30 VSAM update

July 1998

In this issue

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

Subscriptions and back-issues
A year's subscription to *VSAM Update*, comprising four quarterly issues, costs £115.00 in the UK; $170.00 in the USA and Canada; £121.00 in Europe; £128.00 in Australasia and Japan; and £125.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the April 1991 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Editorial panel
Articles published in *VSAM Update* are reviewed by our panel of experts. Members of the panel include John Bradley (UK), Ernie Ishman (USA), and Rem Perretta (UK).

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.

Contributions
Articles published in *VSAM Update* are paid for at the rate of £170 ($250) per 1000 words and £90 ($140) per 100 lines of code 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*.

VSAM Update on-line
Code from *VSAM 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.

KEYLIST – a utility to list VSAM keys (continued)

Here, we conclude the program begun in the last issue, which lists the keys from VSAM KSDS files.

```
*********************************************************************
***                                                               ***
***   ANALYSE PARM= AND CONVERT TO OPTIONS BITS.                  ***
***                                                               ***
*********************************************************************
*
GETPARMS ST    RBAL,SAVGPBAL       SAVE LINKAGE REGISTER
*
         MVI   OPTIONS,LISTBIT     SET DEFAULT OPTION TO 'LIST'
*
         L     R2,R1SAVE           POINT TO PARAMETER LIST
         L     R2,Ø(Ø,R2)          POINT TO PARAMETER
*
         LH    R1,Ø(Ø,R2)          LENGTH OF PARAMETER
         LTR   R1,R1               ZERO LENGTH?
         BZ    GPRETURN             YES
*
         CLC   =C'HEX',2(R2)       IS HEX SPECIFIED?
         BNE   GPNOTHEX             NO
*
         OI    OPTIONS,HEXBIT      SET HEX OPTION
         LA    R2,4(R2)            SKIP 'HEX,'
*
         CH    R1,=H'3'            ONLY 'HEX' SPECIFIED?
         BE    GPRETURN             YES
*
GPNOTHEX CLC   =C'OPTION=',2(R2)   OPTION=?
         BNE   PARMERR              NO
*
         CLI   9(R2),C'L'          LIST OPTION?
         BE    GPRETURN            GO OPEN VSAM FILE(S)
*
GPNOTLST CLI   9(R2),C'M'          MATCH OPTION?
         BNE   GPNOTMTC             NO
*
         OI    OPTIONS,MATCHBIT    SET LIST OPTION
         B     GPCLEAR             GO CLEAR 'LIST' OPTION
*
GPNOTMTC CLI   9(R2),C'U'          UNIQUE OPTION?
         BNE   GPRETURN             NO
*
         OI    OPTIONS,UNIQUBIT    SET LIST OPTION
```

```
*
GPCLEAR  NI    OPTIONS,X'FF'-LISTBIT TURN LIST BIT OFF
*
GPRETURN L     RBAL,SAVGPBAL       RESTORE LINKAGE REGISTER
         BR    RBAL                RETURN
*
         EJECT
*********************************************************************
***                                                               ***
***   PUTS KEY INTO 'LINES' ARRAY.  IF HEX OPTION, SHOW VERTICAL  ***
***   HEX.  PRINT PAGE WHEN FULL.                                 ***
***                                                               ***
*********************************************************************
*
PUTKEY   ST    RBAL,SAVPKBAL       SAVE LINKAGE REGISTER
*
         L     R2,LINEPTR          GET CURRENT LINE POINTER
         L     R15,=A(LPP*L'LINES) SIZE OF 'LINES' ARRAY
         LA    R15,LINES(R15)      POINT PAST END OF ARRAY
*
         CR    R2,R15              PAST BOTTOM OF PAGE?
         BL    PKPAGEOK             NO
*
         LA    R2,LINES            POINT TO FIRST LINE
         ST    R2,LINEPTR          SAVE
*
         LH    R1,COLPTR           GET CURRENT COLUMN POINTER
         AH    R1,KEYLENMN         INCREMENT POINTER
         LA    R1,2(R1)            POINT TO NEXT COLUMN
         STH   R1,COLPTR           SAVE COLUMN POINTER
         CH    R1,LASTCOL          ROOM FOR ANOTHER COLUMN?
         BNH   PKPAGEOK             YES
*
         BAL   RBAL,PRTPAGE        PRINT PAGE
*
PKPAGEOK LH    R1,KEYLENM1         KEY LENGTH - 1
         L     R2,LINEPTR          LINE POSITION
         AH    R2,COLPTR           +COLUMN POSITION
*
         MVC   Ø(1,R2),IDENT       INDICATE IF 'MATCH' RECORD IDENTICAL
         MVI   IDENT,C' '          CLEAR INDICATOR
*
         L     R15,IN1LOC          LOAD LOCATION OF INPUT2 RECORD
         A     R15,IN1RKP          ADD OFFSET TO KEY
*
         LA    R14,NUMBTAB         FOR ADDRESSABILITY
         EX    R1,PKMVCKEY         MOVE KEY TO LINE(S)
         EX    R1,PKTRKEY          TRANSLATE TO PRINTABLE CHARACTERS
```

```
*
         L     R2,LINEPTR          CURRENT LINE
         LA    R2,L'LINES(R2)      POINT TO NEXT LINE
         CH    R1,=AL2(L'LINES)    DID KEY FIT ON 1 LINE?
         BL    PKFITOK1             YES
*
         LA    R2,L'LINES(R2)      POINT TO NEXT LINE
*
PKFITOK1 ST    R2,LINEPTR          SAVE NEXT LINE POSITION
*
         TM    OPTIONS,HEXBIT      HEX LISTING?
         BZ    PKRETURN             NO
*
         AH    R2,COLPTR           ADD COLUMN POSITION TO LINE POINTER
         EX    R1,PKMVCKEY         MOVE KEY TO LINE(S)
         EX    R1,PKTRZONE         TRANSLATE ZONE TO HEX
*
         L     R2,LINEPTR          CURRENT LINE
         LA    R2,L'LINES(R2)      POINT TO NEXT LINE
         CH    R1,=AL2(L'LINES)    DID KEY FIT ON 1 LINE?
         BL    PKFITOK2             YES
*
         LA    R2,L'LINES(R2)      POINT TO NEXT LINE
*
PKFITOK2 ST    R2,LINEPTR          SAVE NEXT LINE POSITION
*
         AH    R2,COLPTR           ADD COLUMN POSITION TO LINE POINTER
         EX    R1,PKMVCKEY         MOVE KEY TO LINE(S)
         EX    R1,PKTRNUMB         TRANSLATE TO PRINTABLE CHARACTERS
*
         L     R2,LINEPTR          CURRENT LINE
         LA    R2,L'LINES(R2)      POINT TO NEXT LINE
         CH    R1,=AL2(L'LINES)    DID KEY FIT ON 1 LINE?
