PAGES    DC    PL2'Ø'
MMDDCCYY DS    CL8
YEARS    DC    D'Ø'
SAVEDAYS DS    D
TODAY    DS    F
         DC    PL2'Ø'              MUST IMMEDIATELY PRECEED 'JANUARY'
JANUARY  DC    P'31'
*                    M  A  M  J  J  A  S  O  N
FEBRUARY DC    P'28,31,3Ø,31,3Ø,31,31,3Ø,31,3Ø'
DECEMBER DC    P'31'
SYNADER  DC    C'SYNAD ERROR:  REG 1='
SYNADR1  DC    C'XX',C', ECB='
SYNADECB DC    C'XXXX',C', IOB='
IOBW1    DC    C'XXXXXXXX',C' '
IOBW2    DC    C'XXXXXXXX',C' '
IOBW3    DC    C'XXXXXXXX',C' '
IOBW4    DC    C'XXXXXXXX',C', SENSE='
SENSE    DC    C'XXXX',C', COMMAND ADDR='
CMDAD    DC    C'XXXXXX',C', STATUS='
STATUS   DC    C'XXXX',C', COUNT='
COUNT    DC    C'XXXX',C' '
SYNADERL EQU   *-SYNADER
SAVENAME DS    CL8
         DC    C' NO DATES CHANGED '
OUTAREA  DC    CL133'Ø'
         ORG   OUTAREA+2
MEMBERNO DS    CL4,C
MEMBNAME DS    CL8
CARDNO   DS    CL6,C
```

```
INAREA   DS    CL8Ø,C
TESTOPTS DS    CL2,C
TESTSWTS DS    CL2,C
TESTLEN  DS    CL4,C
TESTLOC  DS    CL4,C
         ORG
         DS    ØD
HEADING  DC    CL133'1'
         ORG   HEADING+1
         DC    C'DATE= CARD UPDATES'
         DS    8C
DATE     DC    C'MM/DD/CCYY'
         DS    6C
         DC    C'PARM=('
PARMHEAD DS    CL7Ø
         DC    C'—'
         ORG   HEADING+124
         DC    C'PAGE'
PAGENO   DS    CL4
         ORG
TTRN     DS    F
         LTORG
         PUSH  PRINT               SAVE CURRENT PRINT OPTIONS
         PRINT GEN                 PRINT EXPANDED MACRO
         READ  DECBA,SF,PDS,BLOCK,MF=L DEFINE DECBA
         POP   PRINT               REINSTATE PREVIOUS PRINT OPTIONS
PDSDIR   DCB   DDNAME=PDS,DSORG=PS,MACRF=GM,SYNAD=BOMB,BLKSIZE=256,    -
               EODAD=DIREND,RECFM=F,LRECL=256
PDS      DCB   DDNAME=PDS,DSORG=PO,MACRF=R,SYNAD=BOMB,                 -
               EODAD=NEXTMEMB
PRINTER  DCB   DDNAME=PRINTER,DEVD=DA,DSORG=PS,LRECL=133,              -
               BLKSIZE=133,MACRF=(PM),RECFM=FBA
         DS    ØD
DIRBLOCK DS    256C
BLOCK    DS    3120C
         PRINT GEN
         DCBD  DSORG=PO,DEVD=DA
         END   BEGIN
/*
```

Keith H Nicaise
Technical Services Manager
Touro Infirmary (USA)

Problem diagnosis in MVS 'early' code

THE PROBLEM

Recently we encountered problems at subsystem initialization, which proved difficult to diagnose. The reason for the difficulty was because, if the MVS 'early' code abends, then no system dump is taken. Instead control is passed to IEFJSBLD, which produces a record in SYS1.LOGREC. The record in SYS1.LOGREC does not provide any clues as to why the system has terminated.

THE SOLUTION

We were provided with the following ZAP which causes MVS (IEFJSBLD) to take a system dump, when an abend occurs during subsystem initialization.

```
++USERMOD(DB2ERLY).
++VER(Z038) FMID(HBB4430).
  NAME IEEVIPL IEFJSBLD.
  VER 0CFE 47E0,CD1E
  REP 0CFE 4700,CD1E
```

Our problem with subsystem initialization turned out to be that we were using the wrong level early code. The above ZAP enabled us to pinpoint this immediately.

Clifton Hunt
Systems Programmer (UK)

MVS news

Amdahl has announced Version 1 Release 3 of its TDMF (Transparent Data Migration Facility) software for transferring data between storage devices regardless of vendor or model. Improvements include dynamic pacing with better memory and I/O management, extended distance migration, volume copy, and interfaces to automate operations packages. Uses of the feature include data centre moves and migrations, load balancing and continuous availability, year 2000 activity support, currency conversion, data mining application check-out, plus scheduled non-disruptive maintenance of DASD.

For further information contact:
Amdahl Corp , East Arques Avenue, PO Box 3470, Sunnyvale, CA 94088-3470, USA.
Tel: 408 737 5565
Fax: 408 992 3220

* * *

IBM has begun shipping Version 2 Release 4 of Tivoli NetView Performance Monitor, which integrates with TME 10 NetView for OS/390 automation and management, and supports both SNA-based hardware and software and TCP/IP. The integration with TME 10 NetView for OS/390 allows for a single view for managing system and network availability.

Among the enhancements is a TCP/IP session collection capability, which collects data for a complete VTAM through TCP/IP session. It's available for TN3270 and TN3270E servers running OS/390 Release 5 or higher.

Using the monitoring capability for Multinode Persistent Sessions (MNPS), VTAM can preserve sessions across application outages, where hosts are connected through the OS/390 coupling facility. MNPS provides for the recovery of VTAM, MVS, or hardware failures.

Contact your local IBM representative for further information.

* * *

IBM has begun shipping Version 1.3 of Maintenance 2000 mainframe-based source code and JCL cross-reference analysis tool. It now has support for the VisualAge COBOL MLE by generating data identification files for CCCA. There are improvements to its impact analysis and search function, and there is continuing support for PL/I, CA-Easytrieve Plus, and COBOL.

The software integrates with CCCA for OS/390, MVS, and VM Version 2, which uses the data identification files generated by Version 1.3. It also integrates with VisualAge COBOL Millennium Language Extensions for OS/390 and VM.

Also new are program correlation chart enhancements, showing the whole programs having the specified system IDs and subsystem IDs, along with a cross-reference list enhancement, providing the resource information from a job point of view. There are also new search function enhancements.

Contact your local IBM representative for further information.

* * *