# 29 VSE

*March 1998*

## In this issue





update

# VSE Update


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

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

## Australian office
Xephon/RSM
GPO Box 6258
Halifax Street
Adelaide, SA 5000
Australia
Telephone: 88 223 1391


## Editorial panel
Articles published in *VSE Update* are reviewed by our panel of experts. Members of the panel include Stanley Stewart (USA), Robert Botsis (USA), and Jesse Joyner (USA).

## Contributions
Articles published in *VSE Update* are paid for at the rate of £170 ($250) per 1000 words for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.


## Editor
Fiona Hewitt


## 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.

## Subscriptions and back-issues
A year's subscription to *VSE Update*, comprising four quarterly issues, costs £100.00 in the UK, $150.00 in the USA and Canada, £106.00 in Europe, £112.00 in Australasia and Japan, and £110.50 elsewhere. In all cases the price includes postage. Individual issues starting with the March 1991 issue, are available separately to subscribers for £25.00 ($37.50) each including postage.

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




*Printed in England.*

# Year 2000 file expiry

What will happen to your disk files at midnight on 31 December 1999? Although VSE has never just made files disappear when they become ‘expired’, they are vulnerable to being overwritten. The current release of VSE does not offer a utility to alter disk file expiry dates, and I prefer not to have to recreate files such as VSE SYSRES LIBRARY and VSAM catalogs that I consider important on 1-1-2000.

The VTOC for 3380 and 3390 DASD contains date fields in Julian format for creation and expiry dates. The utility presented here was written under VSE/ESA 2.1 to alter the year of expiry for disk files on the IBM RAMAC subsystem in 3390 format. The access logic was copied from another VTOC utility program that I developed on 3380 DASD several years ago, so I have limited the function to these two DASD types. It should work on other CKD-type DASD, but may need some modifications to the channel programs.

UTILITY

```
TITLE 'VTOC UTILITY TO ALTER FILE EXPIRATION YEARS     '
Y2K      CSECT
RØ       EQU   Ø
R1       EQU   1             CCB ADDRESS AND MISC WORK
R2       EQU   2             BASE REGISTER 1
R3       EQU   3             *
R4       EQU   4             *
R5       EQU   5             WORK REGISTER
R6       EQU   6             *
R7       EQU   7             LOOP COUNT FOR BCT
R8       EQU   8             LINK RETURN
R9       EQU   9             *
R1Ø      EQU   1Ø            *
R11      EQU   11            *
R12      EQU   12            *
R13      EQU   13            *
R14      EQU   14            MISC WORK AND MACRO LINK RETURN
R15      EQU   15            ERROR RETURN CODES
         BALR  R2,RØ               SET UP MY BASE REGISTER
         BCTR  R2,RØ               AND ADJUST TO MATCH
         BCTR  R2,RØ               CSECT ENTRY POINT
         USING Y2K,R2
```

```
         TM    Ø(R1),X'8Ø'         NULL PARM STRING ?
         BZ    ERRORØ              YES, SKIP
         L     R5,Ø(R1)            POINT TO PARM DATA
         SLL   R5,8                CLEAR HIGH
         SRL   R5,8                 ORDER BYTE
         MVC   PARM,2(R5)          SAVE PARM DATA
         B     GO1                 SKIP OVER DATA
         DS   ØD
PGMID    DC    CL32'FXP    1.1.7  ASOF 1Ø/16/1997   '
PARM     DC    CL8Ø'                                '
GO1      MVC   CCW9(1),SKIP1           SKIP TO TOP OF FORM
         MVC   PRT,PRT-1               CLEAR PRINT AREA
         BAL   R8,PRINTER              LOG TO SYSLST
         MVC   CCW9(1),SPACE1          WRITE AND SPACE 1 LINE
*        DETERMINE DASD DEVICE TYPE 338Ø OR 339Ø
         GETVCE AREA=AVRADR,DEVICE=SYSØ1Ø,LENGTH=74
         CLI   AVRCDCST,X'8Ø'          IS IT READ ONLY ?
         BE    ERROR5                  YUP, GUESS WHAT ...
* NOTE>>>THIS LOGIC HAS BEEN RUN ON 338Ø AND 339Ø DASD BUT NONE OTHER!
* NOTE>>>IT SHOULD WORK ON OTHER CKD DEVICES, BUT HAS NOT BEEN TESTED!
         CLC   DCTEXTCD(4),=C'338Ø'    THEREFORE, IF THE DEVICE
         BE    GO2                     TYPE IS 338Ø OR 339Ø
         CLC   DCTEXTCD(4),=C'339Ø'    THATS FINE,
         BE    GO2                     CONTINUE,
         B     ERROR2                  OTHERWISE, CONSIDER IT ERROR!
*        READ VOLUME 1 LABEL AND LOCATE THE VTOC
GO2      LA    R1,DISK1                ADDR OF INPUT DISK CCB
         BAL   R8,EXCP                 PERFORM I/O
         MVC   VTOCLOC+2(5),DATA+11    SAVE CHR OF VTOC
         MVC   VOLID1,DATA+4           SAVE VOLUME ID OF INPUT DASD
         ICM   R1,B'ØØ11',VTOCLOC+2    GET THE CYLINDER NUMBER
         SLL   R1,16                   CLEAR THE HIGH
         SRL   R1,16                    ORDER BITS
         CVD   R1,D                    CONVERT 2 PACKED DECIMAL
         OI    D+7,X'ØF'               FIX THE SIGN
         UNPK  D+8(4),D+5(3)           CYL NUM OF VTOC
         ICM   R1,B'ØØØ1',VTOCLOC+5    GET THE HEAD NUMBER
         SLL   R1,26                   CLEAR THE HIGH
         SRL   R1,26                    ORDER BITS
         CVD   R1,D                    CONVERT 2 PACKED DECIMAL
         OI    D+7,X'ØF'               FIX THE SIGN
         UNPK  F(2),D+6(2)             TRK/HEAD NUM OF VTOC
         MVC   PRT,PRT-1               CLEAR PRINT AREA
         MVC   PRT(32),PGMID           IDENTIFY MY SELF
         BAL   R8,PRINTER               ON SYSLST
         LA    R1,CONSOLE               AND THE OPR
         BAL   R8,EXCP                  CONSOLE.
         MVC   PRT,PRT-1
         MVC   PRT(6),VOLID1           VOLUME SERIAL FROM DISK READ
         MVC   PRT+7(1Ø),=C'DASD TYPE='
         MVC   PRT+17(4),DCTEXTCD      DEVICE TYPE FROM GETVCE
```

```
         MVC   PRT+23(28),=C'VTOC LOCATION=CYL NNNN HD NN'
         MVC   PRT+41(4),D+8
         MVC   PRT+49(2),F
         MVC   PRT+52(9),=C'RECS/TRK='
*        READ THE FIRST VTOC RECORD WHICH SHOULD BE TYPE=4
         LA    R1,DISK2                ADDR OF DASD CCB
         BAL   R8,EXCP                 PERFORM I/O
         CLI   F4,X'F4'                FORMAT 4 LABEL (VTOC DESC) ???
         BNE   ERROR1                  NO, BAD NEWS
         MVI   VRPTRK,X'ØØ'            CLEAR HIGH END
         MVC   VRPTRK+1(1),F4BLKS      VTOC BLOCKS PER TRACK
         LH    R5,VRPTRK               IN A REGISTER FOR
         CVD   R5,D                    CHANGE TO PACKED
         AH    R5,=H'1'                ADDING TO IT AND
         STH   R5,VRPTRK               SAVE IT FOR COMPARISON LATER.
         OI    D+7,X'ØF'               FIX THE SIGN
         UNPK  PRT+61(3),D+6(2)        VTOC RECS PER TRACK
         BAL   R8,PRINTER
         LA    R1,CONSOLE
         BAL   R8,EXCP
         MVC   VTOCEND,F4VTOCE         SAVE VTOC END CYL/HEAD ADDR
         TM    AVRFLAG,X'4Ø'           SHARED DEVICE ?
         BZ    GO2A                    NO, SKIP ON
         MVC   PRT,PRT-1               CLEAR I/O AREA
         MVC   PRT(39),=C'SHARED DEVICE! ENTER "YES" TO CONTINUE.'
         LA    R1,CONSOLE
         BAL   R8,EXCP
         MVC   PRT,PRT-1               CLEAR I/O AREA
         MVI   CCW8,X'ØA'              READ INQUIRY FOR CONSOLE
         LA    R1,CONSOLE              GET RESPONSE FROM THE
         BAL   R8,EXCP                  OPERATOR CONSOLE
         MVI   CCW8,X'Ø9'              RESET FOR WRITE COMMAND
         OC    PRT(3),=X'4Ø4Ø4Ø'       FORCE UPPER CASE
         CLC   PRT(3),=C'YES'          WHAT HE SAY ?
         BNE   ERROR6                  NO, CANT CONTINUE
GO2A     BAL   R8,VLOCK                GO LOCK 1ST TRACK OF VTOC
*        VALIDATE PARAMETER INFORMATION
*        YEAR=yyyy,file id up to 44 bytes
*        PERM=ALL
         CLC   PARM(5),=C'YEAR='       MODIFY YEAR OF EXPIRATION ?
         BE    MODFY1                  YUP, EXIT
         CLC   PARM(8),=C'PERM=ALL'    MODIFY YEAR OF EXPIRATION ?
         BE    MODFY2                  YUP, EXIT
         B     ERRORØ
MODFY1   EQU   *   MODIFY THE FILE EXPIRY DATE
         BC    Ø,MODFY1A               1ST TIME NOOP
         CLC   PARM+5(2),=C'19'        VALIDATE CENTURY
         BE    GO3                     OK FOR NOW
         CLC   PARM+5(2),=C'2Ø'        VALIDATE CENTURY
         BE    GO3                     OK FOR LATER
         B     ERROR4                  BAD CENTURY
```

```
GO3      LA    R1Ø,PARM+1Ø             START OF 44 BYTE FILEID
         L     R7,=F'54'               MAX LOOP COUNT
         CLI   Ø(R1Ø),X'4Ø'            IS IT BLANK ?
         BH    *+4+4                   GREATER THAN, LEAVE IT ALONE
         MVI   Ø(R1Ø),X'4Ø'            LESS THAN, BLANK OUT JUNK
         AH    R1Ø,=H'1'               BUMP ADDRESS
         BCT   R7,*-4-4-4-4            LOOP BACK TO COMPARE
         MVC   PRT,PRT-1
         MVC   PRT(64),PARM            DISPLAY EXEC PARAMETER
         BAL   R8,PRINTER
         MVI   MODFY1+1,X'FØ'          CHANGE NOOP TO BRANCH
MODFY1A  LA    R1,DISK2                ADDR OF DASD CCB
         BAL   R8,EXCP                 PERFORM I/O
         LA    R8,MODFY1A              SET RETURN ADDRESS
         CLI   VTOCDATA,X'F1'          FORMAT 1 LABEL  ???
         BNE   BUMP1                   NO, NEVER MIND
         CLC   VTOCKEY,PARM+1Ø         FILE ID MATCH ?
         BNE   BUMP1                   NO, GET NEXT ONE!
         PACK  D,PARM+5(4)             GET 4 DIGIT YEAR
         CVB   R1Ø,D                   HEX IN A REGISTER
         SH    R1Ø,BASEYR              MINUS 19ØØ
         STH   R1Ø,NEWYR               EQUALS HEX YY
         MVC   VTOCREC+56(1),NEWYR+1   PLUG INTO EXPIRY FIELD
         MVI   NRF,X'ØØ'               SET FLAG
         B     REWRITE                 AND DO THE DEED !
MODFY2   EQU   *   MODIFY THE FILE EXPIRY DATE OF ALL FILES
         BC    Ø,MODFY2A               1ST TIME NO OP
*                VTOCREC+56              RELATIVE YEAR IN VTOC
*                1997 = X'61'
*                .... = .....
*                2Ø5Ø = X'96'   52 YEARS FROM NOW... I DONT CARE !!!
         MVC   PRT,PRT-1
         MVC   PRT(8),PARM             DISPLAY EXEC PARAMETER THAT
         BAL   R8,PRINTER              IS USED THIS RUN.
         MVI   MODFY2+1,X'FØ'          UNCONDITIONAL BRANCH
MODFY2A  LA    R1,DISK2                ADDR OF DASD CCB
         BAL   R8,EXCP                 SEEK, SEARCH AND READ VTOC REC
         LA    R8,MODFY2A              SET RETURN POINT
         CLI   VTOCDATA,X'F1'          FORMAT 1 LABEL  ???
         BNE   BUMP1                   NO, NEVER MIND
         MVI   VTOCREC+56,X'96'        YEAR=2Ø5Ø
         LA    R1,DISK3                RE-WRITE/UPDATE VTOC RECORD
         BAL   R8,EXCP                 PERFORM I/O
         MVI   NRF,X'ØØ'               SET FLAG
         MVC   PRT,PRT-1               CLEAR IT OUT
         MVC   PRT(44),VTOCREC         KEY FIELD IS FILE ID
         MVC   PRT+46(15),=C'ALTERED TO 2Ø5Ø'
         UNPK  PRT+62(15),VTOCLOC(8)   HEX CCHH OF VTOC LOCATION
         MVI   PRT+76,X'4Ø'            CLEAR JUNK
         TR    PRT+62(14),TBL          MAKE IT PRINTABLE
         BAL   R8,PRINTER
```

```
         LA    R8,MODFY2A              SET RETURN POINTER
         B     BUMP1
REWRITE  EQU   *        RE-WRITE VTOC RECORD FOR UPDATE
         LA    R1,DISK3                ADDR OF DASD CCB
         BAL   R8,EXCP                 PERFORM I/O
         MVC   PRT,PRT-1
         MVC   PRT(44),VTOCREC
         MVC   PRT+46(12),=C'YEAR ALTERED'
         BAL   R8,PRINTER
         B     EOJ
BUMP1    EQU   *   INCREMENT VTOC ADDR TO NEXT REC, TRACK OR CYL
         ST    R8,SAVE8                SAVE RETURN ADDRESS
* BUMP TO NEXT RECORD
         ICM   R1,B'ØØØ1',VTOCLOC+6    GET THE RECORD NUMBER
         SLL   R1,24                   CLEAR THE HIGH
         SRL   R1,24                    ORDER BITS
         AH    R1,=H'1'                INCREMENT RECORD
         STC   R1,VTOCLOC+6            UPDATE CHR ADDR FOR CCW CHAIN
         CH    R1,VRPTRK               NUMBER VTOC RECORDS + 1
         BL    BUMPEXIT
* BUMP TO NEXT TRACK
         MVI   VTOCLOC+6,X'Ø1'         RESET TO FIRST RECORD NUMBER
         ICM   R1,B'ØØØ1',VTOCLOC+5    GET THE HEAD NUMBER
         SLL   R1,24                   CLEAR THE HIGH
         SRL   R1,24                    ORDER BITS
         AH    R1,=H'1'                BUMP TO NEXT TRACK
         STC   R1,VTOCLOC+5            UPDATE CHR ADDR FOR CCW CHAIN
         CH    R1,DCTTCYL              TRACKS PER CYLINDER ?
         BE    BUMPCYL                 YES, WE CROSSED CYL BOUNDRY
         BH    BUMPCYL                 YES, WE CROSSED CYL BOUNDRY
         BAL   R8,VLOCK                GO LOCK NEXT VTOC TRACK
         CLC   VTOCLOC+2(4),VTOCEND    ARE WE AT END OF THE VTOC ?
         BNH   BUMPEXIT                NO, KEEP ON
         B     EOJ                     YUP, ALLFRU AND OUTTA HERE
BUMPCYL  MVI   VTOCLOC+6,X'Ø1'         RESET TO FIRST RECORD NUMBER
         MVI   VTOCLOC+5,X'ØØ'         RESET TO TRACK ZERO
         ICM   R1,B'ØØ11',VTOCLOC+2    GET THE CYL NUMBER
         SLL   R1,16                   CLEAR THE HIGH
         SRL   R1,16                    ORDER BITS
         AH    R1,=H'1'                BUMP TO NEXT CYL
         STH   R1,H                    SAVE NEW CYLINDER NUMBER
         MVC   VTOCLOC+2(2),H          UPDATE CHR ADDR FOR CCW CHAIN
         BAL   R8,VLOCK                GO LOCK NEXT VTOC TRACK
         CLC   VTOCLOC+2(4),VTOCEND    ARE WE AT END OF THE VTOC ?
         BNH   BUMPEXIT                NO, KEEP ON
         B     EOJ                     YUP, ALLFRU AND OUTTA HERE
BUMPEXIT L     R8,SAVE8                LOAD RETURN ADDRESS
         BR    R8
ERRORØ   EQU   *  INVALID RUN TIME PARAMETER
         MVC   PRT,PRT-1
```

```
         MVC   PRT(37),=C'INVALID OR MISSING RUN TIME PARAMETER'
         BAL   R8,PRINTER
         B     ERROREOJ
ERROR1   EQU   *  1ST VTOC RECORD NOT DESCRIPTOR
         MVC   PRT,PRT-1
         MVC   PRT(4Ø),=C'1ST VTOC REC IS NOT DESCRIPTOR (TYPE=F4)'
         BAL   R8,PRINTER
         B     ERROREOJ
ERROR2   EQU   *  SYSØ1Ø NOT DASD
         MVC   PRT,PRT-1
         MVC   PRT(28),=C'SYSØ1Ø NOT 338Ø OR 339Ø DASD'
         BAL   R8,PRINTER
         B     ERROREOJ
ERROR3   EQU   *  LOCK FAILED
         MVC   PRT,PRT-1               CLEAR PRINT LINE
         MVC   PRT(24),=C'LOCK REQUEST FAILED R15='
         UNPK  PRT+24(2),RC(2)         MAKE HEX RETURN CODE
         MVI   PRT+25,X'4Ø'            PRINTABLE
         TR    PRT+24(1),TBL
         BAL   R8,PRINTER
         LA    R1,CONSOLE
         BAL   R8,EXCP
         MVC   PRT,PRT-1
         MVC   PRT(16),=C'LOCK ATTEMPT ON '
         MVC   PRT+16(12),LOCKVOL+4         SHOW THE 12 BYTE RESOURCE
         UNPK  PRT+3Ø(13),LOCKVOL+4(7)      NAME IN BOTH CHARACTER
         UNPK  PRT+42(13),LOCKVOL+1Ø(7)     AND HEX FORMAT.
         MVI   PRT+54,X'4Ø'
         TR    PRT+3Ø(24),TBL
         BAL   R8,PRINTER
         B     ERROREOJ
ERROR4   EQU   *  INVALID CENTURY
         MVC   PRT,PRT-1
         MVC   PRT(27),=C'CENTURY NOT 19 OR 2Ø, PARM='
         MVC   PRT+27(9),PARM
         BAL   R8,PRINTER
         B     ERROREOJ
ERROR5   EQU   *  VM READ ONLY DEVICE
         MVC   PRT,PRT-1
         MVC   PRT(6),AVRVOLID
         MVC   PRT+7(24),=C'IS A READ-ONLY VM DEVICE'
         BAL   R8,PRINTER
         B     ERROREOJ
ERROR6   EQU   *  SHARED DEVICE - NO VALID RESPONSE
         MVC   PRT,PRT-1
         MVC   PRT(6),AVRVOLID
         MVC   PRT+7(18),=C'IS A SHARED DEVICE'
         MVC   PRT+26(3Ø),=C'AND OPR DID NOT RESPOND "YES".'
         BAL   R8,PRINTER
         MVC   PRT,PRT-1
```

```
         MVC   PRT(31),=C'TERMINATED DUE TO OPR RESPONSE!'
         B     ERROREOJ
ERROREOJ EQU   *
         LA    R1,CONSOLE              LOG ERROR TO CONSOLE
         BAL   R8,EXCP
         MVC   PRT,PRT-1
         MVC   PRT(25),=C'FXP - VTOC UPDATE ABORTED'
         BAL   R8,PRINTER
         LA    R1,CONSOLE              LOG ERROR TO CONSOLE
         BAL   R8,EXCP
         B     EOJ
VLOCK    EQU   *   ATTEMPT TO LOCK THE DASD VOLUME FOR EXCLUSIVE USE
         ST    R8,SAVE8+4              SAVE RETURN ADDRESS
         MVC   LOCKVOL+16(6),VOLID1
         MVI   LOCKVOL+4,C'V'          YUP, THIS VIOLATES THE RULE!
         MVC   LOCKVOL+5(6),VOLID1     VOLUME SERIAL ID
         MVC   LOCKVOL+11(5),VTOCLOC+1 CCCHH OF VTOC
         L     R7,=F'7'                SET MAX RETRY ATTEMPTS
VLOCK1   LOCK  LOCKVOL,FAIL=RETURN
         STC   R15,RC                  SAVE RETURN CODE
         CLI   RC,X'Ø4'                RESOURCE NOT AVAIL ?
         BE    PAUSE2                  TRUE, GO WAIT 2 SECONDS
         CLI   RC,X'ØØ'                OTHER ERROR/LOCK FAILURE ?
         BNE   ERROR3                  YUP, BAIL OUT NOW.
         L     R8,SAVE8+4              LOAD RETURN ADDRESS
         BR    R8                      AND GO THERE
PAUSE2   SETIME 2,TECB
         LA    R1,TECB                 WAIT 2 SECONDS THEN
         BAL   R8,WAIT                 GO RETRY THE LOCK
         BCT   R7,VLOCK1               BUT ONLY 7 TIMES.
         B     ERROR3
PRINTER  LA    R1,PRINT                POINT TO CCB
EXCP     SVC   Ø                       START I/O OPERATION
WAIT     TM    2(R1),X'8Ø'             POSTED COMPLETE ?
         BO    Ø(R8)                   YES, RETURN
         SVC   7                       NO, WAIT
*        LOCKS WILL BE RELEASED AT EOJ STEP (THIS PHASE)
EOJ      MVC   PRT,PRT-1
         MVC   PRT(22),=C'FILE NOT FOUND IN VTOC'
         MVI   PRT+23,C'*'
         MVC   PRT+24(46),PRT+23
         CLI   NRF,X'FF'               CHECK FLAG SETTING
         BNE   *+4+4
         BAL   R8,PRINTER
         SVC   14                      END OF PROGRAM EXECUTION
         LTORG
*        THE "NAME" AND "VOLID" FIELDS WILL BE CHANGED AT EXEC TIME!
LOCKVOL  DTL   NAME=BUBBAWARE,CONTROL=E,KEEP=NO,LOCKOPT=1,VOLID=SAME2U
NRF      DC    X'FF'          FLAG FOR VTOC FILE ID NOT FOUND
RC       DS    X              RETURN CODE
```

```
CCHH     DC    XL7'00000000000003'          CYL Ø  TRK Ø  REC 3
VTOCLOC  DC    XL7'00000000000000'
VTOCEND  DS    XL4                          END CCHH OF VTOC
VOLID1   DS    CL6           VOLUME NAME OF INPUT DASD
CKD      EQU   *
COUNT    DS    XL8
KEY      DS    CL4
DATA     DS    CL8Ø
VRPTRK   DS    H             VTOC RECORDS PER TRACK
         DC    X'FØF1F2F3F4F5F6F7F8F9C1C2C3C4C5C6'
TBL      EQU   *-256
H        DS    H                      HALFWORD WORK AREA
BASEYR   DC    H'19ØØ'
NEWYR    DS    H
TECB     DC    4X'ØØ'
SAVE7    DS    F                      REGISTER 7 SAVE AREA
SAVE8    DS    2F                     REGISTER 8 SAVE AREA
SAVE9    DS    F                      REGISTER 9 SAVE AREA
F        DS    F                      FULLWORD WORK AREA
D        DS    D                      DOUBLEWORD WORK AREA
         DS    D
DISK1    CCB   SYSØ1Ø,CCW1            READ VOLUME 1 RECORD
DISK2    CCB   SYSØ1Ø,CCW2            SEEK, SEARCH AND READ VTOC
DISK3    CCB   SYSØ1Ø,CCW3            SEEK, SEARCH AND REWRITE VTOC
CCW1     CCW   X'Ø7',CCHH,X'4Ø',6          SEEK
         CCW   X'31',CCHH+2,X'4Ø',5        SEARCH ID EQ
         CCW   X'Ø8',*-8,4Ø,Ø              TIC
         CCW   X'ØE',KEY,X'2Ø',84          READ KEY AND DATA
CCW2     CCW   X'Ø7',VTOCLOC,X'4Ø',6       SEEK
         CCW   X'31',VTOCLOC+2,X'4Ø',5     SEARCH ID EQ
         CCW   X'Ø8',*-8,4Ø,Ø              TIC
         CCW   X'ØE',VTOCREC,X'ØØ',14Ø     READ KEY AND DATA
CCW3     CCW   X'Ø7',VTOCLOC,X'4Ø',6       SEEK
         CCW   X'31',VTOCLOC+2,X'4Ø',5     SEARCH ID EQ
         CCW   X'Ø8',*-8,4Ø,Ø              TIC
         CCW   X'ØD',VTOCREC,X'ØØ',14Ø     RE-WRITE KEY AND DATA
CONSOLE  CCB   SYSLOG,CCW8
CCW8     CCW   X'Ø9',PRT,X'2Ø',7Ø
PRINT    CCB   SYSLST,CCW9
CCW9     CCW   X'Ø9',PRT,X'ØØ',13Ø
SPACEØ   DC    X'Ø1'                  WRITE WITHOUT SPACING
SPACE1   DC    X'Ø9'                  WRITE AND SPACE 1
SPACE2   DC    X'11'                  WRITE AND SPACE 2
SPACE3   DC    X'19'                  WRITE AND SPACE 3
SKIP1    DC    X'8B'                  SKIP CHANNEL 1 IMMEDIATE
         AVRLIST DSECT=NO,DEVICE=YES
         DS    CL48
         DC    X'4Ø'
PRT      DS    CL13Ø
F4VTOCE  EQU   VTOCREC+111   XL4
VTOCREC  DS   ØCL14Ø         VTOC RECORD
```

```
VTOCKEY  DS    XL44          FILE ID FOR FORMAT 1 RECORDS
F4BLKS   EQU   VTOCDATA+30   XL1
VTOCDATA DS    CL96
F1       EQU   VTOCDATA      X'F1' = FORMAT 1 LABEL RECORD TYPE
F4       EQU   VTOCDATA      X'F4' = FORMAT 4 LABEL RECORD TYPE
         END
```