         BL    PKFITOK3             YES
*
         LA    R2,L'LINES(R2)      POINT TO NEXT LINE
*
PKFITOK3 ST    R2,LINEPTR          SAVE NEXT LINE POSITION
*
PKRETURN L     RBAL,SAVPKBAL       RESTORE LINKAGE REGISTER
         BR    RBAL                RETURN
*
PKMVCKEY MVC   1(*-*,R2),Ø(R15)
PKTRKEY  TR    1(*-*,R2),CHARTAB-NUMBTAB(R14)
PKTRZONE TR    1(*-*,R2),ZONETAB-NUMBTAB(R14)
PKTRNUMB TR    1(*-*,R2),Ø(R14)
*
         EJECT
```

```
*********************************************************************
***                                                               ***
***   IF OPTION=LIST, READ INPUT1 AND LIST KEYS.                  ***
***                                                               ***
*********************************************************************
*
DOLIST   ST    RBAL,SAVDLBAL       SAVE LINKAGE REGISTER
*
         TM    OPTIONS,LISTBIT     OPTION=LIST?
         BZ    DLRETURN             NO
*
DLREST   BAL   RBAL,READ1          READ RECORD FROM INPUT1
*
         L     R1,IN1LOC           LOAD LOCATION OF INPUT1 RECORD
         A     R1,IN1RKP           ADD OFFSET TO KEY
*
         BAL   RBAL,PUTKEY         GO PUT KEY IMAGE IN PRINT LINE ARRAY
*
         B     DLREST              GO GET ANOTHER PAIR OF RECORDS
*
DLRETURN L     RBAL,SAVDLBAL       RESTORE LINKAGE REGISTER
         BR    RBAL                RETURN
*
         EJECT
*********************************************************************
***                                                               ***
***   SETS PRINT LINE WITH DDNAME, FORMATS RECORD COUNT,          ***
***   SETS DSNAME, AND PRINTS LINE.                               ***
***                                                               ***
*********************************************************************
*
GETNAME  ST    RBAL,SAVGNBAL       SAVE LINKAGE REGISTER
*
         LA    R3,IN1DDN-INPUT1(R2) POINT TO SHOWCB RETURN AREA
*
         SHOWCB ACB=(R2),AREA=(R3),FIELDS=(DDNAME),                    -
               LENGTH=8,OBJECT=DATA,MF=(G,SHOWCB7,LSHOWCB7)
*
         SHOWCB ACB=(R2),AREA=(R3),MF=(E,SHOWCB7)  GET DDNAME
*
         MVC   LINE+1(6),IN1DDN-INPUT1(R2)  SET DDNAME IN PRINT LINE
         MVC   LINE+8(7),=C'RECORDS'        SET RECORD COUNT ID
         MVC   LINE+16(6),=X'202060B202120'  SET EDIT PATTERN
         ED    LINE+15(7),COUNT1-INPUT1(R2) FORMAT RECORD COUNT
*
         L     R15,IN1KEYL-INPUT1(R2) LOAD KEY LENGTH
         CVD   R15,DOUBLE             CONVERT TO DECIMAL
         MVC   LINE+30(3),=X'202120'  SET EDIT PATTERN
```

```
         ED    LINE+29(4),DOUBLE+6    FORMAT KEY LENGTH
         MVC   LINE+23(6),=C'KEYLEN'  IDENTIFY VALUE
*
         L     R15,IN1RKP-INPUT1(R2)  LOAD KEY LENGTH
         CVD   R15,DOUBLE             CONVERT TO DECIMAL
         MVC   LINE+38(6),=X'202062021200'  SET EDIT PATTERN
         ED    LINE+37(7),DOUBLE+6    FORMAT KEY LENGTH
         MVC   LINE+34(3),=C'RKP'     IDENTIFY VALUE
*
         XR    R15,R15             ADDRESS OF PSA
         USING PSA,R15             ESTABLISH ADDRESSABILITY
         L     R14,FLCCVT          ADDRESS OF CVT
         DROP  R15                 DROP ADDRESSABILITY TO PSA
         USING CVTMAP,R14          ESTABLISH ADDRESSABILITY TO CVT
         L     R15,CVTTCBP         ADDRESS OF NEXT TCB POINTER
         L     R15,4(0,R15)        ADDRESS OF CURRENT TCB
         DROP  R14                 DROP ADDRESSABILITY TO CVT
         USING TCB,R15             ESTABLISH ADDRESSABILITY CURRENT TCB
         L     R14,TCBTIO          ADDRESS OF TIOT
         USING TIOT,R14            ESTABLISH ADDRESSABILITY TO TIOT
*
         DROP  R15                 DROP ADDRESSABILITY TO TCB
         LA    R15,TIOELNGH        ADDRESS OF FIRST TIOT ENTRY
         DROP  R14                 DROP ADDRESSABILITY (HLASM OBJECTS)
         USING TIOENTRY,R15        ESTABLISH ADDRESSABILITY TO TIOT
*
GNTIOTLP CLI   TIOELNGH,X'00'      END OF TIOT CHAIN?
         BE    GNFINISH            YES (SHOULDN'T HAPPEN)
         CLC   TIOEDDNM(8),IN1DDN-INPUT1(R2) DDNAME FOUND?
         BE    GNDSN               YES
         XR    R0,R0               CLEAR REGISTER
         IC    R0,TIOELNGH         INSERT ENTRY LENGTH
         AR    R15,R0              POINT TO NEXT ENTRY
         B     GNTIOTLP            CONTINUE
*
GNDSN    XR    R1,R1               CLEAR REGISTER
         ICM   R1,7,TIOEJFCB       ADDRESS OF JFCB
         USING JFCB,R1             ESTABLISH ADDRESSABILITY TO JFCB
         MVC   LINE+45(4),=C'DSN=' SET DSN ID IN HEADER
         MVC   LINE+49(44),JFCBDSNM MOVE DSNAME TO HEADER
         DROP  R1,R15              DROP ADDRESSING TO JFCB,TIOT,ENTRY
*
GNFINISH BAL   RBAL,PRINT                  PRINT TOTALS, ETC
*
         L     RBAL,SAVGNBAL       RESTORE LINKAGE REGISTER
         BR    RBAL                RETURN
*
         EJECT
```

```
*********************************************************************
***                                                               ***
***   READ RECORD FROM 'RANGES' FILE.                             ***
***                                                               ***
*********************************************************************
*
GETRANGE ST    RBAL,SAVGRBAL       SAVE LINKAGE REGISTER
*
         TM    EOFFLAGS,X'8Ø'      E-O-F REACHED?
         BO    GRRETURN             YES
*
         NI    OPTIONS,X'FF'-EXCLBIT  TURN OFF 'EXCLUDE' FLAG
*
GRNEXT   GET   RANGES,RINAREA      READ RECORD
         BAL   RBAL,FINDWILD       GO PROCESS RANGE DATA
*
         CLC   =C'FROM',RINAREA    'FROM' OPTION?
         BNE   GRNOTF               NO
*
         MVC   FROMLEN,RANGELEN    MOVE LENGTH OF 'FROM' KEY
         MVC   FROMKEY,RINAREA+5   MOVE 'FROMKEY' VALUE
         OI    OPTIONS,FROMBIT     SET 'FROMKEY' OPTION ON
*
         B     GRNEXT              GO GET NEXT RANGE RECORD
*
GRNOTF   CLC   =C'THRU',RINAREA    'THRU' OPTION?
         BNE   GRNOTT               NO
*
         MVC   THRULEN,RANGELEN    MOVE LENGTH OF 'THRU' KEY
         MVC   THRUKEY,RINAREA+5   MOVE 'THRUKEY' VALUE
         OI    OPTIONS,THRUBIT     SET 'THRUKEY' OPTION ON
*
         B     GRNEXT              GO GET NEXT RANGE RECORD
*
GRNOTT   CLC   =C'MAXL',RINAREA    'THRU' OPTION?
         BNE   GRNOTM               NO
*
         LH    R1,RANGELEN         LOAD LENGTH-1 OF DECIMAL VALUE
         EX    R1,RVNUMBER         IS IT NUMERIC?
         BNZ   RVBADKEY             NO
         EX    R1,RVPACK           PACK VALUE
         CP    MAXKEYL,=P'256'     TOO LARGE?
         BL    GRNEXT               NO
*
RVBADKEY ZAP   MAXKEYL,=P'Ø'       IGNORE INPUT
         B     GRNEXT              GO GET NEXT RANGE RECORD
*
RVPACK   PACK  MAXKEYL,RINAREA+5(*-*)
```

```
RVNUMBER TRT   ZONETAB+1Ø,RINAREA+5
*
GRNOTM   CLC   =C'EXCL',RINAREA    'EXCL' OPTION?
         BE    GREXCL               YES
         CLC   =C'FIND',RINAREA    'FIND' OPTION?
         BNE   GRNEXT               NO, TREAT AS COMMENT
*
GREXCL   MVC   EXCLLEN,RANGELEN    MOVE LENGTH OF 'EXCL' KEY
         MVC   EXCLKEY,RINAREA+5   MOVE 'EXECKEY' VALUE
         OI    OPTIONS,EXCLBIT     SET 'THRUKEY' OPTION ON
*
GRRETURN L     RBAL,SAVGRBAL       RESTORE LINKAGE REGISTER
         BR    RBAL                RETURN
*
RGEOF    TM    EOFFLAGS,X'4Ø'      WAS THIS SECOND PASS (DOTOTALS?
         BO    DTRETURN             YES
*
         OI    EOFFLAGS,X'8Ø'      FLAG EOF FOR 'RANGES'
         B     GRRETURN
*
         EJECT
*********************************************************************
***                                                               ***
***   GO PROCESS STATEMENT FOR WILD-CARD END                      ***
***                                                               ***
*********************************************************************
*
FINDWILD ST    RBAL,SAVFWBAL       SAVE LINKAGE REGISTER
*
         XC    TRTWORK,TRTWORK     CLEAR TRT TABLE
         XR    R1,R1               CLEAR REGISTER
         IC    R1,RINAREA+4        SET WILD CARD CHARACTER
         LA    R1,TRTWORK(R1)      POINT TO RELATIVE POSITION
         MVI   Ø(R1),X'FF'         SET POSITION ON
*
         LA    R1,RINAREA+L'RINAREA   POINT TO LAST POSITION
         LA    RØ,RINAREA+6           POINT TO BEGINNING POSITION
         TRT   RINAREA+6(L'RINAREA-6),TRTWORK FIND POSSIBLE WILD CARD
*
         SR    R1,RØ               COMPUTE LENGTH - 1
         LH    RØ,KEYLENM1         KEY LENGTH - 1
         CR    R1,RØ               IS WILD CARD WITHIN KEY LENGTH?
         BNH   FWFINISH             YES
         LR    R1,RØ               SET TO KEY LENGTH - 1
*
FWFINISH STH   R1,RANGELEN         SAVE LENGTH OF RANGE KEY
*
         L     RBAL,SAVFWBAL       RESTORE LINKAGE REGISTER
```

```
         BR    RBAL                RETURN
*
* END STUB DEFINE
*
*
         EJECT
*********************************************************************
***                                                               ***
***  ERROR RETURNS                                                ***
***                                                               ***
*********************************************************************
*
VSAMCERR MVC   LINE+1(OERRORL),OERROR SET MESSAGE
         MVC   OECLOSE-OERROR+LINE+1(5),=C'CLOSE'  CHANGE OPEN TO CLOSE
         B     VEFMTRET            GO FORMAT RETURN CODE, ETC
*
VSAMOERR MVC   LINE+1(OERRORL),OERROR SET MESSAGE
*
VEFMTRET CVD   R15,DOUBLE                    CONVERT REG15 TO DECIMAL
         ED    OEREG15-OERROR+LINE+1,DOUBLE+6 FORMAT REG 15 VALUE
*
         LA    R3,VSAMRETC         POINT TO AREA
*
         SHOWCB ACB=(R2),AREA=(R3),FIELDS=(ERROR,DDNAME),              -
               LENGTH=12,OBJECT=DATA,MF=(G,SHOWCB4,LSHOWCB4)
*
         SHOWCB ACB=(R2),AREA=(R3),MF=(E,SHOWCB4)
*
         MVC   OEDDNAME-OERROR+LINE+1,DDNAME SET DDNAME
         L     R1,VSAMRETC                   LOAD RETURN CODE
         CVD   R1,DOUBLE                     CONVERT TO DECIMAL
         ED    OERC-OERROR+LINE+1,DOUBLE+6   FORMAT RETURN CODE
*
         B     ERROR
*
VSAMGERR MVC   LINE+1(GERRORL),GERROR SET MESSAGE
         CVD   R15,DOUBLE                           CNVRT REG15 TO DEC
         ED    GEREG15-GERROR+LINE+1,DOUBLE+6 FORMAT REGISTER 15
                VALUE
*
         LA    R3,VSAMACB          POINT TO AREA
*
         SHOWCB RPL=(R2),AREA=(R3),FIELDS=(ACB,FDBK),                  -