EXAMPLE 1

```
* alter all files to permanent on volume 100=DOSRES
// ASSGN SYS010,100
EXEC PGM=FXP,SIZE=12K,PARM='PERM=ALL'
```

EXAMPLE 2

```
* alter specific file
// ASSGN SYS010,100
EXEC PGM=FXP,SIZE=12K,PARM='YEAR=1997,CICS TEMP DUMP A'
/&
```

*Robert Payne*
*Senior Systems Programmer (USA)*


## Convert date subroutine

This subroutine converts a job, system, or user date in the form MMDDYY or MMDDCCYY into one of nine different formats, depending on a value in the second parameter, the option field.

One parameter must be passed, and there are two additional optional parameters.

### FIRST PARAMETER

The first parameter is used to indicate what type of date is to be converted. If formats 'U' or '2' (see below) are indicated, it must contain the actual date to be converted. It also contains the result of the date conversion on return to the calling program. Its length is variable, and depends on the value specified or defaulted in the second parameter, the option byte (see below).

The first byte of this field must contain one of the following, indicating the type of data to be converted:

- 'C' – indicates that the hardware time-of-day (TOD) clock value, inserted by the STCK instruction as MMDDYYCC, is to be converted via the GETIME macro with the CLOCK=YES operand.
- 'J' – indicates that the date in the partition communications region, inserted there by a VSE '// DATE' statement, is to be converted.
- 'S' – indicates that the date in the system communications region, inserted there by an IPL, is to be converted.
- 'U' – indicates that the date in the next six bytes, in the form MMDDYY, is to be converted. The date isn't checked for validity. 'U' is the default if none of 'C', 'J', 'S', or '2' are specified.
- '2' – indicates that the date in the next eight bytes, in the form MMDDCCYY, is to be converted. The date isn't checked for validity.

SECOND PARAMETER

The second parameter is optional, and indicates the format of the date conversion. If this parameter isn't passed, or if the value passed isn't 1 through 9, the default format returned is the same as that shown for option 1 (ie MM/DD/YY). The values, formats, and lengths of the options are shown in Figure 1.

| Option | Format | Length |
|---|---|---|
| '1' | MM/DD/YY | 08 |
| '2' | Jan. 15, 1989 | 13 |
| '3' | January 15, 1989 | 19 |
| '4' | Mon. Jan. 15, 1989 | 19 |
| '5' | Monday January 15, 1989 | 29 |
| '6' | Julian format - YYDDDBB | 07 (BB=2 blanks) |
| '7' | Julian format - YYDDDCC | 07 (CC=2 byte centur) |
| '8' | Julian format - CCYYDDD | 07 (CC=2 byte centu) |
| '9' | MM/DD/CCYY | 10 (CC=2 byte centur) |

*Figure 1: Second parameter options*

THIRD PARAMETER

The third parameter is also optional. It is 8 bytes long, and indicates that the time of day is to be returned in the form HH.MM.SS. If this field contains the value '1' before the call, the time will be converted to am/pm (ie HH.MM AM) instead of the 24-hour time.

NOTES

1 If a particular option is used and the length of the first parameter, the target field, isn't long enough to contain the output, data corruption will occur.

2 All data returned is left-justified.

3 If the third parameter is passed, the second parameter must also be passed.

4 Except in option '1', which does not return the century, the century returned depends on the value of the first byte of the first parameter:

- With 'C', the current TOD clock is returned.
- With 'J', the current job date century is returned.
- With 'S', the current system date century is returned.
- With '2', the century, passed in bytes 5-6 of the first parameter, is returned.

CALLING SEQUENCES

The calling sequences are shown below.

**COBOL**

```
CALL 'DPDATE' USING DATEFLD.
```

Or

```
CALL 'DPDATE' USING DATEFLD, OPTION, TIMEFLD.
```

## ALC

```
     LA    13,SAVEAREA (13 CAN ALSO BE R13 OR RD).
                        CALL  DPDATE,(DATEFLD)
```

Or

```
     CALL  DPDATE,(DATEFLD,OPTION,TIMEFLD)
                       .  (MAINLINE PART OF PROGRAM).
     SAVEAREA DC    18F'0'
```

## RPGII

```
     CALL 'DPDATE'
                    PARM           DTEFLD
```

Or

```
                    PARM           DTEFLD
                    PARM           OPTION
                    PARM           TIMFLD
```

An 18-word save area must be passed through register 13 by the user (STD COBOL LINKAGE).

DPDATE

```
DPDA     TITLE 'DPDATE - 1.Ø - CONVERT DATE SUBROUTINE.'
DPDATE   CSECT
DPDATE   AMODE 31
DPDATE   RMODE ANY
         BALR  RF,Ø                LOAD TEMPORARY BASE.
         USING *,RF                INFORM ASSEMBLER.
         SAVE  (14,12)
         DROP  RF                  DROP TEMPORARY BASE.
         BALR  R3,Ø
         USING *,R3
         ST    RD,SAVEAREA+4
         LA    RD,SAVEAREA
         B     BEGIN
*
RØ       EQU   Ø                   EQUATE RØ TO REGISTER ZERO.
R1       EQU   1                   EQUATE R1 TO REGISTER ONE.
R2       EQU   2                   EQUATE R2 TO REGISTER TWO.
R3       EQU   3                   EQUATE R3 TO REGISTER THREE.
R4       EQU   4                   EQUATE R4 TO REGISTER FOUR.
R5       EQU   5                   EQUATE R5 TO REGISTER FIVE.
R6       EQU   6                   EQUATE R6 TO REGISTER SIX.
```

```
R7       EQU   7                   EQUATE R7 TO REGISTER SEVEN.
R8       EQU   8                   EQUATE R8 TO REGISTER EIGHT.
R9       EQU   9                   EQUATE R9 TO REGISTER NINE.
RA       EQU   1Ø                  EQUATE RA TO REGISTER TEN.
RB       EQU   11                  EQUATE RB TO REGISTER ELEVEN.
RC       EQU   12                  EQUATE RC TO REGISTER TWELVE.
RD       EQU   13                  EQUATE RD TO REGISTER THIRTEEN.
RE       EQU   14                  EQUATE RE TO REGISTER FOURTEEN.
RF       EQU   15                  EQUATE RF TO REGISTER FIFTEEN.
*
DC    C'DPDATE STARTS HERE. ' INSERT EYE CATCHER.
*
BEGIN    EQU   *
         L     R4,Ø(R1)            GET ADDRESS OF PARAMETER FIELD.
         MVC   SAVEPARM,Ø(R4)      SVE PASSED PARAMETER.
         SR    R6,R6               CLEAR REG 6.
*
PRMCOUNT EQU   *
         TM    Ø(R1),X'8Ø'         IS THIS THE LAST PARAMETER PASSED.
         BO    LASTPRM             YES-BRANCH TO LASTPRM.
         LA    R1,4(R1)            INCREMENT REG 1 BY ONE (1).
         LA    R6,4(R6)            INCREMENT REG 6 BY ONE (1).
         B     PRMCOUNT            BRANCH TO PRMCOUNT.
*
LASTPRM  EQU   *
         SR    R1,R6               RESTORE REG 1.
         SRL   R6,2                DIVIDE REG 6 BY 2.
         LA    R6,1(R6)            INCREMENT IT BY ONE (1) FOR FIRST TI
         STC   R6,NUMPRM           SVE NUMBER OF PARAMETERS PASSED.
         MVI   MONTHS+12,X'1C'     SET FEBRUARY TO 28 DAYS. (NOT LEAP Y
         MVI   OPTION,X'Ø1'        SET OPTION TO DEFAULT OF 1.
         CLI   NUMPRM,X'Ø1'        WAS ONE (1) PARAMETER PASSED.
         BE    GETDATES            YES-BRANCH TO GETDATES.
         L     R7,4(R1)            GET ADDRESS OF OPTION BYTE.
         MVC   OPTION,Ø(R7)        SVE IT.
         NI    OPTION,X'ØF'        REMOVE SIGN.
         CLI   OPTION,X'Ø9'        IS OPTION BYTE HIGHER THAN 9.
         BNH   LASTPRM3            NO-BRANCH TO LASTPRM3.
         MVI   OPTION,X'Ø1'        FORCE OPTION BYTE TO 1. (ERROR).
*
LASTPRM3 EQU   *
         CLI   OPTION,X'ØØ'        IS OPTION BYTE LOWER THAN 1.
         BH    LASTPRM5            NO-BRANCH TO LASTPRM5.
         MVI   OPTION,X'Ø1'        FORCE OPTION BYTE TO 1. (ERROR).
*
LASTPRM5 EQU   *
         SR    R7,R7               CLEAR REG 7.
         LR    R2,R7               CLEAR REG 2.
         IC    R7,OPTION           INSERT OPTION BYTE INTO REG 7.
```

```
         BCTR  R7,Ø                REDUCE IT BY ONE (1) FOR INDEXING.
         BCTR  R7,Ø                (OPTION ONE (1) HAS NO CLEAR).
         M     R6,=F'6'            MULTIPLY LENGTH OF MVC INST.
         EX    R2,CLEAR2(R7)       CLEAR TARGET AREA.
         CLI   NUMPRM,X'Ø3'        WERE THREE (3) PARAMETERS PASSED.
         BNE   GETDATES            NO-BRANCH TO GETDATES.
         L     R5,8(R1)            GET ADDRESS OF TIME FIELD.
         GETIME STANDARD           GET TIME AND PUT INTO REG 1.
         ST    R1,WORK             SVE IT.
         MVC   TARGET,PATTERN      MVE IN EDIT WORD.
         ED    TARGET,WORK         ...
         MVI   TARGET+4,C'.'       MVE PERIOD.
         MVI   TARGET+7,C'.'       ...
         OI    TARGET+9,C'Ø'       ENSURE GOOD SIGN.
         MVC   CODE,Ø(R5)          SVE CODE.
         MVC   Ø(8,R5),TARGET+2    MVE TIME.
         CLI   CODE,C'1'           IS TIME TO BE CONVERTED TO AM/PM.
         BNE   GETDATES            NO-BRANCH TO GETDATES.
         PACK  TIMEP,Ø(2,R5)       PACK HOURS.
         CP    TIMEP,=P'12'        IS IT GREATER THAN 12.
         BL    *+14                NO-SKIP NEXT TWO (2) INST.
         MVC   5(3,R5),=C' PM'     MVE PM INDICATOR TO AMPM.
         B     *+1Ø                SKIP NEXT INST.
         MVC   5(3,R5),=C' AM'     MVE AM INDICATOR TO AMPM.
         CP    TIMEP,=P'12'        IS IT GREATER THAN 12.
         BNH   *+14                NO-SKIP NEXT THREE (3) INST.
         SP    TIMEP,=P'12'        SUBTRACT 12 FROM IT.
         B     *+2Ø                SKIP NEXT THREE (3) INST.
         CP    TIMEP,=P'Ø'         IS IT EQUAL TO ZERO.
         BNE   *+1Ø                NO-SKIP NEXT INST.
         SP    TIMEP,=P'12'        SUBTRACT TWELVE (12).
         UNPK  TIMEU,TIMEP         UNPACK HOURS.
         MVZ   TIMEU(2),=C'ØØ'     ENSURE GOOD SIGN.
         MVC   Ø(2,R5),TIMEU       MVE IT BACK TO USER
         CLI   Ø(R5),C' '          IS FIRST POSITION OF HOURS BLANK.
         BNE   *+8                 NO-SKIP NEXT INST.
         MVI   Ø(R5),C'Ø'          MAKE IT ZERO (Ø).
*
GETDATES EQU   *
         COMRG                     GET COMMUNICATIONS REGION.
         USING COMREG,R1           INFORM ASSEMBLER.
         MVC   SJOBDATC,JOBDATWC   MVE JOB DATE WITH CENTURY. (X'ØØ').
         MVC   SSYSCENT,SYSCENT    MVE SYSTEM CENTURY. (X'64').
         CLI   SAVEPARM,C'J'       ARE WE TO CONVERT JOB DATE.
         BNE   GETCDATE            NO-BRANCH TO GETCDATE.
         MVC   UMONTH,JOBDATE      MVE JOB MONTH. (X'ØØ').
         MVC   UDAY,JOBDATE+3      MVE JOB DAY. (X'Ø3').
         MVC   UYEAR,JOBDATE+6     MVE JOB YEAR. (X'Ø6').
         MVC   UCENT,JOBDATE+9     MVE JOB YEAR. (X'Ø9').
```

```
         B     CHKOPTN             BRANCH TO CHKOPTN
*
GETCDATE EQU   *
         CLI   SAVEPARM,C'C'       ARE WE TO CONVERT TOD CLOCK.
         BNE   GETSDATE            NO-BRANCH TO GETSDATE.
         STCK  TOD                 STORE CLOCK VALUE TO TOD.
         LM    RØ,R1,TOD           LOAD IT TO REGS Ø AND 1.
         GETIME STANDARD,CLOCK=YES CONVERT TOD CLOCK VALUE.
         ST    RE,TODMMDD          STORE MMDD TO TODMMDD.
         ST    RF,TODYYCC          STORE MMDD TO TODYYCC.
         MVC   UMONTH(4),TODMMDD
         MVC   UYEAR(2),TODYYCC
         MVC   UCENT(2),TODYYCC+2
         B     CHKOPTN             BRANCH TO CHKOPTN
*
GETSDATE EQU   *
         CLI   SAVEPARM,C'S'       ARE WE TO CONVERT SYSTEM DATE.
         BNE   GET2DATE            NO-BRANCH TO GET2DATE.
         MVC   UMONTH,MMDD         MVE SYSTEM MONTH. (X'4F').
         MVC   UDAY,MMDD+2         MVE SYSTEM DAY. (X'51').
         MVC   UYEAR,YYDDD         MVE SYSTEM YEAR. (X'53').
         MVC   UCENT,SSYSCENT      MVE SYSTEM CENTURY. (X'64').
         DROP  R1                  (COMREG).
         B     CHKOPTN             BRANCH TO CHKOPTN
*
GET2DATE EQU   *
         CLI   SAVEPARM,C'2'       ARE WE TO CONVERT USER EIGHT (8) BYT
         BNE   GETUDATE            NO-BRANCH TO GETUDATE.
         MVC   UDATE(4),SAVEPARM+1 MVE USER DATE MONTH/DAY.
         MVC   UDATE+4(2),SAVEPARM+7 MVE USER DATE YEAR.
         MVC   UCENT,SAVEPARM+5    MVE USER DATE CENTURY.
         MVZ   UDATE(8),=C'ØØØØØØØØ' MAKE IT NUMERIC.
         MVC   SSYSCENT,UCENT      MVE USER DATE CENTURY.
         B     CHKOPTN             BRANCH TO CHKOPTN
*
GETUDATE EQU   *
         MVC   UDATE,SAVEPARM+1    MVE USER DATE.
         MVZ   UDATE,=C'ØØØØØØ'    MAKE IT NUMERIC.
         MVC   UCENT,SSYSCENT      MVE SYSTEM DATE CENTURY.
*
CHKOPTN  EQU   *
         CLI   OPTION,X'Ø1'        IS OPTION 1 REQUESTED.
         BE    ONEPRM              YES-BRANCH TO ONEPRM.
         CLI   OPTION,X'Ø9'        IS OPTION 9 REQUESTED.
         BE    GREG                YES-BRANCH TO GREG.
         PACK  DOUBWORD,UYEAR      PACK YEAR.
         CVB   R2,DOUBWORD         CONVERT IT TO BINARY.
         LTR   R2,R2               IS YEAR ZERO.
         BZ    CHKOPTN3            YES-BRANCH TO CHKOPTN3.
```

```
         LA    RA,1
         LR    R6,R2               LOAD IT TO REG 6.
         XR    R7,R7               CLEAR REG 7.
         SRDL  R6,2                SHIFT IT.
         LTR   R7,R7               IS THERE A REMAINDER.
         BNZ   CHKOPTN5            NO-BRANCH TO CHKOPTN5.
         MVI   MONTHS+12,X'1D'     SET FEB TO 29 (LEAP YEAR).
         LA    RA,2
         B     CHKOPTN5            BRANCH TO CHKOPTN5.
*
CHKOPTN3 EQU   *
         MVC   SUCENT,UCENT
         MVC   SUCENT+2(2),UYEAR
         PACK  FULLWORD,SUCENT     PACK CENTURY/YEAR.
         DP    FULLWORD,=P'4ØØ'    DIVIDE CENTURY/YEAR BY 4ØØ.
         CP    FULLWORD+2(2),=P'ØØ' IS THERE A REMAINDER. (NOT LEAP YEA
         BNE   CHKOPTN5            YES-BRANCH TO CHKOPTN5.
         LA    RA,3
         MVI   MONTHS+12,X'1D'     SET FEB TO 29 (LEAP YEAR).
*
CHKOPTN5 EQU   *
         LR    R6,R2               RESTORE REG 6.
         LTR   R2,R2
         BNZ   CHKOPTN9
         CP    FULLWORD+2(2),=P'ØØ'
         BNE   CHKOPTN7
         BCTR  R2,Ø
         LA    RA,5
         B     CHKOPTN9
*
CHKOPTN7 EQU   *
         CLI   SUCENT,C'Ø'
         BE    CHKOPTN9
         CLI   SUCENT,C'2'
         BE    CHKOPTN9
         CLI   SUCENT,C'4'
         BE    CHKOPTN9
         CLI   SUCENT,C'6'
         BE    CHKOPTN9
         CLI   SUCENT,C'8'
         BE    CHKOPTN9
         LA    R2,1(R2)
         LA    RA,7
*
CHKOPTN9 EQU   *
         LA    RA,9
         LA    R6,3(R6)            SET UP FOR ROUNDING.
         SRL   R6,2                DIVIDE BY 4.
         AR    R2,R6               ADD TO ACCUMULATOR.
```

```
         LA    RA,Ø
         SRDL  RA,2
         PACK  DOUBWORD,UMONTH     PACK MONTH.
         CVB   R6,DOUBWORD         USE IT AS AN INDEX.
         BCTR  R6,Ø                ...
         LR    RB,R6               MVE MONTH TEXT.
         M     RA,=F'12'           MULTIPLY TIMES TABLE WIDTH.
         LA    RA,MONTHS           LOAD TABLE ADDRESS TO REG 1Ø.
         AR    RB,RA               ADD TABLE ADDRESS TO DISPLACEMENT.
         MVC   MONTHFLD,Ø(RB)      MVE MONTH TEXT.
         LR    RB,R6               RESTORE REG 11.
         LA    RC,MONTHS           LOAD TABLE ADDRESS TO REG 12.
         SR    R9,R9               CLEAR REG 9.
         SR    R8,R8               CLEAR REG 8.
         SR    R6,R6               CLEAR REG 6.
         LTR   RB,RB               IS IT JANUARY.
         BZ    THISMO              YES-BRANCH TO THISMO.
*
TOTALDAY EQU   *
         IC    R8,Ø(RC)            INSERT NUMBER OF DAYS IN MONTH TO RE
         AR    R9,R8               ADD IT TO TOTAL.
         LA    RC,12(RC)           INCREMENT REG 12 TO NEXT TABLE POSIT
         LA    R6,1(R6)            INCREMENT REG 6 BY ONE (1).
         CR    R6,RB               ARE WE DONE.
         BL    TOTALDAY            NO-BRANCH TO TOTALDAY.
*
THISMO   EQU   *
         PACK  DOUBWORD,UDAY       PACK DAY.
         CVB   R8,DOUBWORD         CONVERT IT TO BINARY.
         AR    R9,R8               ADD IT TO TOTAL.
         ST    R9,JULIAN           SVE IT.
         CLI   OPTION,X'Ø6'        IS JULIAN OPTION REQUESTED.
         BE    JULIAND1            YES-BRANCH TO JULIAND1.
         CLI   OPTION,X'Ø7'        IS JULIAN OPTION W/CENT REQUESTED.
         BE    JULIAND1            YES-BRANCH TO JULIAND1.
         CLI   OPTION,X'Ø8'        IS JULIAN OPTION W/CENT REQUESTED.
         BE    JULIAND1            YES-BRANCH TO JULIAND1.
         AR    R9,R2               ADD ACCUMULATOR.
         SR    R8,R8               CLEAR REG 8.
         D     R8,=F'7'            DIVIDE TOTAL BY 7.
         LR    R9,R8               REMAINDER IS INDEX.
         M     R8,=F'1Ø'           MULTIPLY TIMES TABLE WIDTH.
         LA    R8,DAYS             LOAD TABLE ADDRESS TO REG 8.
         AR    R9,R8               ADD TABLE ADDRESS TO DISPLACEMENT.
         MVC   DAYFLD,Ø(R9)        MVE DAY.
         CLI   OPTION,X'Ø3'        IS DAY AND MONTH IN TEXT REQUESTED.
         BH    DAYMO               YES-BRANCH TO DAYMO.
         SR    R2,R2               CLEAR REG 2.
         IC    R2,MONTHFLD+1       GET NUMBER CHARACTERS FOR EX MVC.
```

```
         CLI   OPTION,X'Ø2'        IS ABBREVIATED OPTION REQUESTED.
         BNE   MOVEMON2            NO-BRANCH TO MOVEMON2.
*
MOVEMON1 EQU   *
         LA    R2,4                SET UP FOR THREE CHAR NAME PLUS PERI
         MVI   MONTHFLD+5,C'.'     ..
         MVI   MONTHFLD+6,C' '     ..
*
MOVEMON2 EQU   *
         EX    R2,MOVEMO           MVE MONTH.
         LA    R4,1(R4,R2)
         CLI   UDAY,C'Ø'           IS DAYS TEN DIGIT ZERO.
         BE    SKIPONE             YES-BRANCH TO SKIPONE.
         MVC   Ø(1,R4),UDAY        MVE TEN DIGIT.
         LA    R4,1(R4)            INCREMENT REG 4 BY ONE (1).
*
SKIPONE  EQU   *
         MVC   Ø(1,R4),UDAY+1      MVE IN UNIT DIGIT.
         MVI   1(R4),C','          MVE IN COMMA.
         MVC   3(2,R4),UCENT       MVE IN SYSTEM DATE CENTURY.
         MVC   5(2,R4),UYEAR       ..
         B     EXIT                BRANCH TO EXIT.
*
DAYMO    EQU   *
         SR    R2,R2               CLEAR REG 2.
         IC    R2,DAYFLD           INSERT LENGTH OF DAY TEXT TO REG 2.
         CLI   OPTION,X'Ø4'        IS ABBREVIATED FORMAT REQUESTED.
         BNE   DAYMO2              NO-BRANCH TO DAYMO2.
         LA    R2,3                SET UP MVC LENGTH.
         CLC   DAYFLD+1(2),THURS   IS IT THURSDAY.
         BNE   DAYMO1              NO-BRANCH TO DAYMO1.
         LA    R2,4                YES-INCREASE LENGTH COUNT.
*
DAYMO1   EQU   *
         LA    R6,DAYFLD+1         ADD START OF DATA TO DISPLACEMENT.
         AR    R6,R2               ...
         MVI   Ø(R6),C'.'          MVE IN PERIOD.
*
DAYMO2   EQU   *
         EX    R2,MOVEDAY          MVE DAY.
         LA    R4,1(R4,R2)         BUMP BUFFER POINTER.
         MVI   Ø(R4),C' '          MVE IN BLANK.
         LA    R4,1(R4)            INCREMENT REG 4 BY ONE (1).
         CLI   OPTION,X'Ø4'        IS ABBREVIATED VERSION REQUESTED.
         BE    MOVEMON1            YES-BRANCH TO MOVEMON1.
         IC    R2,MONTHFLD+1       INSERT LENGTH OF MONTH TEXT TO REG 2
         B     MOVEMON2            BRANCH TO MOVEMON2.
*
JULIAND1 EQU   *
```

```
         L     R7,JULIAN           GET DAY IN JULIAN.
         CVD   R7,DOUBWORD         CONVERT IT TO PACKED.
         MVC   Ø(7,R4),BLANKS
         UNPK  2(3,R4),DOUBWORD+6(2) MVE IT.
         OI    4(R4),C'Ø'          ENSURE GOOD SIGN.
         MVC   Ø(2,R4),UYEAR       MVE IN YEAR.
         MVI   5(R4),C' '
         CLI   OPTION,X'Ø8'        IS CENTURY W/JULIAN OPTION REQUESTED
         BE    JULIAND3            YES-BRANCH TO JULIAND3.
         CLI   OPTION,X'Ø7'        IS JULIAN W/CENTURY OPTION REQUESTED
         BNE   EXIT                NO-BRANCH TO EXIT.
         MVC   5(2,R4),UCENT       MVE IN JOB DATE CENTURY.
         CLI   SAVEPARM,C'J'       ARE WE TO CONVERT JOB DATE.
         BE    EXIT                YES-BRANCH TO EXIT.
         MVC   5(2,R4),UCENT       MVE IN SYSTEM DATE CENTURY.
         B     EXIT                BRANCH TO EXIT.
*
JULIAND3 EQU   *
         MVC   Ø(2,R4),UCENT       MVE IN JOB DATE CENTURY.
         CLI   SAVEPARM,C'J'       ARE WE TO CONVERT JOB DATE.
         BE    JULIAND5            YES-BRANCH TO JULIAND5.
         MVC   Ø(2,R4),UCENT       MVE IN SYSTEM DATE CENTURY.
*
JULIAND5 EQU   *
         MVC   2(2,R4),UYEAR       MVE IN YEAR.
         UNPK  4(3,R4),DOUBWORD+6(2) MVE IT.
         OI    6(R4),C'Ø'          ENSURE GOOD SIGN.
*
EXIT     EQU   *
         SR    RF,RF               CLEAR REG 15.
         ST    RF,SAVEAREA+16      STORE REG 15 TO SAVEAREA+16.
         L     13,SAVEAREA+4
         RETURN (14,12)
*
ONEPRM   EQU   *
         MVC   Ø(2,R4),UMONTH      MOVE DATE TO USER AREA
         MVI   2(R4),C'/'          ...
         MVC   3(2,R4),UDAY        ...
         MVI   5(R4),C'/'          ...
         MVC   6(2,R4),UYEAR       ...
         B     EXIT                RETURN
*
GREG     EQU   *
         MVC   Ø(2,R4),UMONTH      MOVE DATE TO USER AREA
         MVI   2(R4),C'/'          ...
         MVC   3(2,R4),UDAY        ...
         MVI   5(R4),C'/'          ...
         MVC   6(2,R4),UCENT       ...
         MVC   8(2,R4),UYEAR       ...
```

```
         B     EXIT                RETURN
*
DPDATES  DC    C'DPDATE STORAGE HERE. ' INSERT EYE CATCHER.
SJOBDATC DS    CL11                JOB DATE W/CENTURY. (IE MM/DD/YY/CC
SSYSCENT DC    C'19'               SYSTEM CENTURY. (IE 19, 20).
NUMPRM   DC    X'00'
OPTION   DC    X'00'
CODE     DC    C' '
TIMEP    DC    PL2'0'
TIMEU    DC    C'  '
SAVEPARM DS    CL9
UDATE    DS    0CL6
UMONTH   DS    CL2
UDAY     DS    CL2
UYEAR    DS    CL2
UCENT    DS    CL2
SUCENT   DS    CL4
TARGET   DC    CL10'          '
PATTERN  DC    X'40202120402020402020'
         DS    0F
DOUBWORD DS    0D
         DC    F'0'
FULLWORD DC    F'0'
WORK     DC    F'0'
TOD      DS    CL8                 TOD CLOCK VALUE FROM STCK.
TODMMDD  DS    CL4                 TOD MMDD.
TODYYCC  DS    CL4                 TOD YYCC.
MONTHFLD DC    CL12'            '
DAYFLD   DC    CL10'          '
         DC    C' '
         DS    0H
JULIAN   DC    F'0'
BLANKS   DC    28C' '
THURS    DC    C'TH'
SAVEAREA DC    18F'0'
         DS    0H
MOVEMO   MVC   0(1,R4),MONTHFLD+2
MOVEDAY  MVC   0(1,R4),DAYFLD+1
CLEAR2   MVC   0(13,R4),BLANKS
CLEAR3   MVC   0(18,R4),BLANKS
CLEAR4   MVC   0(19,R4),BLANKS
CLEAR5   MVC   0(28,R4),BLANKS
CLEAR6   MVC   0(05,R4),BLANKS
CLEAR7   MVC   0(05,R4),BLANKS
CLEAR8   MVC   0(05,R4),BLANKS
CLEAR9   MVC   0(10,R4),BLANKS
*
MONTHS   EQU   *
         DC    AL1(31)
```

```
         DC    X'07'
         DC    C'JANUARY   '
         DC    AL1(28)
         DC    X'08'
         DC    C'FEBRUARY  '
         DC    AL1(31)
         DC    X'05'
         DC    C'MARCH     '
         DC    AL1(30)
         DC    X'05'
         DC    C'APRIL     '
         DC    AL1(31)
         DC    X'03'
         DC    C'MAY       '
         DC    AL1(30)
         DC    X'04'
         DC    C'JUNE      '
         DC    AL1(31)
         DC    X'04'
         DC    C'JULY      '
         DC    AL1(31)
         DC    X'06'
         DC    C'AUGUST    '
         DC    AL1(30)
         DC    X'09'
         DC    C'SEPTEMBER '
         DC    AL1(31)
         DC    X'07'
         DC    C'OCTOBER   '
         DC    AL1(30)
         DC    X'08'
         DC    C'NOVEMBER  '
         DC    AL1(31)
         DC    X'08'
         DC    C'DECEMBER  '
*
DAYS     EQU   *
         DC    AL1(8)
         DC    C'SATURDAY '
         DC    AL1(6)
         DC    C'SUNDAY   '
         DC    AL1(6)
         DC    C'MONDAY   '
         DC    AL1(7)
         DC    C'TUESDAY  '
         DC    AL1(9)
         DC    C'WEDNESDAY'
         DC    AL1(8)
         DC    C'THURSDAY '
         DC    AL1(6)
```

```
         DC    C'FRIDAY   '
*
DPDATEE  DC    X'FF'
*
COMREG   MAPCOMR ,
*
         END
```