               LENGTH=8,OBJECT=DATA,                                   -
               MF=(G,SHOWCB5,LSHOWCB5) GENERATE SHOWCB FOR FOLLOWING
*
         SHOWCB RPL=(R2),MF=(E,SHOWCB5) GET ACB ADDRESS, FEEDBACK DATA
*
```

```
         XR    R1,R1               CLEAR REGISTER
         IC    R1,VSAMRETC+3       LOAD RETURN CODE
         CVD   R1,DOUBLE           CONVERT TO DECIMAL
         ED    GERC-GERROR+LINE+1,DOUBLE+6   FORMAT RETURN CODE
*
         UNPK  GEFDBK-GERROR+LINE+1(L'GEFDBK+1),VSAMRETC (L'VSAMRETC+1)
         MVI   GEFDBK-GERROR+LINE+L'GEFDBK+1,C'.' TERMINATE LINE
         TR    GEFDBK-GERROR+LINE+1,NUMBTAB       FORMAT HEX DISPLAY
*
         L     R2,VSAMACB          GET ADDRESS OF ACB
         LA    R3,DDNAME           POINT TO AREA
*
         SHOWCB ACB=(R2),AREA=(R3),FIELDS=DDNAME,LENGTH=8,             -
               OBJECT=DATA,MF=(G,SHOWCB6,LSHOWCB6)
*
         SHOWCB ACB=(R2),MF=(E,SHOWCB6) GET DDNAME
*
         MVC   GEDDNAME-GERROR+LINE+1,DDNAME SET DDNAME
*
         B     ERROR               GO PROCESS ERROR
*
PARMERR  MVC   LINE(27),=C'ØINVALID PARAMETER'
         B     ERROR               GO PROCESS
*
ERROR    LA    RØ,16               SET COMPLETION CODE
         ST    RØ,COMPCODE         SAVE
*
         BAL   RBAL,HEADPAGE       EJECT PAGE
*
         BAL   RBAL,PRINT          PRINT ERROR MESSAGE
*
         DC    H'Ø'                BOMB WITH ALL AREAS AVAILABLE
*
         B     ENDING              GO EXIT
*
         EJECT
*********************************************************************
***                                                               ***
***      PRINT ROUTINE                                            ***
***                                                               ***
*********************************************************************
*
PRINT    PUT   PRINTER,LINE        PRINT LINE
         MVI   LINE,C' '           SET SEED
         MVC   LINE+1(L'LINE),LINE CLEAR LINE
DOUBLESP BCTR  R9,RBAL             RETURN IF PAGE NOT FULL
*
HEADPAGE MVC   PAGENO,=X'4Ø2Ø212Ø' SET EDIT PATTERN
```

```
         ED    PAGENO,PAGES        FORMAT PAGE NUMBER
         AP    PAGES,=P'1'         INCREMENT PAGE COUNT
         PUT   PRINTER,HEADER      PRINT PAGE HEADING
         LA    R9,LPP              SET LINES/PAGE
         MVI   LINE,C'Ø'           SET TO DOUBLE SPACE AFTER HEADER
         BR    RBAL                RETURN
*
LPP      EQU   54  MUST BE MULTIPLE OF 6 (2 LINES/KEY * 3 LINES IF HEX)
*
         EJECT
*********************************************************************
***                                                               ***
***      FIXED DATA AREA                                          ***
***                                                               ***
*********************************************************************
*
HEAD     DC    C'1&MYNAME - LIST VSAM KEYS. '
*
OERROR   DC    C'VSAM '
OECLOSE  DC    C'OPEN  ERROR.  FILE '
OEDDNAME DC    CL8'........'
         DC    C'  REG 15='
OEREG15  DC    X'4Ø2Ø212Ø'
         DC    C'  VSAM RETURN CODE='
OERC     DC    X'4Ø2Ø212Ø'
         DC    C'.'
OERRORL  EQU   *-OERROR
*
GERROR   DC    C'VSAM ERROR.  FILE '
GEDDNAME DC    CL8'........'
         DC    C'  REG 15='
GEREG15  DC    X'4Ø2Ø212Ø'
         DC    C'  VSAM REASON CODE='
GERC     DC    X'4Ø2Ø212Ø'
         DC    C'  FEADBACK='
GEFDBK   DC    CL8' '
         DC    C'.'
GERRORL  EQU   *-GERROR
*
OPEND    OPEN  (,),MF=L
CLOSED   CLOSE (),MF=L
*
* BEGIN DCB CONSTANTS
*
PRINTERD DCB   DDNAME=PRINTER,DEVD=DA,DSORG=PS,LRECL=133,             -
               BLKSIZE=133,MACRF=(PM),RECFM=FBA
*
INPUT1D  ACB   AM=VSAM,DDNAME=INPUT1,MACRF=(KEY,SEQ),EXLST=IN1EXLST
```

```
*
IN1EXLST EXLST EODAD=I1EOF
*
INPUT2D  ACB   AM=VSAM,DDNAME=INPUT2,MACRF=(KEY,SEQ),EXLST=IN2EXLST
*
IN2EXLST EXLST EODAD=I2EOF
*
RANGESD  DCB   DDNAME=RANGES,DSORG=PS,MACRF=GM,EODAD=RGEOF
*
* END DCB CONSTANTS
*
JGMOTBLD DC    PL2'Ø,31,28,31,3Ø,31,3Ø,31,31,3Ø,31,3Ø,31'
*
* END CONSTANTS
*
         LTORG
*
PACKLEN  PACK  DOUBLE,2(*-*,R2)    PACK LENGTH OF KEY
MOVELEN  MVC   HEADER+L'HEAD-1(*-*),2(R2) MOVE LENGTH OF KEY TO HEADER
CHECKNUM TRT   2(*-*,R2),TRTABLE   CHECK FOR NUMERIC
*
TRTABLE  DC 256X'FF'
         ORG   TRTABLE+C'Ø'
         DC    1ØX'Ø'
         ORG
*
NUMBTAB  DC    16C'Ø123456789ABCDEF'
*
ZONETAB  DC    16C'Ø',16C'1',16C'2',16C'3',16C'4',16C'5',
                16C'6',16C'7'
         DC    16C'8',16C'9',16C'A',16C'B',16C'C',16C'D',
                16C'E',16C'F'
*
         DC    1ØX'Ø'              USED FOR NUMERIC TEST
*
CHARTAB  DS    ØCL256              CHANGE FOLLOWING TABLE FOR SPECIFIC
*                Ø123456789ABCDEF         PRINT/DISPLAY DEVICE
         DC    C'................'  Ø
         DC    C'................'  1
         DC    C'................'  2
         DC    C'................'  3
         DC    C' ...............'  4
         DC   C'&&.........|$*);¨'  5
         DC    C'-/........|,%_>?'  6
         DC    C'.........`:#@''="' 7
         DC    C'................'  8
         DC    C'................'  9
         DC    C'.~.............'  A
```

```