*Editor's note: See the next issue of* VSE Update *for the DPDAJ2 date adjust subroutine, which increments or decrements a passed MMDDCCYY date by a given number of days.*

*Robert Botsis*
*Senior Systems Programmer (USA)*


## REXX/VSE and VSE/ESA job control

VM/ESA users are generally pretty familiar with REXX, and Issue 23 of *VSE Update* (September 1996) included an article on how to run REXX/VSE procedures under VSE/ESA. However, if your REXX/VSE procedure is designed to interact with VSE/ESA job control, some special points need to be considered. These are discussed below.

EXAMPLE 1 (GETPART)

The GETPART procedure is shown below.

```
/*****************************************************************/
/* GETPART.PROC                                    REXX/VSE V6.1 */
/*                                                               */
/*     Assign partition-ID to symbolic job control parameter     */
/*****************************************************************/
call sysvar("SYSPID")
address JCL "// SETPARM PID="syspid
exit
```

The extract from a VSE/ESA job reproduced below shows how to use the procedure.

 

```
...
// LIBDEF PROC,SEARCH=(sublibrary containing member GETPART.PROC)
// SETPARM PID=
// EXEC REXX=GETPART,PID
/*
// IF PID = 'FA' THEN
// GOTO PIDOK
// PAUSE WRONG PARTITION
// GOTO $EOJ
/. PIDOK
...
```

If you want to develop REXX/VSE procedures to interact with VSE/ESA job control, make sure that you read the *VSE/ESA System Control Statements*, and, in particular, the topic EXEC command or statement.

In our example, job control parameter PID must be specified as part of the EXEC REXX statement, otherwise address JCL '// SETPARM PID=...' will have only a local scope within the procedure.

I would advise you to insert an end-of-data file statement (/*) after the EXEC REXX statement even if the procedure itself does not try to read data from SYSIPT. Once, when testing a REXX/VSE procedure, I omitted '/*' and later inserted a 'trace ?i' statement into the REXX/VSE procedure. The trace function then started to read all of the job control statements, until '/*' or end-of-job (/&) was encountered. Every job control statement caused the REXX/VSE procedure to issue the following message:

```
ARX0100I +++ Interactive Trace. Error 35: ...
```

Of course, the corresponding job control statements were never executed. If '/&' was encountered, the procedure and the whole job ended with the following message:

```
1N90I  EOP WAS FORCED BY EOJ
```

Job control substitutes symbolic parameters found in the PARM string of the EXEC statement. So the value of such a parameter assigned by a REXX/VSE procedure can easily be passed to other procedures or programs. This is illustrated by the following example:

```
...
// LIBDEF PROC,SEARCH=(sublibrary containing member GETPART.PROC)
// SETPARM PID=,STEP=
// EXEC REXX=GETPART,PID
/*
...
// EXEC REXX=SAMPLE2,STEP,PARM='&PID'
...
// EXEC PGM=SAMPLE1,PARM='&PID &STEP'
...
```

EXAMPLE 2 (GETPID)

This example shows the GETPID procedure.

```
/*****************************************************************/
/* GETPID.PROC                                      REXX/VSE V6.1 */
/*                                                                */
/*            Assign partition-ID and partition use               */
/*            to  symbolic  job  control parameters               */
/*****************************************************************/
call sysvar("SYSPID")
queue "// SETPARM PID="syspid",PUSE=DYN,XUSE"syspid"="
queue "// EXEC PROC=CPUVAR1,XUSE"syspid
queue "// IF XUSE"syspid "NE '' THEN"
queue "// SETPARM PUSE=&XUSE"syspid
exit
```

In this example, the job control statements are not executed by address JCL because the EXEC statement is not supported. Instead, they are stored on the data stack and executed by job control at the end of the procedure.

It is assumed that the IBM-supplied procedure CPUVAR1 in your system looks like that shown below:

```
...
SETPARM XUNATT='NO'
SETPARM XUSEBG='BØ' * PROGRAM RUNNING IN PARTITION *
SETPARM XUSEFA='NONE'
SETPARM XUSEFB='NONE'
SETPARM XUSEF1='PW'
SETPARM XUSEF2='CI'
SETPARM XUSEF3='VT'
SETPARM XUSEF4='B4'
SETPARM XUSEF5='B5'
SETPARM XUSEF6='NONE'
SETPARM XUSEF7='NONE'
```

 

```
SETPARM XUSEF8='NONE'
SETPARM XUSEF9='NONE'
```

The GETPID procedure will assign the name (pi) of the actual partition to the symbolic job control parameter PID. If the job is running in a dynamic partition, the value DYN is assigned to the PUSE job control parameter; otherwise, the value of the corresponding XUSEpi parameter from CPUVAR1 is used.

Consider the following VSE/ESA job:

```
...
// LIBDEF PROC,SEARCH=(sublibrary containing member GETPID.PROC)
// SETPARM PUSE=
// EXEC REXX=GETPID,PUSE
/*
// IF PUSE = 'VT' THEN
// GOTO PUSEOK
// PAUSE NO VTAM PARTITION
// GOTO $EOJ
/. PUSEOK
...
```

If the above job is running in partition F3, the procedure will put the following statements on the program stack:

```
// SETPARM PID=F3,PUSE=DYN,XUSEF3=
// EXEC PROC=CPUVAR1,XUSEF3
// IF XUSEF3 NE '' THEN
// SETPARM PUSE=&XUSEF3
```

At the end of the REXX/VSE procedure, job control will process the stack and assign the value F3 to PID and the value VT to PUSE. Since PID was not specified as part of the EXEC REXX statement, this parameter has only a local scope within the GETPID procedure.

IBM's CPUVAR1 procedure has the attribute DATA=YES and the job control statements on the stack including '// EXEC PROC=CPUVAR1' are processed as part of the REXX/VSE procedure GETPID. The REXX/VSE procedure GETPID must therefore also have the attribute DATA=YES, otherwise you will receive the following message:

```
1S59D   CONFLICTING DATA=YES/NO OPTIONS
```

So far, even IBM staff have been unable to give me any information about compiling procedures with the attribute DATA=YES under

VM/ESA or MVS/ESA. Since I am a native VSE/ESA user, I cannot try it myself.

EXAMPLE 3 (POFFLOAD)

```
/*****************************************************************/
/* POFFLOAD.PROC                                  REXX/VSE V6.1 */
/*                                                               */
/*     Save all VSE/POWER queue entries on a labelled tape       */
/*****************************************************************/
arg class
/*  Retrieve own VSE/POWER jobname                               */
CALL SYSVAR('SYSPOWJNM')
address POWER
/*  Retrieve entry of list queue with LISTIO output              */
'GETQE LST JOBNAME' syspowjnm 'CLASS' class 'STEM' list.
/*  Retrieve logical unit and physical device                    */
do i = 1 to list.Ø
  if word(list.i,2) = 'TEM'
  then do
    unit = word(list.i,1)
    cuu = word(list.i,3)word(list.i,4)
    leave
  end
end
/*  Delete entry of list queue with LISTIO output                */
'PDELETE LST,'syspowjnm',CCLASS='class
/*  Unassign logical unit                                        */
address JCL '// ASSGN' unit',UA'
/*  Start OFFLOAD operation                                      */
CALL SENDCMD('POFFLOAD BACKUP,ALL,'cuu',Ø8,TLBL=OFFLOAD,LTAPE=YES')
return rc
```

After installing VSE/ESA V2.2, I was very happy to read that VSE/POWER now supports the processing of labelled tapes. One of our back-up jobs in the evening uses 3480 labelled cartridges. A stack of 12 cartridges is mounted but, even in the worst case, only ten of them are used by the back-up job. So at the end of this back-up job I wanted to save all VSE/POWER queue entries on the next free cartridge without destroying the VOL1 label.

First, in the VSE/POWER partition F1 I added the statement

```
// TLBL OFFLOAD,'POWER.OFFLOAD'
```

to the partition standard subarea of the system's label information area. You can use any other valid names, but the filename and the value

 

of keyword TLBL used by the POFFLOAD command must be identical.

The back-up job that was selected to start the POFFLOAD operation looked like the follwing sample:

```
* $$ JOB JNM=BACKUP
// JOB BACKUP
// ASSGN SYS005,3480,08,VOL=volume    <-- VOL1 of first cartridge
-- Steps doing various backup tasks --
/&
* $$ EOJ
```

I inserted the following statements into that back-up job, just before end-of-job:

```
* $$ LST DISP=H,CLASS=X
// LISTIO SYS005
// LIBDEF PROC,SEARCH=(sublibrary containing member POFFLOAD.PROC)
* $$ LST DISP=H,CLASS=A
// EXEC REXX=POFFLOAD,PARM='X'
/*
```

Note that if you change the class of the first '* $$ LST' statement, you will have to change the PARM string of the EXEC REXX statement accordingly.