         DC    C'................'  B
         DC    C'.ABCDEFGHI......'  C
         DC    C'.JKLMNOPQR......'  D
         DC    C'..STUVWXYZ......'  E
         DC    C'Ø123456789......'  F
*
         EJECT
*********************************************************************
***                                                               ***
***      DSECT FOR MY SAVE AREA AND VARIABLES.                    ***
***                                                               ***
*********************************************************************
WORKD    DSECT
MYSAVE   DS    18F                      MY REGISTER SAVE AREA
COMPCODE DS    F                        PROGRAM COMPLETION CODE
R1SAVE   DS    F                        INITIAL VALUE IN R1
PAGES    DS    PL2
DOUBLE   DS    D
KEYLENMN DS    H
KEYLENM1 DS    H
EOFFLAGS DS    X
RETCODE  DS    X                        INTERNAL RETURN CODE
COUNTDUP DS    PL3
LINEPTR  DS    F
COLPTR   DS    H
LASTCOL  DS    H
KEYLEN   DS    2F
RECLEN   EQU   KEYLEN
VSAMACB  DS    A
VSAMRETC DS    F
DDNAME   DS    CL8
CZN      DS    CL6
OPTIONS  DS    X
LISTBIT  EQU   1
MATCHBIT EQU   2
UNIQUBIT EQU   4
HEXBIT   EQU   8
FROMBIT  EQU   16
THRUBIT  EQU   32
EXCLBIT  EQU   64
MAXKBIT  EQU   128
*
IDENT    DS    C
IDENTS   DS    PL3
COUNTUNQ DS    PL3
FROMLEN  DS    H
THRULEN  DS    H
EXCLLEN  DS    H
```

```
RANGELEN DS    H
COUNTFRM DS    PL3
COUNTTHR DS    PL3
COUNTXCL DS    PL3
COUNTMAX DS    PL3
COUNT1F  DS    PL3
COUNT1T  DS    PL3
COUNT1E  DS    PL3
COUNT1M  DS    PL3
COUNT2F  DS    PL3
COUNT2T  DS    PL3
COUNT2E  DS    PL3
COUNT2M  DS    PL3
MAXKEYL  DS    PL2
*
* BEGIN STUB LINK SAVE
*
SAVCPBAL DS    A                    BAL REGISTER SAVE AREA FOR CLRPAGE
SAVCVBAL DS    A                    BAL REGISTER SAVE AREA FOR CLOSVSAM
SAVDMBAL DS    A                    BAL REGISTER SAVE AREA FOR DOMATCH
SAVDTBAL DS    A                    BAL REGISTER SAVE AREA FOR DOTOTALS
SAVGPBAL DS    A                    BAL REGISTER SAVE AREA FOR GETPARMS
SAVJGBAL DS    A                    BAL REGISTER SAVE AREA FOR JULGREG
SAVOVBAL DS    A                    BAL REGISTER SAVE AREA FOR OPENVSAM
SAVPKBAL DS    A                    BAL REGISTER SAVE AREA FOR PUTKEY
SAVPPBAL DS    A                    BAL REGISTER SAVE AREA FOR PRTPAGE
SAVR1BAL DS    A                    BAL REGISTER SAVE AREA FOR READ1
SAVR2BAL DS    A                    BAL REGISTER SAVE AREA FOR READ2
SAVRVBAL DS    A                    BAL REGISTER SAVE AREA FOR READVSAM
SAVDLBAL DS    A                    BAL REGISTER SAVE AREA FOR DOLIST
SAVGNBAL DS    A                    BAL REGISTER SAVE AREA FOR GETNAME
SAVGRBAL DS    A                    BAL REGISTER SAVE AREA FOR GETRANGE
SAVFWBAL DS    A                    BAL REGISTER SAVE AREA FOR FINDWILD
*
* END STUB LINK SAVE
*
*
* BEGIN OPEN/CLOSE LIST
*
         DS    ØD
*
PROPENL  OPEN  (,),MF=L
PROPENLN EQU   *-PROPENL
PRCLOSL  CLOSE (),MF=L
PRCLOSLN EQU   *-PRCLOSL
*
RGOPENL  OPEN  (,),MF=L
RGOPENLN EQU   *-RGOPENL
```

```
RGCLOSL  CLOSE (),MF=L
RGCLOSLN EQU   *-RGCLOSL
*
* END OPEN/CLOSE LIST
*
*
* BEGIN DCB DSECTS
*
PRINTER  DCB   DDNAME=PRINTER,DEVD=DA,DSORG=PS,LRECL=133,             -
               BLKSIZE=133,MACRF=(PM),RECFM=FBA
PRINTERL EQU   *-PRINTER
*
         DS    ØF
IN1RPL   RPL   AM=VSAM,ACB=INPUT1,OPTCD=(KEY,SEQ,FWD,NUP,MVE),        -
               AREA=IN2LOC,AREALEN=4
LIN1RPL  EQU   *-IN1RPL
         DS    ØF
IN1RPLX  DS    CL(LRPL)
*
         DS    ØF
INPUT1   ACB   AM=VSAM,DDNAME=INPUT1,MACRF=(KEY,SEQ),EXLST=IN1EXLST
INPUT1L  EQU   *-INPUT1
*
         DS    ØF
SHOWCB1  DS    CL(LSHOWCB1)
*        SHOWCB RPL=IN1RPL,AREA=IN1RECL,LENGTH=4,FIELDS=RECLEN
*
IN1LOC   DS    A                MUST MAINTAIN ORDER AND FOLLOW SHOWCB1
IN1RECL  DS    F                 "
IN1KEYL  DS    F                 "
IN1RKP   DS    F                 "
IN1MAXRL DS    F                 "
IN1DDN   DS    CL8               "
IN1DSN   DS    CL44              "
COUNT1   DS    PL3
OLDKEY1  DS    CL256
*
         DS    ØF
IN2RPL   RPL   AM=VSAM,ACB=INPUT2,OPTCD=(KEY,SEQ,FWD,NUP,LOC),        -
               AREA=IN2LOC,AREALEN=4
         DS    ØF
IN2RPLX  DS    CL(LRPL)
*
         DS    ØF
INPUT2   ACB   AM=VSAM,DDNAME=INPUT2,MACRF=(KEY,SEQ),EXLST=IN2EXLST
INPUT2L  EQU   *-INPUT2
*
         DS    ØF
```

```
SHOWCB2  DS    CL(LSHOWCB1)
*        SHOWCB RPL=IN2RPL,AREA=IN2RECL,LENGTH=4,FIELDS=RECLEN
*
IN2LOC   DS    A               MUST MAINTAIN ORDER AND FOLLOW SHOWCB2
IN2RECL  DS    F                "
IN2KEYL  DS    F                "
IN2RKP   DS    F                "
IN2MAXRL DS    F                "
IN2DDN   DS    CL8              "
IN2DSN   DS    CL44             "
COUNT2   DS    PL3
OLDKEY2  DS    CL256
*
RANGES   DCB   DDNAME=RANGES,DSORG=PS,MACRF=GM,EODAD=RGEOF
RANGESL  EQU   *-RANGES
*
* END DCB DSECTS
*
         DS    ØF
SHOWCB3  DS    CL(LSHOWCB3)
*        SHOWCB ACB=(R2),OBJECT=DATA,FIELDS=(KEYLEN,RKP,LRECL),        -
               AREA=KEYLEN,LENGTH=12
         DS    ØF
SHOWCB4  DS    CL(LSHOWCB4)
*        SHOWCB ACB=(R2),AREA=VSAMRETC,FIELDS=(ERROR,DDNAME),          -