Now the following happens when the above steps are executed:

- By the LISTIO statement, an entry in class X of the VSE/POWER list queue is created with disposition H. This entry identifies the physical device to which the logical unit SYS005 is temporarily assigned. This information is needed because the POFFLOAD command does not support logical units like SYS005.
- The POFFLOAD procedure retrieves that entry and stores the value SYS005 in the REXX variable unit, and the physical device (for example 581) in the REXX variable cuu. Then, the entry for the list queue is deleted and the logical unit is unassigned, so that VSE/POWER can use the physical device for the POFFLOAD operation.

---

*Walter Richters*
*(Germany)*

# REXX VSE/ESA console automation

Some years ago, I contributed an Assembler program called VSECMDS to *VSE Update* (Issue 2, June 1991, updated in Issue 11, September 1993). I have now had to rewrite this program, because some of the functions in VSECMDS don't work with VSE/ESA Release 2, and seem to be impossible to implement with Assembler. Specifically, the determination of reply-ids forced me to look at alternative ways of implementing a 'batch operator' program. REXX/VSE, with the REXX/VSE console automation component, provides all of the functions and capabilities required to implement such a program.

Although REXX/VSE alone could have been used to implement all of the functions of VSECMDS, the associated overhead did not seem worth it for some of the routines I needed. These routines were written in Assembler and implemented as a REXX/VSE function package – a means of extending the REXX language itself with new functions. So, in addition to the REXX program ESACMDS, there is an Assembler program, ARXFUSER, which provides the new REXX functions ISBUSY and ISJOB. ARXFUSER is the standard name for a user function package, and is well documented in the *REXX/VSE Reference Manual*. Note that if you have never used REXX function packages you may need to assemble and catalog ARXPARMS – you should find the source in PRD1.BASE as ARXPARMS.Z. Skeleton assembly JCL is also included in this module. If you implement the two new functions ISBUSY and ISJOB using the standard user function package, you do not need to make any changes to the source of ARXPARMS.

The ESACMDS program allows most VSE console commands to be issued from a batch job. This simplifies the automation of tasks which involve both batch programs and console command activity. As well as all the normal VSE console commands, three pseudo commands are recognized by ESACMDS, namely the CPU, WAIT, and DEF commands.

## CPU COMMAND

The CPU command is used to make sure that commands are issued on

the right CPU. In a multi-CPU environment, this can be very important. The format for the command is:

```
CPU nnnnn
```

where *nnnnn* is the five-digit CPU-id of the CPU you wish to run on. The command must begin in column 1 and be followed by one or more spaces and then the five-digit CPU-id. If the specified CPU-id does not match that of the CPU running the batch job, the remaining ESACMDS control statements are bypassed and the VSE return code is set to a non-zero value.

### WAIT COMMAND

The WAIT command has two formats. In both formats, the command must begin in column 1 and be followed by one or more spaces:

- The first format is used to wait for a specified number of seconds.

  ```
  WAIT nnn
  ```

  where *nnn* is a one- to three-digit number specifying the number of seconds to wait. After the specified time expires, the next ESACMDS control statement will be processed.

- The second format is used to wait until a specified partition becomes free.

  ```
  WAIT partid [nn [CONT]]
  ```

  where partid is either a two-character real partition-id (eg F1, BG, etc) or a pseudo partition-id (see below). When the specified partition becomes free – ie it has a jobname of 'NO NAME' – the next ESACMDS control statement is processed.

  An optional timeout value *nn* may be specified. The timeout value specifies the number of seconds to wait for the partition to become free before continuing. If the timeout value is exceeded, ESACMDS terminates, setting the VSE return code to 11. This allows VSE JCL logic to test for the timeout and control the flow of the rest of the job (see Example 4 below). In addition, if *nn* is specified, the keyword CONT may optionally be specified after it, to allow ESACMDS to continue without terminating with

return code 11. Without a timeout value, ESACMDS will wait indefinitely for the specified partition to become free.

## DEF COMMAND

The DEF command makes it possible to communicate with programs in dynamic partitions. The problem with dynamic partitions is that several partitions may belong to a dynamic class. For instance, dynamic class C may have two partitions defined for it, and might even be used to run two different CICS systems.

As an example, let's call one of these PROD and the other TEST – that is, the POWER job names are PROD and TEST, and they both have CLASS=C specified. The first job to be released, either PROD or TEST, will start running in dynamic partition C1, and the second job to be released will run in dynamic partition C2. The operators may always release PROD first and TEST second, so that you can always communicate with PROD by doing 'MSG C1' and with TEST by doing a 'MSG C2'. This is no problem – ESACMDS can do this if you want. But what happens if TEST gets released first and starts running in dynamic partition C1, and PROD starts running in dynamic partition C2, and you have jobs which use ESACMDS commands which were based on the premise that PROD always ran in dynamic partition C1 and TEST always ran in dynamic partition C2? This could be a major problem.

This is where the pseudo command DEF comes in handy. DEF is used to define a pseudo partition-id which will be associated with a certain job name regardless of which partition the job is running in.

```
DEF ##=jobname
```

The command must begin in column 1 and be followed by one or more spaces and then the parameter defining the pseudo partition-id. '*##*' is the two-character pseudo partition-id being defined. The first character of the pseudo partition-id must be a '#'. The second character can be any other character, as long as you have not already used that character for some other pseudo partition-id. Examples of valid pseudo partition-ids are #1, #2, #3, #P, #X, etc. The pseudo partition-id must be immediately followed by an equals sign(=), which, in turn, must be

followed by the ‘jobname’. ‘jobname’ is the POWER job name of the job with which you want the pseudo partition-id to be associated. For instance, using the example of the CICS jobs PROD and TEST, we might code the following ESACMDS:

```
DEF #1=PROD
DEF #2=TEST
```

Now, instead of referencing a real partition-id on an ESACMDS statement, we can instead use the pseudo partition-id. For example, assuming the above DEF commands, issuing an ‘MSG #1’ would issue an MSG to the partition in which PROD is running.

Jobnames on the DEF statement can also be coded in a generic form by appending an asterisk suffix. For instance:

```
DEF #1=PROD*
```

would match any job name starting with the four characters PROD. This can be useful in situations where the job name may vary by some suffix. For example, at the installation where ESACMDS was developed, each CICS has a five-character job name. The first four characters identify the CICS system, and the fifth character is a numeric digit used as a modification level. Each time the CICS start-up job stream is modified, the numeric fifth character is incremented by 1. This new job is then released the next time that CICS is cycled. In this environment, the job name of a specific CICS system is always subject to change. The generic job name capability is used to associate a pseudo partition-id with the correct job.

Note: the routines that locate the job name to associate with a partition-id work by scanning the active partitions and stopping at the first match. You should be aware of this, especially when using generic job names, since you might get a job name match for a partition when you are not expecting it. DEF statements can be defined for either static or dynamic partitions.

### Commands preceded by a reply-id

There is also a slight modification to the normal syntax of commands which are preceded by a reply-id. Normally, if you were to issue a command at the console, say ‘MSG F2’, you would be prompted with

the reply-id which you should use for your response. This reply-id cannot always be predicted, and may vary depending on the tasks running in the partition. To handle this situation and allow ESACMDS to issue MSG commands and properly handle responses to it, the reply-id is not actually coded on the ESACMDS statement. Instead, the two-character partition-id is substituted for it and the actual reply-id will be determined by ESACMDS as the job is running.

Also, when coding a statement which requires the reply-id for a partition, a pseudo partition-id is valid if one has been defined for the partition via a DEF statement. The reply-id must be followed by a hyphen (-). ESACMDS identifies replies by looking for a hyphen in the third, fourth, or fifth position of the control statement. This was not required by the older program VSECMDS, but, to maintain some compatibility, if a control statement has a space in the third position it will also be considered as a reply.

**Other points to note**

Note that issuing an 'MSG' to an inactive partition or to a partition that is not set up for operator communication will cause ESACMDS to bypass the remaining control statements and exit with a non-zero VSE return code.

The 'F' and 'PAUSE' VSE commands can also reference a pseudo partition-id. The pseudo partition-id will be replaced by a real partition-id if a match can be made. The 'F' command may also contain the 'HOLD' parameter with the pseudo partition-id. The 'PAUSE' command may also contain the 'EOJ' parameter with the pseudo partition-id.

ESACMDS control statements begining with an asterisk in column 1 are considered to be comments.

The following examples may help to illustrate the various commands and formats.

EXAMPLE 1

```
// EXEC REXX=ESACMDS
DEF #1=TEST1                        associate pseudo id #1 with job TEST1
```

```
MSG #1                              MSG to the partition
#1-CEMT S DAT(FILEA) CLO            close a file
#1                                  EOB
/*
```

## EXAMPLE 2

```
// EXEC REXX=ESACMDS
DEF #A=ADABAS                       associate pseudo id #A with job ADABAS
MSG #A                              MSG to the partition
#A-ADAEND                           shutdown
/*
```

## EXAMPLE 3

```
// EXEC REXX=ESACMDS
DEF #J=JOBTRACK                     associate pseudo id #J with job
JOBTRACK
MSG #J                              MSG to the partition
#J-STOP                             send command to partition
WAIT #J                             wait for partition to become free
R RDR,JOBTRACK                      release the job
WAIT 60                             wait 60 secs
* we need to redefine pseudo #J
* since there is no guarantee
* that it is still valid after
* cycling JOBTRACK
DEF #J=JOBTRACK                     redefine pseudo id #J
MSG #J                              MSG to the partition
#J-STATUS                           send command to partition
#J                                  EOB
/*
```

## EXAMPLE 4

```
*****************************************************************
*  WAIT #1 20 - WAITS FOR THE PARTITION WITH ADAMPM* TO BECOME  *
*  FREE. IF IT DOES NOT BECOME FREE AFTER 20 SECONDS THE JOB    *
*  WILL CONTINUE WITH $RC = 11 TO INDICATE A TIMEOUT            *
*****************************************************************
// EXEC REXX=ESACMDS
DEF #1=ADAMPM*
MSG #1
#1-DSTAT
#1
WAIT #1 20                 <==== 20 IS THE TIMEOUT VALUE
/*
```

```
// IF $RC NE 11 THEN       <==== RETURN CODE = 11 IF TIMED OUT
// GOTO SKP
// EXEC REXX=ESACMDS       <==== THIS STEP WILL ONLY BE
DEF #1=ADAMPM*                    EXECUTED IF THE PREVIOUS STEP
MSG #1                            TIMES OUT AFTER 20 SECONDS
#1-DRES
#1
/*
/. SKP
```

As we saw above, if you issue an 'MSG' to an inactive partition, ESACMDS would bypass the remaining statements, possibly discarding statements which you really would like to go ahead and issue. It is sometimes useful not to issue all commands via the same execution of ESACMDS (see Examples 5 and 6)

EXAMPLE 5

```
// EXEC REXX=ESACMDS
MSG F2
F2-CJSI STOP,OFF
F2
MSG F6
F6-CJSI STOP,OFF
F6
/*
```

This would fail to issue commands in F6 if F2 was not active. Instead, you might do the following so that F6 commands are issued regardless of whether F2 is active or not.

EXAMPLE 6

```
// EXEC REXX=ESACMDS
MSG F2
F2-CJSI STOP,OFF
F2
/*
// EXEC REXX=ESACMDS
MSG F6
F6-CJSI STOP,OFF
F6
/*
```

## RUNNING THE PROGRAM

Our current environment is VSE/ESA 2.1.2 with CICS/VSE 2.3. This version of ESACMDS has been running in this environment for more than a year, with no significant problems. This does not mean that it can be expected to be totally bug free. We use a very small subset of the possible commands and functions of which the program is capable.

If you need further information on REXX, function packages, or console automation, the following manuals should prove helpful:

- SC33-6598 *REXX/VSE Console Automation.*
- SC33-6642 *VSE/REXX Reference.*
- SC33-6641 *REXX/VSE User's Guide.*

## REXX BATCH CONSOLE AUTOMATION

```
// EXEC LIBR
ACC S=IJSYSRS.SYSLIB
CATALOG ESACMDS.PROC R=Y DATA=YES
/*****************************************************************/
/*  REXX BATCH CONSOLE AUTOMATION                                */
/*****************************************************************/
/* TRACE R */
 SIGNAL ON HALT NAME CLEANUP
 SIGNAL ON SYNTAX NAME SYNTAX
 RC=OPERMSG(ON)                              /* COMM EXIT ACTIVE  */
 "EXECIO * DISKR SYSIPT (STEM IPT."
 LASTPID ='ØØ'
 LASTRID ='ØØ'
 MAXWAIT = 999
 DEF. = ''
 DEF.Ø = Ø
 OLDIN =ASSGN('STDIN','SYSLOG')
 OLDOUT=ASSGN('STDOUT','SYSLST')
 CALL SYSVAR(SYSPID)
 RC=LOCKMGR('LOCK',ESACMD)                   /* SERIALIZE ESACMD  */
 IF RC/=Ø THEN                               /* LOCK FAILED       */
    DO
     SAY 'ESACMD14-LOCK FAILED'
     RC=ASSGN('STDOUT','SYSLOG')
     SAY 'ESACMD14-LOCK FAILED'
     EXIT 13
    END
```

```
 ADDRESS CONSOLE
 ACTIVATE 'NAME ESA'SYSPID ' PROFILE REXALLRC'
 IF RC/=Ø THEN                                  /* ACTIVATE FAILED   */
    DO
     RC=LOCKMGR('UNLOCK',ESACMD)                /* REMOVE THE LOCK   */
     SAY 'ESACMD13-CONSOLE ACTIVATE FAILED'
     RC=ASSGN('STDOUT','SYSLOG')
     SAY 'ESACMD13-CONSOLE ACTIVATE FAILED'
     EXIT 13
    END
 DO N = 1 TO IPT.Ø                              /* PROCESS COMMANDS  */
  CMD = IPT.N
  SAY CMD                                       /* ECHO COMMAND      */
  SELECT
    WHEN SUBSTR(CMD,1,1) = '*'      THEN CALL COMMENT
    WHEN SUBSTR(CMD,1,5) = 'WAIT '  THEN CALL WAIT
    WHEN SUBSTR(CMD,1,6) = 'CPUID ' THEN CALL CPUID
    WHEN SUBSTR(CMD,1,4) = 'DEF '   THEN CALL DEF
    WHEN SUBSTR(CMD,1,4) = 'MSG '   THEN CALL MSG
    WHEN SUBSTR(CMD,1,2) = 'F '     THEN CALL FLUSH
    WHEN SUBSTR(CMD,1,6) = 'PAUSE ' THEN CALL PAUSE
    WHEN POS('-',CMD)    = '3'      THEN CALL REPLY
    WHEN POS(' ',CMD)    = '3'      THEN CALL REP
    WHEN POS('-',CMD)    = '4'      THEN CALL REPLY
    WHEN POS('-',CMD)    = '5'      THEN CALL REPLY
    OTHERWISE CALL CMDS
  END
 END
/*********************************************************************/
/* NORMAL EOJ                                                        */
/*********************************************************************/
EXIT:
 RC=OPERMSG(OFF)                        /* REMOVE COMM EXIT         */
 DEACTIVATE 'ESA'SYSPID                 /* DEACTIVATE OUR CONSOLE   */
 RC=LOCKMGR('UNLOCK',ESACMD)            /* RELEASE THE LOCK         */
 EXIT Ø
/*********************************************************************/
/* EOJ WITH ERRORS                                                   */
/*********************************************************************/
ERROR:
 RC=OPERMSG(OFF)                        /* REMOVE COMM EXIT         */
 DEACTIVATE 'ESA'SYSPID                 /* DEACTIVATE OUR CONSOLE   */
 RC=LOCKMGR('UNLOCK',ESACMD)            /* RELEASE LOCK             */
 RC=ASSGN('STDOUT','SYSLST')
 SAY 'ESACMDØ1-ERROR CAUSED REMAINING COMMANDS TO BE BYPASSED'
 RC=ASSGN('STDOUT','SYSLOG')
 SAY 'ESACMDØ1-ERROR CAUSED REMAINING COMMANDS TO BE BYPASSED'
 EXIT 1
/*********************************************************************/
/* HALT TRAP                                                         */
/*********************************************************************/
```

```
CLEANUP:
 RC=OPERMSG(OFF)                        /* REMOVE COMM EXIT         */
 DEACTIVATE 'ESA'SYSPID                 /* DEACTIVATE OUR CONSOLE   */
 RC=LOCKMGR('UNLOCK',ESACMD)            /* RELEASE LOCK             */
 EXIT 2
/*********************************************************************/
/* SYNTAX TRAPS                                                      */
/*********************************************************************/
SYNTAX:
 SAY 'ESACMD99-REXX ERROR ' RC 'ON LINE ' SIGL ':' ERRORTEXT(RC)
 SAY 'ESACMD99-SOURCELINE ' SOURCELINE(SIGL)
 RC=OPERMSG(OFF)                        /* REMOVE COMM EXIT         */
 DEACTIVATE 'ESA'SYSPID                 /* DEACTIVATE OUR CONSOLE   */
 RC=LOCKMGR('UNLOCK',ESACMD)            /* RELEASE LOCK             */
 EXIT 3
/*********************************************************************/
/*  NO RESPONSE                                                      */
/*********************************************************************/
NORESP:
 RC=ASSGN('STDOUT','SYSLOG')
 SAY 'ESACMD1Ø-NO RESPONSE'
 RC=ASSGN('STDOUT','SYSLST')
 SAY 'ESACMD1Ø-NO RESPONSE'
 SIGNAL ERROR
/*********************************************************************/
/* COME HERE TO PROCESS COMMENTS                                     */
/*********************************************************************/
COMMENT:
 RETURN Ø
/*********************************************************************/
/* COME HERE TO PROCESS MSG COMMANDS. FORMAT IS EITHER MSG 'PARTID'*/
/* OR MSG 'PSEUDO PARTID'.                                           */
/*********************************************************************/
MSG:
 IF LENGTH(CMD) < 5 THEN
    SIGNAL ERROR
 CMD = SUBSTR(CMD,4)                    /* REMOVE 'MSG'             */
 CMD = STRIP(CMD,B)                     /* STRIP SPACES             */
 IF LENGTH(CMD) < 2 THEN                /* MUST BE 2 CHAR PARTID    */
    SIGNAL ERROR
 IF SUBSTR(CMD,1,1) = '#' THEN          /* IS IT A PSEUDO?          */
    CMD = GETPID(CMD)                   /* GET REAL PART ID         */
 'MSG 'CMD                              /* MSG THE PARTITION        */
 FC = GETMSG(MSG.,'RESP',,,3Ø)          /*                          */
 RC = WAIT4(CMD)                        /* WAIT FOR RESPONSE        */
 IF RC = Ø THEN                         /* GOT A RESPONSE?          */
    RETURN Ø
 SIGNAL NORESP                          /* DID NOT GET A RESPONSE   */
/*********************************************************************/
/* COME HERE TO PROCESS FLUSH COMMANDS. THEY MAY HAVE A PSEUDO ID  */
/*********************************************************************/
```

```
FLUSH:
 IF LENGTH(CMD) < 3 THEN                /*                          */
    SIGNAL ERROR
 CMD = SUBSTR(CMD,2)                    /* REMOVE 'F '              */
 PARSE VAR CMD PARTID ',' HOLD          /* STRIP OFF PARTID         */
 HOLD = STRIP(HOLD,B)                   /* STRIP SPACES             */
 PARTID = STRIP(PARTID,B)               /* STRIP SPACES             */
 IF LENGTH(PARTID) /= 2 THEN            /* MUST BE 2 CHARS          */
    SIGNAL ERROR
 IF SUBSTR(PARTID,1,1) = '#' THEN       /* IS IT A PSEUDO?          */
    PARTID = GETPID(PARTID)             /* GET REAL PART ID         */
 CMD = 'F 'PARTID                       /* FLUSH THE PARTITION      */
 IF HOLD /= '' THEN
   CMD = CMD','HOLD
 CMD                                    /* DO IT                    */
 FC = GETMSG(MSG.,'RESP',,,3Ø)          /* GET ANY RESPONSES        */
 RETURN Ø
/*********************************************************************/
/* COME HERE TO PROCESS PAUSE COMMANDS. THEY MAY HAVE A PSEUDO ID  */
/*********************************************************************/
PAUSE:
 IF LENGTH(CMD) < 7 THEN
    SIGNAL ERROR
 CMD = SUBSTR(CMD,6)                    /* REMOVE 'PAUSE '          */
 PARSE VAR CMD PARTID ',' EOJ           /* STRIP OFF PARTID         */
 EOJ = STRIP(EOJ,B)                     /* STRIP SPACES             */
 PARTID = STRIP(PARTID,B)               /* STRIP SPACES             */
 IF LENGTH(PARTID) /= 2 THEN
    SIGNAL ERROR
 IF SUBSTR(PARTID,1,1) = '#' THEN       /* IS IT A PSEUDO?          */
    PARTID = GETPID(PARTID)             /* GET REAL PART ID         */
 CMD = 'PAUSE 'PARTID                   /* FLUSH THE PARTITION      */
 IF EOJ /= '' THEN
   CMD = CMD','EOJ
 CMD                                    /* DO IT                    */
 FC = GETMSG(MSG.,'RESP',,,3Ø)          /* GET ANY RESPONSES        */
 RETURN Ø
/*********************************************************************/
/* FUNCTION WHICH WAITS FOR A REPLID TO APPEAR ON THE CONSOLE      */
/*********************************************************************/
WAIT4: PROCEDURE EXPOSE SIGL
 PID=ARG(1)
 FC=SLEEP(2)
 CNT = Ø
 DO WHILE GETRID(PID) = '' & CNT <= 3 /* WAIT FOR MAX OF 3Ø SECS   */
   FC=SLEEP(1Ø)
   CNT = CNT + 1
 END
 IF CNT > 3 THEN                      /* NO RESPONSE              */
    RETURN 1
 RETURN Ø
```

```
/*********************************************************************/
/* EXTRACTS CPUID FROM 1ST LINE OF SIR MESSAGES                      */
/*   THIS ROUTINE ISSUES THE VSE SIR COMMAND TO DETERMINE THE        */
/*   CPUID. THE SIR COMMAND OUTPUTS SEVERAL LINES OF INFO TO THE     */
/*   CONSOLE WHICH MAY NOT BE DESIRABLE. AN ADDITIONAL FUNCTION      */
/*   COULD BE ADDED TO ARXFUSER THAT EXTRACTS THE CPUID WITH         */
/*   A SYSTEM MACRO AND RETURNS THE CPUID.                           */
/*********************************************************************/
CPUID:
 IF LENGTH(CMD) < 7 THEN
    SIGNAL ERROR
 CMD=SUBSTR(CMD,6)                      /* STRIP OFF 'CPUID'         */
 CMD=STRIP(CMD,B)                       /* STRIP OFF SPACES          */
 IF LENGTH(CMD) /= 5 THEN               /* MUST HAVE 5 CHARS         */
    SIGNAL ERROR
 "SIR"                                  /* ISSUE SIR COMMAND         */
 FC = GETMSG(MSG.,'RESP',,,3Ø)          /* GET RESPONSES             */
 IF FC = Ø THEN
    DO
      LIT = 'CPUID     = '             /* LOOK FOR CPUID            */
      X = POS(LIT,MSG.1)
      X = X + LENGTH(LIT) + 3           /* POSITION ON CPUID         */
      CPUID = SUBSTR(MSG.1,X,5)         /* EXTRACT CPUID             */
      IF CPUID /= CMD THEN              /* DOES IT MATCH?            */
         DO
          SAY 'ESACMDØ2-CPUID 'CPUID' DOES NOT MATCH 'SUBSTR(CMD,7,5)
          SIGNAL ERROR
         END
    END
 RETURN
/*********************************************************************/
/* USE STEM VARIABLE DEF. TO MAINTAIN A LIST OF PSEUDO PARTITION     */
/* IDS. PSEUDO ID MUST BE 2 CHARACTERS, A # PLUS A NON BLANK         */
/* CHARACTER. PSEUDO ID MUST BE IMMEDIATELY FOLLOWED BY =            */
/*********************************************************************/
DEF:
 IF LENGTH(CMD) < 5 THEN
    SIGNAL ERROR
 CMD = SUBSTR(CMD,4)                    /* STRIP OFF 'DEF'           */
 CMD = STRIP(CMD,B)                     /* STRIP OFF SPACES          */
 IF LENGTH(CMD) < 5 THEN                /* NEED AT LEAST 4 CHARS     */
    SIGNAL ERROR
 IF SUBSTR(CMD,1,1) /= '#' THEN         /* 1ST CHAR MUST BE A #      */
    SIGNAL ERROR
 IF SUBSTR(CMD,2,1)  = ' ' THEN         /* 2ND CHAR CANNOT BE BLANK */
    SIGNAL ERROR
 IF SUBSTR(CMD,3,1) /= '=' THEN         /* MUST BE FOLLOWED BY =     */
    SIGNAL ERROR
 I = POS('=',CMD)                       /* LOCATE =                  */
 IF I /= 3 THEN                         /* MUST BE 3RD CHAR          */
```

```
   SIGNAL ERROR
 I = I + 1                              /* POINT PAST =             */
 JOBN = SUBSTR(CMD,I)                   /* GET JOBNAME              */
 JOBN = STRIP(JOBN,B)                   /* STRIP ANY SPACES         */
 JOBN = LEFT(JOBN,8)                    /* ONLY 8 CHARS             */
 PID  = ISJOB(JOBN)                     /* GET PART ID              */
 PSEUDO = SUBSTR(CMD,1,3)
 CMD  = SUBSTR(CMD,1,3)PID              /* PSEUDO ID PLUS REAL ID   */
 DO  I = Ø TO DEF.Ø                     /* SCAN TABLE               */
   IF DEF.I = PSEUDO THEN
      DO
       DEF.I = CMD                      /* EXISTS, SO REPLACE IT    */
       RETURN
      END
 END
 DEF.I = CMD                            /* ADD TO TABLE             */
 DEF.Ø = I                              /* SAVE TABLE SIZE          */
 RETURN
/*********************************************************************/
/*  ISSUE CONSOLE COMMAND                                            */
/*********************************************************************/
CMDS:
 CMD                                     /* PERFORM THE COMMAND     */
 FC = GETMSG(MSG.,'RESP',,,3Ø)           /* GET ANY RESPONSES       */
 RETURN
/*********************************************************************/
/* CHANGE BLANK SEPARATOR TO '-'                                     */
/*********************************************************************/
REP:
 HOLD = CMD                              /* SAVE CMD                */
 PARSE VAR HOLD R1 ' ' HOLD              /* SEPARATE AT 1ST SPACE   */
 CMD = R1'-'HOLD                         /* PUT IT BACK WITH A -    */
 CALL REPLY                              /* NOW PROCESS IT          */
 RETURN
/*********************************************************************/
/* COME HERE TO ISSUE REPLIES                                        */
/*********************************************************************/
REPLY:
 PARSE VAR CMD PARTID '-' CMD           /* STRIP OFF PARTID         */
 CMD = STRIP(CMD,B)                     /* STRIP OFF SPACES         */
 IF SUBSTR(PARTID,1,1) = '#' THEN       /* IS IT A PSEUDO           */
    DO
      PARTID=GETPID(PARTID)             /* CONVERT IT TO A REAL ID  */
    END
 CMD = PARTID' 'CMD                     /* PUT IT BACK TOGETHER     */
 LEN = LENGTH(CMD)-3
 IF LEN < 1 THEN
    REPLY = ' '
 ELSE
    REPLY = SUBSTR(CMD,4,LEN)
```

```
 REPLYID=GETRID(PARTID)
 IF LENGTH(REPLYID) > Ø & REPLY = ' ' THEN      /* EOB             */
    DO
     REPLYID
     RETURN
    END
 IF LENGTH(REPLYID) > Ø THEN
    DO
     REPLYID REPLY
     FC = GETMSG(MSG.,'RESP',,,15)
     IF FC = 5 THEN DO
        CALL SYSVAR 'SYSERRCODES'
        SAY SYSERRCODES
        FC = GETMSG(MSG.,'MSG',,,1Ø)
        END
    END
 ELSE
    DO
     SAY CMD
     SAY 'ESACMDØ3-INVALID OR INACTIVE PARTITION. COMMAND IGNORED'
    END
 RETURN
/*********************************************************************/
/* WAIT FORMAT IS 'WAIT NNN' OR 'WAIT PARTID', WHERE NNN IS A TIME */
/* INTERVAL AND PARTID IS A REAL OR PSEUDO PARTITION ID.           */
/* ADDITIONALLY A TIMEOUT INTERVAL MAY BE SPECIFIED WHEN WAITING   */
/* ON A PARTITION 'WAIT PARTID NNN'.                               */
/*********************************************************************/
WAIT:
 IF LENGTH(CMD) < 7 THEN
    SIGNAL ERROR
 PARTID = SUBSTR(CMD,6)
 PARTID = STRIP(PARTID,B)
 WAIT=PARTID
 IF LENGTH(PARTID) = Ø THEN
    SIGNAL ERROR
 SELECT
   WHEN DATATYPE(PARTID) = 'NUM' THEN
      IF WAIT > MAXWAIT THEN
         DO
          SAY 'ESACMDØ4-WAIT INTERVAL INVALID.  COMMAND IGNORED'
          SAY 'ESACMDØ5-CURRENT MAXIMUM WAIT INTERVAL IS ' MAXWAIT'.'
          SIGNAL ERROR
         END
      ELSE
         DO
          SAY 'ESACMDØ6- WAITING FOR ' WAIT 'SECONDS...'
          RC = SLEEP(WAIT)
          SAY 'ESACMDØ7- PROCESSING CONTINUES...'
          RETURN
```

```
        END
   WHEN DATATYPE(PARTID) = 'CHAR' THEN
      DO
         TIMEOUT = Ø
         CONTINUE = ''
         HOLD = PARTID
         PARSE VAR HOLD PARTID ' ' HOLD
         PARTID = STRIP(PARTID,B)
         TIMEOUT = STRIP(HOLD,B)
         IF POS('CONT',TIMEOUT) /= Ø THEN
            DO
              PARSE VAR TIMEOUT TIMEOUT ' ' CONTINUE
              TIMEOUT = STRIP(TIMEOUT,B)
              CONTINUE = STRIP(CONTINUE,B)
            END
         IF SUBSTR(PARTID,1,1) = '#' THEN
            DO
              PARTID=GETPID(PARTID)
            END
         SAY 'ESACMD12-WAITING FOR 'PARTID' TO BECOME FREE'
         YN = ISBUSY(PARTID)
         T  = Ø
         DO WHILE YN = '1'
            RC = SLEEP(1)                  /* WAIT 1 SECOND       */
            YN = ISBUSY(PARTID)            /* IS PARTITION BUSY?  */
            T = T + 1
            IF T > TIMEOUT THEN
               DO
                SAY 'ESACMD15-TIMED OUT WAITING ON 'PARTID
                YN = 'Ø'                   /* TERMINATE DO-WHILE  */
                IF CONTINUE /= 'CONT' THEN  /* CONTINUE ANYWAY?    */
                   DO
                    RC=OPERMSG(OFF)        /* REMOVE COMM EXIT    */
                    DEACTIVATE 'ESA'SYSPID  /* DEACTIVATE CONSOLE  */
                    RC=LOCKMGR('UNLOCK',ESACMD)      /* REMOVE LOCK*/
                    EXIT 11                /* EXIT WITH TIMEOUT RC*/
                   END
               END
         END  /* DO WHILE */
        RETURN
      END /* DO */
   OTHERWISE
      DO
       SIGNAL ERROR
      END
 END
 RETURN
/*****************************************************************/
/* FUNCTION WHICH GIVEN A PARTITION ID WILL RETURN EITHER THE     */
/* REPLY ID FOR THAT PARTITION OR A NULL STRING IF THERE IS NO    */
```

```
/* OUTSTANDING REPLY OR IF THE PARTITION IS INACTIVE.             */
/*****************************************************************/
GETRID: PROCEDURE EXPOSE SIGL LASTRID LASTPID
 PARTID = ARG(1)                        /* GET PARTITION ID       */
 'REPLID'                               /* REQUEST LIST OF REPLIDS */
 FC = GETMSG(MSG.,'RESP',,,3Ø)          /* RETRIEVE LIST          */
 IF FC = Ø THEN
    DO N = 1 TO MSG.Ø                   /* PROCESS LINE BY LINE   */
     X = POS(PARTID'-',MSG.N,1)         /* LOOK FOR PARTITION ID  */
     IF X > Ø THEN                      /* FOUND A MATCH          */
        DO                              /* EXTRACT REPLID         */
          REPLYID = SUBSTR(MSG.N,X+3,4)
          RETURN REPLYID                /* AND RETURN IT          */
        END
    END
 RETURN ''                              /* NOT FOUND, RETURN NULL */
/*****************************************************************/
/* FUNCTION WHICH GIVEN A PSEUDO PARTITION ID WILL RETURN THE REAL*/
/* PARTITION ID. IF THE PSEUDO ID CANNOT BE FOUND IN THE STEM     */
/* VARIABLE DEF., THE PSEUDO ID IS RETURNED.                      */
/*****************************************************************/
GETPID: PROCEDURE EXPOSE SIGL DEF.
 PSEUDO = ARG(1)
 DO I = 1 TO DEF.Ø
   IF SUBSTR(PSEUDO,1,2) = SUBSTR(DEF.I,1,2) THEN
      DO
       X = POS('=',DEF.I)
       X = X + 1
       PSEUDO = STRIP(SUBSTR(DEF.I,X),B)
       RETURN PSEUDO
      END
 END
 RETURN PSEUDO
/+
/*
/&


* ARXFUSER Function Package
// LIBDEF *,CATALOG=USRE.ESA21
// LIBDEF *,SEARCH=(PRD2.PROD,PRD1.BASE,PRD2.GEN1,PRD1.MACLIB)
// OPTION CATAL,XREF,NODECK
 PHASE ARXFUSER,*
// EXEC ASMA9Ø,SIZE=1M
         PRINT NOGEN
*************************************************
* REXX USER FUNCTION PACKAGE                    *
*  ADDS 2 new functions to REXX:                *
*      ISBUSY - DETERMINES IF A PARTITION IS BUSY *
*      ISJOB  - DETERMINES IN WHICH PARTITION     *
```

```
*               IF ANY A SPECIFIC JOB IS          *
*               RUNNING                           *
*************************************************
*        EQUREGS
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         MAPPCB
COMREG   MAPCOMR
SYSCOM   SYSCOM
         ARXEFPL
         ARXEVALB
ARXFUSER CSECT
ARXFUSER AMODE 31
ARXFUSER RMODE ANY
         DC    CL8'ARXFPACK'  FUNCTION PACKAGE DIRECTORY NAME
         DC    FL4'24'        LENGTH OF DIRECTORY HEADER
         DC    FL4'2'         NUMBER OF FUNCTIONS IN PACKAGE
         DC    FL4'Ø'         RESERVED
         DC    FL4'32'        LENGTH OF DIRECTORY ENTRY
*
         DC    CL8'ISBUSY   ' FUNCTION NAME
         DC    VL4(ISBUSY)    ADDR OF FUNCTION
         DC    FL4'Ø'         RESERVED
         DC    CL8' '         ENTRY POINT-USES FUNCTION ADDR IF BLANK
         DC    CL8' '         RESERVED
*
         DC    CL8'ISJOB   '  FUNCTION NAME
         DC    VL4(ISJOB)     ADDR OF FUNCTION
         DC    FL4'Ø'         RESERVED
         DC    CL8' '         ENTRY POINT-USES FUNCTION ADDR IF BLANK
         DC    CL8' '         RESERVED
*********************************************************************
* FUNCTION WHICH GIVEN A PARTITION ID WILL RETURN 1 IF THE          *
* PARTITION HAS A JOB RUNNING IN IT OR Ø IF NO JOB IS RUNNING.      *
* PARTITION ID CAN BE FOR EITHER A STATIC OR A DYNAMIC PARTITION.   *
*********************************************************************
```

```
ISBUSY   CSECT
ISBUSY   AMODE 31
ISBUSY   RMODE ANY
         USING ISBUSY,R15
         STM   R14,R12,12(R13)
         LR    R12,R15                 COPY ENTRY ADDRESS
         DROP  R15
         USING ISBUSY,R12              AND USE AS BASE
         LR    R2,R13
         LA    R13,SAVEREGS
         ST    R13,8(R2)
         ST    R2,4(R13)
         B     BEGIN
         DC    CL8'ISBUSY '
SAVEREGS DS    18F
BEGIN    EQU   *
         SR    R15,R15                    CLEAR REG
         LA    R15,1                      RC = 1 (ERRORS)
         LTR   R1,R1                      WERE ANY FIELDS PASSED?
         BZ    RETURN                     ERROR, RETURN
         USING EFPL,R1
         L     R8,EFPLEVAL                POINTER TO ADDR OF
                                           EVALBLOCK
         L     R8,Ø(R8)                   ADDR OF EVALBLOCK
         USING EVALBLOCK,R8
         L     R3,EFPLARG                 REXX ARGS
         L     R4,Ø(R3)                   GET ADDR OF 1ST ARG
         L     R5,ANCHOR                  APCBATAB (1ST PCB IS AR)
LOOPP    EQU   *
*        MODESET KEY=ZERO                 MAY OR MAY NOT NEED PKØ
         LA    R5,4(,R5)                  BUMP THE PCB POINTER
         CLI   Ø(R5),X'FF'                END OF PCB TABLE?
         BE    RETØ
         L     R6,Ø(R5)                   LOAD A PCB
         LTR   R6,R6                      EMPTY SLOT
         BZ    LOOPP
         L     R6,Ø(R5)                   LOAD A PCB
         USING PCBADR,R6
         L     R2,PCECOMRA                GET COMREG
         USING COMREG,R2
         MVC   PARTID,PCELID              SAVE PARTID
         MVC   FLAG,PCEFLAG               SAVE FLAG
         MODESET KEY=USER
         CLC   PARTID(2),Ø(R4)            IS IT THE RIGHT PARTID?
         BE    FOUND
         B     LOOPP
*********************************************************************
*  FOUND AN ACTIVE PARTITION. IF IT IS A STATIC PARTITION WE NEED   *
*  TO CHECK FURTHER TO SEE IF A PGM IS RUNNING IN IT. IF IT IS A    *
```

```
*  DYNAMIC PARTITION WE CAN RETURN SINCE IT WOULD NOT HAVE A PCB IF *
*  IT WAS INACTIVE.                                                 *
********************************************************************
FOUND    EQU   *
         SR    R15,R15                  CLEAR RC
         TM    FLAG,X'4Ø'               TEST FOR DYNAMIC PARTITION
         BO    RETØ                     MUST BE DYNAMIC
         CLC   COMNAME(8),=CL8'NO NAME '  INACTIVE STATIC PARTITION?
         BE    RET1
RETØ     EQU   *
         SR    R15,R15                  CLEAR REG
         LA    R15,1                    RETURN FIELD LENGTH
         ST    R15,EVALBLOCK_EVLEN      SAVE IT FOR REXX
         MVC   EVALBLOCK_EVDATA(1),=C'1'  1 = ISBUSY
         B     RETURN
RET1     EQU   *
         SR    R15,R15                  CLEAR REG
         LA    R15,1                    RETURN FIELD LENGTH
         ST    R15,EVALBLOCK_EVLEN      SAVE IT FOR REXX
         MVC   EVALBLOCK_EVDATA(1),=C'Ø'  Ø = INACTIVE
RETURN   EQU   *
         SR    R15,R15                  CLEAR REG
         L     R13,4(R13)               RETURN TO CALLER
         LM    R2,R12,28(R13)
         L     R14,12(R13)
         MVI   12(R13),X'FF'
         BR    R14                      R15 CONTAINS RC FOR REXX
PARTID   DC    CL2' '
FLAG     DC    X'ØØ'
PHOLD    DC    CL2' '
PADDR    DC    F'Ø'
         LTORG
*        END   ISBUSY
********************************************************************
* FUNCTION WHICH GIVEN A JOBNAME WILL RETURN THE PARTITION ID      *
* THAT THE JOB IS RUNNING IN. IF THE JOBNAME IS NOT FOUND A        *
* BLANK PARTID WILL BE RETURNED. EXTERNAL PGM ISJOB RETURNS        *
* RC=1 IF FOUND AND RC=2 IF NOT FOUND.                             *
********************************************************************
ISJOB    CSECT
ISJOB    AMODE 31
ISJOB    RMODE ANY
         USING ISJOB,R15
         STM   R14,R12,12(R13)
         LR    R12,R15                  COPY ENTRY ADDRESS
         DROP  R15
         USING ISJOB,R12                AND USE AS BASE
         LR    R2,R13
         LA    R13,SAVEREG
         ST    R13,8(R2)
```

```
         ST    R2,4(R13)
         B     BEGIN1
         DC    CL8'ISJOB'
SAVEREG  DS    18F
BEGIN1   EQU   *
         SR    R15,R15                  CLEAR REG
         LA    R15,1                    RC = 1 (ERRORS)
         LTR   R1,R1                    WERE ANY FIELDS PASSED?
         BZ    RETURN1                  ERROR, RETURN
         USING EFPL,R1
         L     R8,EFPLEVAL              POINTER TO ADDR OF
                                         EVALBLOCK
         L     R8,Ø(R8)                 ADDR OF EVALBLOCK
         USING EVALBLOCK,R8
         L     R3,EFPLARG               REXX ARGS
         L     R4,Ø(R3)                 GET ADDR OF 1ST ARG
         L     R5,ANCHOR                APCBATAB (1ST PCB IS AR)
LOOPP1   EQU   *
*        MODESET KEY=ZERO               MAY OR MAY NOT NEED PKØ
         LA    R5,4(,R5)                BUMP THE PCB POINTER
         CLI   Ø(R5),X'FF'              END OF PCB TABLE?
         BE    ENDPCB1
         L     R6,Ø(R5)                 LOAD A PCB
         LTR   R6,R6                    EMPTY SLOT
         BZ    LOOPP1
         L     R6,Ø(R5)                 LOAD A PCB
         USING PCBADR,R6
         L     R2,PCECOMRA              GET COMREG
         USING COMREG,R2
         MVC   PARTID1,PCELID           SAVE PARTID
         MVC   JNAME,COMNAME            SAVE JOB NAME
         MODESET KEY=USER               RESET PROTECT KEY
         MVI   TRTAB+C' ',X'FF'         SET CHAR FOR SEARCH
         MVI   TRTAB+C'*',X'FF'         SET CHAR FOR SEARCH
         LR    R7,R4                    POINT TO JOBNAME
         LA    R1,8(,R7)                POINT TO MAX POSITION
         TRT   Ø(8,R7),TRTAB            LOOK FOR * OR SPACE
         CLI   Ø(R1),C' '               FOUND SPACE?
         BE    MAXLEN                   YES, USE MAX LEN - 1
         SR    R1,R7                    CALC LENGTH
         LR    R7,R1                    SAVE IT
         BCTR  R7,Ø                     SUBTRACT 1 FOR THE EX
         B     CHKNAME
MAXLEN   EQU   *
         LA    R7,7
CHKNAME  EQU   *
         EX    R7,COMPARE
         BE    FOUND1                   YES, FOUND IT
         B     LOOPP1                   GO CHECK PCB FIELDS
ENDPCB1  EQU   *
```

```
         SR    R15,R15                    NOT FOUND
         LA    R15,Ø                      RETURN NULL LENGTH
         ST    R15,EVALBLOCK_EVLEN        SAVE IT FOR REXX
         MVC   EVALBLOCK_EVDATA(2),=C'  ' SPACES=NOT FOUND
         B     RETURN1                    GO EXIT FUNCTION
FOUND1   EQU   *
         SR    R15,R15                    CLEAR RC
         LA    R15,2                      RETURN LENGTH
         ST    R15,EVALBLOCK_EVLEN        SAVE IT FOR REXX
         MVC   EVALBLOCK_EVDATA(2),PARTID1   RETURN PARTID
RETURN1  EQU   *
         SR    R15,R15
         L     R13,4(R13)                 RESTORE REGS
         LM    R2,R12,28(R13)
         L     R14,12(R13)
         BR    R14                        RETURN
ANCHOR   EQU   X'2C4'                     PCB TABLE ANCHOR
PARTID1  DC    CL2' '
JNAME    DC    CL8' '
COMPARE  CLC   Ø(Ø,R4),JNAME              CHK JOB NAME
TRTAB    DC    256X'ØØ'
         LTORG
         END   ARXFUSER
/*
// IF $RC>8 THEN
// GOTO SKP
// EXEC LNKEDT
/. SKP
/*
/&
```