               LENGTH=12,OBJECT=DATA         GET CODE, DDNAME
         DS    ØF
SHOWCB5  DS    CL(LSHOWCB5)
*        SHOWCB RPL=(R2),AREA=VSAMACB,FIELDS=(ACB,FDBK),               -
               LENGTH=8,OBJECT=DATA GET ACB ADDRESS, FEEDBACK AREA
         DS    ØF
SHOWCB6  DS    CL(LSHOWCB6)
*        SHOWCB ACB=(R2),AREA=DDNAME,FIELDS=DDNAME,LENGTH=8,
          OBJECT=DATA
         DS    ØF
SHOWCB7  DS    CL(LSHOWCB4)
*        SHOWCB ACB=(R2),AREA=VSAMRETC,FIELDS=(DDNAME),                -
               LENGTH=8,OBJECT=DATA         GET CODE, DDNAME
*
*                M  A  M  J  J  A  S  O  N
JGMOTBL  DS    PL2'Ø'
JANUARY  DS    P'31'
*                M  A  M  J  J  A  S  O  N
FEBRUARY DS    P'28,31,3Ø,31,3Ø,31,31,3Ø,31,3Ø'
DECEMBER DS    P'31'
JGDAYS   DS    PL2
JGMONTHS DS    PL2
JGMMDDYY DC    C'MM/DD/YY'
```

```
JGYYDDD  DS    F
* END DSECT INSERT
*
HEADER   DS    CL133
         ORG   HEADER+L'HEADER/2-4
HEADDATE DS    CL8
         ORG   HEADER+L'HEADER-5
PAGENO   DS    CL4
         ORG
LINE     DS    CL133
ERRORMSG EQU   LINE
*
FROMKEY  DS    CL256
THRUKEY  DS    CL256
EXCLKEY  DS    CL256
TRTWORK  DS    CL256
RINAREA  DS    CL261
*
LINES    DS    (LPP)CL132,C
*
         DS    ØD
WORKDLEN EQU   *-WORKD
*
         IHAPSA                          MAP OF PSA  DSECT=PSA
         IKJTCB                          MAP OF TCB  DSECT=TCB
TIOT     DSECT
         IEFTIOT1                        MAP OF TIOT
         CVT   DSECT=YES                 MAP OF CVT  DSECT=CVTMAP
JFCB     DSECT                           MAP OF JFCB
JFCBPREF DS    CL16                        PREFIX
         IEFJFCBN LIST=NO                  JFCB PROPER
*
*        DCBD   DSORG=PO,DEVD=DA                                  A.T.
*
         EJECT
*********************************************************************
***                                                               ***
***      REGISTER EQUATES                                         ***
***                                                               ***
*********************************************************************
*
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
*
         END
```

Keith Nicaise
(USA)

Updating VSAM definitions in the CSD (continued)

Here, we conclude the article on updating VSAM definitions in the CSD, which was begun in the last issue. Published below are the source code for DYNALLOC and sample JCL.

PROGRAM DYNALLOC SOURCE CODE

```
* PROGRAM: DYNALLOC
* MACRO DEFINITION
         MACRO                     MACRO HEADER
         CSSET                     PROTOTYPE STATEMENT
PGMNAME  DC    CL8'&SYSECT'        PROGRAM NAME
         MEND                      MACRO END
DYNALLOC CSECT
* BATCH PROGRAM FOR DYNAMIC ALLOCATION/DEALLOCATION
*-------------------------------------------------------------
* CALL Sample:                                                *
*                  CALL DYNALLOC,(AS)                         *
*              AFTER CLOSE DS IS DEALLOCATED                  *
*-------------------------------------------------------------
RWKR1    EQU   R1
RWKR2    EQU   R2
RWKR3    EQU   R3
RWKR14   EQU   R14
RWKR15   EQU   R15
RBAL2    EQU   R2
RAS      EQU   R8
RBASE1   EQU   R4
RBASE2   EQU   R5
```

```
RBASE3   EQU   R6
* PROGRAM IDENTIFIER
         B     28(Ø,R15)          BRANCH AROUND CONSTANTS
         DC    CL8'DYNALLOC'      PROGRAM NAME
         DC    CL8'&SYSDATE'      TODAY'S DATE
         DC    CL8'&SYSTIME'      TIME OF COMPILE
         STM   R14,R12,12(R13)
         LR    RBASE1,R15
         USING DYNALLOC,RBASE1,RBASE2,RBASE3
         LA    RBASE2,2Ø48(RBASE1)
         LA    RBASE2,2Ø48(RBASE2)
         LA    RBASE3,2Ø48(RBASE2)
         LA    RBASE3,2Ø48(RBASE3)
         LA    RØ,SAVEAREA          ADDRESS OF SAVEAREA
         ST    R13,SAVEAREA+4       INVOKERS SAVE AREA ADDR IN
*                                   MY SAVE AREA
         ST    RØ,8(R13)            MY SAVE AREA IN INVOKERS
*                                   SAVE AREA
         LR    R13,RØ               LOAD R13 WITH MY SAVE AREA
*                                   ADDRESS
         ICM   RAS,B'1111',Ø(RWKR1) CALLING AREA ADDRESS
         BZ    RETURN               ..NO CALLING AREA
         MVI   SWTO,X'ØØ'           INITIALIZE SWITCHES
         MVI   SFUNC,X'ØØ'          WITH ZERO
         USING AS,RAS               ADDRESSABILITY COMMON DATA AREA
         CLI   RES,C'X'             NO WTO AND DEALLOCATION
         BE    NWTO                 ... BRANCH IF YES
         CLI   RES,C'N'             NO WTO FOR SVC 99  ERROR
         BNE   YWTO                 ... BRANCH IF NO
NWTO     DS    ØH
         MVI   SWTO,X'FF'           SET NO WTO
YWTO     DS    ØH
         CLI   RES,C'X'             DEALLOCATION WITHOUT WTO ?
         BE    SETDEAL              ... YES
         CLI   RES,C'Y'             DEALLOCATION WITH WTO ?
         BNE   SETALL               ... NO
SETDEAL  DS    ØH
         MVI   SFUNC,X'Ø1'          SET DEALLOCATION
SETALL   DS    ØH
         MVI   RES,X'Ø'             SET RC = OK
* RESET BUILDING TABLE
         CLI   SFUNC,X'Ø1'          DEALLOCATION REQUIRED ?
         BNE   NODEAL               ... NO
         MVI   REQTIP,X'Ø2'         SET VERB FOR DEALLOCATION
         B     ASFUNC
NODEAL   DS    ØH
         MVI   REQTIP,X'Ø1'         SET VERB FOR ALLOCATION
ASFUNC   DS    ØH
         LA    RWKR1,TEXTPTR        LOAD TEXT UNIT TABLE ADDRESS
LOOPZ    DS    ØH
```

```
         CLI   Ø(RWKR1),X'FF'       END OF TABLE ?
         BE    FLOOPZ               ... YES
         XC    Ø(L'TEXTPTR,RWKR1),Ø(RWKR1) SET ELEMENT TO ZERO
         LA    RWKR1,L'TEXTPTR(RWKR1) NEXT ELEMENT
         B     LOOPZ                CONTINUE LOOP
FLOOPZ   DS    ØH                   END OF TABLE
         LA    RWKR2,TEXTPTRD       LOAD TEXT UNIT TABLE FOR DEALLOC.
LOOPZD   DS    ØH
         CLI   Ø(RWKR1),X'FF'       END OF TABLE ?
         BE    FLOOPZD              ... YES
         XC    Ø(L'TEXTPTRD,RWKR1),Ø(RWKR1) SET ELEMENT TO ZERO
         LA    RWKR1,L'TEXTPTRD(RWKR1) NEXT ELEMENT
         B     LOOPZD               CONTINUE LOOP
FLOOPZD  DS    ØH                   END OF TABLE
* RESET OPTIONAL TABLES
         LA    RWKR1,TEXUNITO       LOAD TEXT UNIT INDEX OPTIONAL
*                                   ELEMENTS ADDRESS
LOOPZ1   DS    ØH
         CLI   Ø(RWKR1),X'FF'       END OF TABLE ?
         BE    FLOOPZ1              ... YES
         L     RWKR2,Ø(RWKR1)       LOAD ADDRESS
         SH    RWKR2,=H'1'          POINT TO ACTIVE BYTE
         MVI   Ø(RWKR2),X'Ø'        SET TO NOT ACTIVE
         LA    RWKR1,L'TEXUNITO(RWKR1) NEXT ELEMENT
         B     LOOPZ1               CONTINUE LOOP
FLOOPZ1  DS    ØH                   END OF TABLE
         MVC   TXTPP,=A(TEXTPTR)    SET TEXT POINTER FOR ALLOCATION
         CLI   SFUNC,X'Ø1'          DEALLOCATION REQUIRED ?
         BNE   FLOOPZ2              ... NO
         MVI   TEXTUNL-1,X'FF'      FOR DEALLOCATION REQUIRED ENTRY
         MVC   TXTPP,=A(TEXTPTRD)   SET TEXT POINTER FOR DEALLOCATION
FLOOPZ2  DS    ØH
* NORMAL DISPOSITION
         LA    RWKR1,TABDISP        DISPOSITION TABLE
LOOPDISP DS    ØH
         CLC   ASDISP,Ø(RWKR1)      SET DISPOSITION AS REQUIRED
         BE    SETDISP              ... OK FOUND
         LA    RWKR1,L'TABDISP(RWKR1) NEXT ELEMENT
         CLI   Ø(RWKR1),X'FF'       END OF TABLE ?
         BNE   LOOPDISP             ... NO CONTINUE LOOP
         SH    RWKR1,=Y(L'TABDISP)  POINT TO LAST ELEMENT (DEFAULT)
SETDISP  DS    ØH                   SET DISPOSITION
         MVC   DISP,L'TABDISP-1(RWKR1)
         MVC   MSSTS(L'TABDISP-1),Ø(RWKR1)
* CONDITIONAL DISPOSITION
         LA    RWKR1,TABDISP        CONDITIONAL DISPOSITION TABLE
LOOPDISC DS    ØH
         CLC   ASDISPC,Ø(RWKR1)     SET DISPOSITION AS REQUIRED
         BE    SETDISPC             ... OK FOUND
         LA    RWKR1,L'TABDISP(RWKR1) NEXT ELEMENT
```

```
         CLI   Ø(RWKR1),X'FF'       END OF TABLE ?
         BNE   LOOPDISC             ... NO CONTINUE LOOP
         SH    RWKR1,=Y(L'TABDISP)  POINT TO LAST ELEMENT (DEFAULT)
SETDISPC DS    ØH                   SET CONDITIONAL DISPOSITION
         MVC   DISPC,L'TABDISP-1(RWKR1)
         MVC   MSSTSC(L'TABDISP-1),Ø(RWKR1)
* STATUS
         LA    RWKR1,TABSTS         STATUS TABLE
LOOPSTS  DS    ØH
         CLC   ASSTS,Ø(RWKR1)       SET STATUS AS REQUIRED
         BE    SETSTS               ... OK FOUND
         LA    RWKR1,L'TABSTS(RWKR1) NEXT ELEMENT
         CLI   Ø(RWKR1),X'FF'       END OF TABLE
         BNE   LOOPSTS              ...NO CONTINUE LOOP
         SH    RWKR1,=Y(L'TABSTS)   POINT TO LAST ELEMENT (DEFAULT)
SETSTS   DS    ØH STATUS            SET STATUS
         MVC   STATUS,L'TABSTS-1(RWKR1)
         MVC   MSDISP(L'TABSTS-1),Ø(RWKR1)
*        OPTIONAL ENTRIES
* LABEL
         CLC   ASLABEL,BLANK        LABEL SPECIFIED ?
         BE    NOSETLBL             ... NO
         CLC   ASLABEL,ZERIEX       LABEL SPECIFIED ?
         BE    NOSETLBL             ...NO
         MVI   TABLBL-1,X'FF'       ACTIVE ENTRY
         LA    RWKR1,TABLBL         LABEL TABLE
LOOPLBL  DS    ØH
         CLC   ASLABEL,Ø(RWKR1)     SET LABEL AS REQUIRED
         BE    SETLBL               ... OK FOUND
         LA    RWKR1,L'TABLBL(RWKR1) NEXT ENTRY
         CLI   Ø(RWKR1),X'FF'       END OF TABLE ?
         BNE   LOOPLBL              ... NO CONTINUE LOOP
         SH    RWKR1,=Y(L'TABLBL)   POINT TO LAST ELEMENT (DEFAULT)
SETLBL   DS    ØH LABEL
         MVC   LABEL,L'TABLBL-1(RWKR1)
         MVC   MSLBL(L'TABLBL-1),Ø(RWKR1)
NOSETLBL DS    ØH
* UNIT
         CLC   ASUNIT,BLANK         UNIT SPECIFIED ?