*Stanley Stewart*
*(USA)*





 

# A CMS utility to facilitate work with VSE libraries

Our installation runs two VSE/ESA machines under VM/ESA. Because much of the administrative work involving VSE libraries is done from CMS, I wrote the Quicklib tools described here in order to simplify matters.

## WHAT CAN QUICKLIB DO?

Quicklib generates a complete directory listing of all the libraries in all the VSE machines, making it easy to find library objects without having to submit LISTDIR jobs. The directory listing is generated automatically by a disconnected CMS machine and is written to a mini-disk which is available to all our users. Although we generate one listing every day, they could be more frequent if required. This procedure is called QL1 EXEC.

Quicklib offers a panel-driven procedure to search for objects in the directory listing and to create a LIBR job for punching, deleting, renaming, or listing library objects by simply selecting them in a list. This job is done by QL2 EXEC which uses the panel QL3 IOS3270.

During the installation of a product, you will often have to punch out individual VSE library members. QL4 EXEC can process a LIBR output file which contains several punched library objects, and create a separate CMS file for each of them.

## INSTALLING AND USING QUICKLIB

QL1 EXEC calls our own submit procedure, SUBS EXEC, which expects the file-id of the file to be submitted and the name of the VSE machine where the jobs should be run. You will have to code a call to your submit procedure.

QL2 EXEC uses IBM’s IOS3270 product to display a panel. If you don’t have this product, you will have to use ISPF or simple PULL statements to display the variable version and to prompt the user for the variables mname, mtype, lib, sublib, and env.

You will have to change the names of the VSE machines in QL1 EXEC.

You will have to prepare the LIBR jobs used by QL1 EXEC.

### QL2 PANEL

## The following panel is displayed by QL2:

```
QuickLib
QuickLib can be used to browse the directory of all VSE LIBR objects
and to generate LIBR jobs for selected objects.
Please fill in the fields below to narrow your selection.

   LIBR data: Version: 7/18/97 13:48:Ø7


 Membername     $a
 Membertype

 Library
 Sublibrary

 Machine                  T:Test  P:Prod  empty:all

 Tip: PUNCH Output from LIBR containing many objects can be
      chopped and written to separate members with procedure QL4

      ENTER:Execute
 PF3:Exit
```

### QL2 SELECTION LIST

## The following selection list is displayed by QL2:

```
WRITE ONE OF THE FOLLOWING COMMANDS IN POS. 1 OF SELECTED LINES:
 L:LIST  P:PUNCH  D:DELETE  R:RENAME  X:OTHER
 THEN START JOB GENERATION WITH PF3
* * * Top of File * * *
   $AJCL    PROC     94-Ø7-27   -  -       1Ø R <PROD> IJSYSRS.SYSLIB
p  $AJCL    PROC     94-Ø7-19   -  -       1Ø R <TEST> IJSYSRS.SYSLIB
   $AJCLEFP PROC     94-Ø7-27 96-Ø8-15     24 R <PROD> IJSYSRS.SYSLIB
p  $AJCLEFT PROC     94-Ø7-19 96-Ø8-13     24 R <TEST> IJSYSRS.SYSLIB
   $AJCLNET PROC     94-Ø7-27   -  -       1Ø R <PROD> IJSYSRS.SYSLIB
p  $AJCLNET PROC     94-Ø7-19   -  -       1Ø R <TEST> IJSYSRS.SYSLIB
   $ASIPROC PROC     94-Ø7-27 96-Ø8-15      3 R <PROD> IJSYSRS.SYSLIB
p  $ASIPROC PROC     94-Ø7-19 96-Ø8-13      4 R <TEST> IJSYSRS.SYSLIB
* * * End of File * * *
```

QL2 JOB

## The following job is generated by QL2:

```
* $$ JOB JNM=SYSHLIBR,DISP=D,CLASS=A
* $$ LST DEST=(*,SYSH),CLASS=W,RBS=Ø
* $$ PUN DEST=(*,SYSH),CLASS=X,RBS=Ø
// JOB SYSHLIBR
// EXEC PGM=LIBR
ACC S=IJSYSRS.SYSLIB
PUNCH   $AJCL.PROC
PUNCH   $AJCLEFT.PROC
PUNCH   $AJCLNET.PROC
PUNCH   $ASIPROC.PROC
/*
/&
* $$ EOJ
```

PROCEDURES

## QL1 EXEC

```
/*====================================================================*/
/* Name        :  QL1      EXEC                                       */
/*====================================================================*/
/* Application :  LIBR Utility                                        */
/*                                                                    */
/* Usage       :  Procedure                                           */
/*                                                                    */
/* Arguments   :  -                                                   */
/*                                                                    */
/* Result      :  -                                                   */
/*                                                                    */
/* Function    :  Generates condensed LIBR Listing for QuickLib       */
/*                                                                    */
/* This procedure runs in a disconnected server machine which         */
/* is autologged by vmutil once a day.                                */
/*                                                                    */
/* It submits a LIBR LISTDIR job to every VSE machine, waits for      */
/* the resulting listings to appear in its reader, receives them,     */
/* and produces the file LIBR LIST which contains a sorted            */
/* directory of all the objects in all the VSE libraries.             */
/*                                                                    */
/* LIBR LIST is then used by the procedure QL2.                       */
/*                                                                    */
/* The LIBR jobs must have the name <shortid> INFOLIB A, where        */
/* <shortid> is either TEST or PROD                                   */
/*                                                                    */
/*====================================================================*/
```

```
/*====================================================================*/
/* inits                                                              */
/*====================================================================*/
jobname='INFOLIB'

/* machine. contains the machine name (word 1) and the short id    */
/* (word 2) of all the VSE machines                                   */
machine.1='ESATEST TEST'
machine.2='ESAPROD PROD'
machine.0=2

ofid='libr list a'
out.0=0

/*====================================================================*/
/* submit libr jobs to all machines                                   */
/*====================================================================*/
do i=1 to machine.0
   call subs  word(machine.i,2) jobname 'a' word(machine.i,1)
end

/*====================================================================*/
/* Receive and process returned reader files                          */
/*====================================================================*/
processed=0

do while processed < machine.0
   'pipe cp q rdr * all | locate /'jobname'/ | stem in.'
   if in.0 > 0
   then
      do
        call process_file
        processed=processed+1
      end
   'cp sleep 5 sec'
end

say 'all reader files have been processed'

'pipe stem out. | pad 80 | sort 1-20 | >' ofid 'fixed 80'
say out.0 'record written to file' ofid

return
/*====================================================================*/
/* receive and process a file                                         */
/*====================================================================*/
process_file:

origin=word(in.1,1)
```

```
spoolid=word(in.1,2)

do i=1 to machine.Ø
   if origin = word(machine.i,1) then shortid=word(machine.i,2)
end

'pipe reader file' spoolid '| specs 2-* 1 | stem r.'
say 'received' r.Ø 'records from machine' origin shortid

SUBLIB=''
DO j=1 to r.Ø
  t=r.j
  IF SUBSTR(T,26,11)='SUBLIBRARY=' THEN SUBLIB=SUBSTR(T,37,17)
  IF SUBSTR(T,21,1)='-' & SUBSTR(T,24,1)='-' & SUBSTR(T,25,1)¬='-'
  THEN
     DO
       X=LEFT(T,44)' <'shortid'> 'SUBLIB
       'pipe var x | stem out. append'
     END
END

return
```

## QL2 EXEC

```
/*===================================================================*/
/* Name        :  QL2      EXEC                                      */
/*===================================================================*/
/* Application :  LIBR Utilities                                     */
/*                                                                   */
/* Usage       :  Procedure                                          */
/*                                                                   */
/* Arguments   :  -                                                  */
/*                                                                   */
/* Result      :  -                                                  */
/*                                                                   */
/* Function    :  QuickLib: browse LIBR LIST and generate jobs       */
/*                                                                   */
/*===================================================================*/

/*===================================================================*/
/* inits                                                             */
/*===================================================================*/
tempfile='lib2 list a'
mname=''
mtype=''
lib=''
sublib=''
env=''
```

```
cursor=''
machine.Ø=2
machine.1='PROD'
machine.2='TEST'

'set cmstype ht'
do i=1 to machine.Ø
  'erase quicklib' machine.i 'a'
end
'set cmstype rt'

librfile='libr list a'  /* the file generated by QL1 EXEC          */

/*====================================================================*/
/* find out when the file was created                                 */
/*====================================================================*/

'pipe cms listfile' librfile '(date | drop first 1 | stem in.'
if in.Ø = Ø
then
   do
     say 'File' librfile 'not found|'
     exit
   end

version='Version:' word(in.1,8) word(in.1,9)

/*====================================================================*/
/* main                                                               */
/*====================================================================*/

do forever
   call panel1
   if iosk = 'PFØ3' then leave
   call select_members
   call edit_list
   do ixØ3=1 to machine.Ø
      call process_env machine.ixØ3
   end
   'filelist quicklib * a'
   leave
end

'erase' tempfile

return


/*====================================================================*/
/* display and process panel                                          */
```

```
/*====================================================================*/
PANEL1:

IF CURSOR ¬= ''
THEN
   DO
     OPTIONS='(UPDATE 'CURSOR
     ALARM='.A'
   END
ELSE
   DO
     OPTIONS=''
     ALARM=''
   END
 'IOS327Ø QL3 ' OPTIONS
IF RC ¬= Ø
THEN
   DO
     SAY 'IOS327Ø ERROR:' RC
     EXIT
   END
MELDUNG=''
CURSOR=''


return


/*====================================================================*/
/* extract selected objects from LIBR LIST                            */
/*====================================================================*/
select_members:

mname=translate(strip(translate(mname,'','*')))
mtype=translate(strip(translate(mtype,'','*')))
lib=translate(strip(lib))
sublib=translate(strip(sublib))
env=translate(env)
select
  when env = 'T' then env='TEST'
  when env = 'P' then env='PROD'
  otherwise           env=''
end
slib=strip(lib'.'sublib,'T','.')

pipestring='<' librfile

if length(mname) > Ø
then
   do
     endpos=1+length(mname)-1
```

```
     pipestring=pipestring '| locate 1-'endpos' /'mname'/'
   end

if length(mtype) > 0
then
   do
     endpos=10+length(mtype)-1
     pipestring=pipestring '| locate 10-'endpos' /'mtype'/'
   end

if length(env) > 0
then
   do
     endpos=47+length(env)-1
     pipestring=pipestring '| locate 47-'endpos' /'env'/'
   end

if length(slib) > 0
then
   do
     endpos=53+length(slib)-1
     pipestring=pipestring '| locate 53-'endpos' /'slib'/'
   end

pipestring=pipestring '| specs 77-80 1 1-77 4',
   'pad 80 | >' tempfile 'fixed 80'


'pipe' pipestring
return


/*====================================================================*/
/* let the user edit the object list                                  */
/*====================================================================*/
edit_list:


QUEUE 'SET RESERVED 1 Yellow REV NO ',
'Write one of the following commands in pos. 1 of selected lines:'
QUEUE 'SET RESERVED 2 REV NO ',
'L:List  P:Punch  D:Delete  R:Rename  X:other'
QUEUE 'SET RESERVED 3 Red REV NO ',
'Then start job generation with PF3'
queue 'set prefix off'
'xedit'  tempfile

return

/*====================================================================*/
/*  build a job for a machine                                         */
```

```
/*====================================================================*/
process_env:
arg enx

outnum=Ø

jobname=left(userid(),4)'LIBR'
call out '* $$ JOB JNM='jobname',DISP=D,CLASS=A'
call out '* $$ LST DEST=(*,'userid()'),CLASS=W,RBS=Ø'
call out '* $$ PUN DEST=(*,'userid()'),CLASS=X,RBS=Ø'
call out '// JOB 'jobname
call out '// EXEC PGM=LIBR'



'pipe <' tempfile '| locate 5Ø-53 /'enx'/ | nlocate 1-3 /   /' ,
       '| sort 56-74 4-2Ø | stem x.'

lastlib=''

do i=1 to x.Ø
   mid=strip(substr(x.i,4,8))'.'strip(substr(x.i,13,8))
   lib=strip(substr(x.i,56,17))
   select
     when word(x.i,1) = 'L' then cmd='LIST   '
     when word(x.i,1) = 'P' then cmd='PUNCH  '
     when word(x.i,1) = 'D' then cmd='DELETE '
     when word(x.i,1) = 'R' then cmd='RENAME '
     otherwise                   cmd='????   '
   end
   if lib ¬= lastlib
   then
      do
        call out 'ACC S='lib
        lastlib=lib
      end
   call out cmd mid
end

call out '/*'
call out '/&'
call out '* $$ EOJ'

if outnum > Ø
then
   do
     OFID='QUICKLIB' ENX 'A'
     outstem.Ø=outnum
     'pipe stem outstem. | pad 8Ø | >' ofid 'fixed 8Ø'
     say 'File' ofid 'created'
```

```
   end

return

/*====================================================================*/
/* write an output line                                               */
/*====================================================================*/
out:

arg outline

outnum=outnum + 1
outstem.outnum=outline

return
```

## QL3

```
;*====================================================================*/
;* Name         :  QL3      IOS327Ø                                   */
;*====================================================================*/
;* Application :  LIBR Utility                                        */
;*                                                                    */
;* Usage       :  IOS327Ø Panel                                       */
;*                                                                    */
;* Arguments   :  -                                                   */
;*                                                                    */
;* Result      :  -                                                   */
;*                                                                    */
;* Function    :  Panel for QuickLibr                                 */
;*                                                                    */
;*                                                                    */
;*====================================================================*/
.n
.y
.F F1 F2 F3 F4 F5 F6 F7 F8 F9 F1Ø F11 F12
.JX SET CTL . off
.jx set  normal intensity color=white
.jx set  high   intensity color=green
.JX SET CTL | HIG=underline col=yel TYPE=(SKIP UNPROTECTED NULLS)
.JX SET CTL | col=red TYPE=(SKIP UNPROTECTED NULLS)
.JX SET CTL % col=red
.JX SET CTL # col=green
.JX SET CTL } col=green hig=rev
.e|
.&alarm
.c
}QuickLib                                                           |
```

QuickLib can be used to browse the directory of all VSE LIBR objects

```
and to generate LIBR jobs for selected objects.

Please fill in the fields below to narrow your selection.

  LIBR data:%&version


Membername    |8&mname
Membertype    |8&mtype

Library       |8&lib
Sublibrary    |8&sublib

Machine       |1&env     T:Test  P:Prod  empty:all

Tip: PUNCH Output from LIBR containing many objects can be
     chopped and written to separate members with procedure QL4

.b
     ENTER:Execute                                              PF3:Exit
```

## QL4

```
/*================================================================*/
/* Name        :  QL4      EXEC                                   */
/*================================================================*/
/* Application :  LIBR Utilities                                  */
/*                                                                */
/* Usage       :  Procedure                                       */
/*                                                                */
/* Arguments   :  spoolid                                         */
/*                                                                */
/* Result      :  -                                               */
/*                                                                */
/* Function    :  Create separate files fron LIBR PUNCH output    */
/*                                                                */
/* This procedure receives a reader file containing the output    */
/* from LIBR PUNCH. For every punched library member, it creates  */
/* a CMS file                                                     */
/*================================================================*/
arg spoolid

dip   'filemode for output files?'
pull outmode

dip   'replace option:   1=prompt  2=replace'
pull replopt

dip   'write CATALOG and /+ /* to the files (Y/N) ?'
pull  outopt
```

```
'vmfclear'

'pipe reader file' spoolid '| stem inrec.'

mid=''
orec.=''
oc=Ø

do i=1 to inrec.Ø
   inrec.i=substr(inrec.i,2)
   if pos('CATALOG',word(inrec.i,1)) > Ø
   then
      do
        if mid ¬= '' then call create_member
        mid=word(inrec.i,2)
        oc=1
        outrec.=''
        outrec.1=inrec.i
      end
   else
      do
        oc=oc+1
        outrec.oc=inrec.i
      end
end

if mid ¬= '' then call create_member

return

create_member:

parse value mid with mn '.' mt
ofile=mn mt outmode

'SET CMSTYPE HT'
'STATE' ofile
STATERC=RC
'SET CMSTYPE RT'
IF STATERC=Ø & replopt = 1
THEN
   do
     'vmfclear'
     say 'file' ofile 'exists|',
         'please enter different file id'
     pull ofile
     if words(ofile) < 3 then ofile=ofile outmode
   end
```

```
outrec.Ø=oc
if outopt = 'N'
then
  'pipe stem outrec. | drop first 1 | drop last 2 | stem outrec.'
'pipe stem outrec. | pad 8Ø | specs 1-8Ø 1 | >' ofile 'fixed 8Ø'
say 'File' ofile 'was created'

return
```

TEST INFOLIB A

The following is one of the jobs submitted by QL1 EXEC:

```
* $$ JOB JNM=INFOLIB,CLASS=A,DISP=D
* $$ LST CLASS=W,FNO=MAKU,DEST=(*,SYSH),RBS=Ø
// JOB INFOLIB
// EXEC PGM=LIBR,SIZE=256K
LISTD L=EFV PRD1 PRD2 IJSYSRS
/*
/&
* $$ EOJ
```



## *VSE Update* contributions

Tell us what you have done to make working with VSE easier or quicker at your site. We welcome contributions from VSE novices as well as from more experienced users, and are keen to receive anything from very short 'hints and tips' type articles to longer discussion articles and example code.

Articles for *VSE Update* can be sent to the editor, Fiona Hewitt, at any of the addresses shown on page 2. Alternatively, articles can be sent using the Internet to 100336.1412@compuserve.com.

Remember that we pay $250 (£170) per 1,000 words and $140 (£90) per 100 lines of code published (if you give us copyright).

Computer Associates has started shipping a new version of CA-Impact/2000. The new version uses new scanning technology to pinpoint non-compliant millennium-sensitive code in any mainframe language, including COBOL, Assembler, FORTRAN, PL/I, and 4GLs from CA and other vendors.

For further information, contact:
Computer Associates, 1 Computer Associates Plaza, Islandia, NY 11788-7000, USA.
Tel: (516) 342 5224.
Computer Associates, 183-187 Bath Road, Slough, SL1 4AA, UK.
Tel: (1753) 577733.

* * *

Software AG has announced that its data warehouse manager system, SourcePoint, will support SAP's Business Application Program Interface (BAPI) once it arrives.

The company has also announced that it now provides full Year 2000 support for its ADABAS database and Natural application development products.

For further information, contact:
Software AG, 11190 Sunrise Valley Drive, Reston VA 22091, USA.
Tel: (703) 860 5050.
Software AG, Charter Court, 74-78 Victoria Street, St Albans, AL1 3XH, UK.
Tel: (1727) 844455.

* * *

Syncsort has announced Release 2 of its FilePort automated tool for translating data during mainframe-to-Unix migration projects and for preparing data for data warehouses. All processing is done on the Unix system, with the product requiring only data plus a record layout as input.

For further information, contact:
Syncsort, 50 Tice Boulevard, Woodcliff Lake, NJ 07675, USA.
Tel: (201) 930 8200.
Syncsort Ltd, 60 Churchill Square, Kings Hill, West Malling, Kent, ME 19 4DU, UK.
Tel: (01732) 849000.

* * *

IBM has added Sterling Software's VM: Webserver Gateway to its portfolio of mainframe access solutions. The product enables CGI scripts to communicate with a VM server through the existing application client. It also lets CGI scripts provide Web interfaces to all applications running on VSE, VM, and OS/390.

For further information, contact your local IBM representative.

* * *

Hitachi Data Systems has enhanced the storage management capabilities in its 7700 storage array so that the synchronous Hitachi Remote Copy feature can now transfer data between System/370 and System/390 processors running versions of VSE/ESA that don't support peer-to-peer remote copy.

For further information, contact:
Hitachi Data Systems, PO Box 54996, 750 E Central Expressway, Santa Clara, CA 95056-0996, USA.
Tel: (408) 970 1000.
Hitachi Data Systems, Sefton Park, Bells Hill, Stoke Poges, Buckinghamshire, SL2 4HB, UK.
Tel: (1753) 618000.