         BE    NOSETU               ... NO
         CLC   ASUNIT,ZERIEX        UNIT SPECIFIED ?
         BE    NOSETU               ... NO
         CLI   ASUNIT+L'ASUNIT-1,C' ' 4 POSITIONS ?
         BE    LUNIT4               ... YES
         CLI   ASUNIT+L'ASUNIT-1,X'Ø' 4 POSITIONS ?
         BE    LUNIT4               ... YES
         MVC   UNITL,=Y(L'ASUNIT)     5 POSITIONS LENGTH
         B     ALUNIT4
LUNIT4   DS    ØH
         MVC   UNITL,=Y(L'ASUNIT-1)   4 POSITIONS LENGTH
```

```
ALUNIT4  DS    ØH
         MVC   UNIT,ASUNIT            SET UNIT AS REQUIRED
         MVC   MSUNIT(L'ASUNIT),ASUNIT
         MVI   TEXTUNIT-1,X'FF'       ACTIVE ENTRY
NOSETU   DS    ØH
* VOLSER
         CLC   ASVOLSER,BLANK         VOLSER SPECIFIED ?
         BE    NOSETVOL               ... NO
         CLC   ASVOLSER,ZERIEX        VOLSER SPECIFIED ?
         BE    NOSETVOL               ... NO
         MVC   VOLSER,ASVOLSER        SET VOLSER AS REQUIRED
         MVC   MSVOL(L'ASVOLSER),ASVOLSER
         MVI   TEXTVOL-1,X'FF'        ACTIVE ENTRY
NOSETVOL DS    ØH
* SPACE
         CLI   ASSPACET,C' '          SPACE SPECIFIED ?
         BE    NOSPACE                ... NO
         CLI   ASSPACET,C'T'          SPACE IN TRACKS ?
         BE    FORTRK                 ... YES
         MVI   TEXTCYL-1,X'FF'        SPACE IS IN CYLINDERS
         B     SISPACE
FORTRK   DS    ØH
         MVI   TEXTTRK-1,X'FF'        SPACE IS IN TRACKS
SISPACE  DS    ØH
         MVI   TEXTSPP-1,X'FF'        ACTIVE ENTRY
         MVC   SPACEP,=FL3'2Ø' DEFAULT 2Ø SED DEFULT SPACE
         TRT   ASSPACEP,TABTRT        QUANTITY SPECIFIED IS NUMERIC ?
         BNZ   SISPACES               ... NO - ASSUME NO SPACE
         CLC   ASSPACEP,ZERIZ         QUANTITY EQUAL TO ZERO ?
         BE    SISPACES               ... YES (NO SPACE)
         PACK  DOUBLE,ASSPACEP        PREPARE SPACE QUANTITY
         CVB   RWKR1,DOUBLE
         STCM  RWKR1,B'Ø111',SPACEP
SISPACES DS    ØH
         CLC   ASSPACES,BLANK         SECONDARY SPACE SPECIFIED ?
         BE    NOSPACE                ... NO
         CLC   ASSPACES,ZERIEX        SECONDARY SPACE SPECIFIED ?
         BE    NOSPACE                ... NO
         MVI   TEXTSPS-1,X'FF'        ACTIVE ENTRY
         MVC   SPACES,=FL3'1Ø'        SET DEFAULT TO 1Ø
         TRT   ASSPACES,TABTRT        QUANTITY SPECIFIED IS NUMERIC ?
         BNZ   NOSPACE                NO - ASSUME NO SPACE
         CLC   ASSPACES,ZERIZ         QUANTITY EQUAL TO ZERO ?
         BE    NOSPACE                YES (NO SPACE)
         PACK  DOUBLE,ASSPACES        PREPARE SPACE QUANTITY
         CVB   RWKR1,DOUBLE
         STCM  RWKR1,B'Ø111',SPACES
NOSPACE  DS    ØH
* NUMBER OF FILE IN TAPE
         CLC   ASTAPES,BLANK          SPECIFIED ?
```

```
         BE    NOTAPES                 ... NO
         CLC   ASTAPES,ZERIEX          SPECIFIED ?
         BE    NOTAPES                 ... NO
         CLC   ASTAPES,ZERIZ           NUMBER EQUAL TO ZERO
         BE    NOTAPES                 ... YES
         MVI   TEXTDSS-1,X'FF'         ACTIVE ENTRY
         MVC   TAPESEQ,=XL2'0001'      SET DEFAULT
         TRT   ASTAPES,TABTRT          NUMERIC VALUE ?
         BNZ   NOTAPES                 ... NO
         MVC   MSTAPES,ASTAPES         PREPARE NUMBER
         PACK  DOUBLE,ASTAPES
         CVB   RWKR1,DOUBLE
         STCM  RWKR1,B'0011',TAPESEQ
NOTAPES  DS    0H
* REFERRING DCB
         CLC   ASDCBR,BLANK            SPECIFIED ?
         BE    NODCBR                  ... NO
         CLC   ASDCBR,ZERIEX           SPECIFIED ?
         BE    NODCBR                  ... NO
         LA    RWKR1,ASDCBR+L'ASDCBR-1 LAST BYTE
         LA    RWKR2,L'ASDCBR
LOOPDCB  DS    0H                      COMPUTE LENGTH
         CLI   0(RWKR1),C' '
         BE    DCBN
         CLI   0(RWKR1),X'0'
         BNE   EXLDCB
DCBN     DS    0H
         SH    RWKR1,=H'1'
         BCT   RWKR2,LOOPDCB
EXLDCB   DS    0H
         STCM  RWKR2,B'0011',LDCBDD    SET LENGTH OF DDNAME
         SH    RWKR2,=H'1'             -1 FOR EXECUTE
         BM    NODCBR                  NEGATIVE ????
         EX    RWKR2,MOVEDD
         MVI   TEXTDCB-1,X'FF'         ACTIVE ENTRY
         MVC   MSDCBR,ASDCBR
         B     NODCBR
MOVEDD   MVC   DCBDD(0),ASDCBR
NODCBR   DS    0H
* DSORG
         CLC   ASDSORG,BLANK           SPECIFIED ?
         BE    NODSORG                 ... NO
         CLC   ASDSORG,ZERIEX          SPECIFIED ?
         BE    NODSORG                 ... NO
         LA    RWKR1,TABDSRG           DSORG TABLE ADDRESS
LOOPDSRG DS    0H
         CLI   0(RWKR1),X'FF'          END OF TABLE ?
         BE    NODSORG                 ... YES INVALID DSORG SPECIFIED
         CLC   ASDSORG,2(RWKR1)        CHECK IF DSORG VALID
         BE    SETDSORG
```

```
         LA    RWKR1,L'TABDSRG(RWKR1) NEXT ELEMENT
         B     LOOPDSRG               CONTINUE LOOP
SETDSORG DS    ØH
         MVC   DSORG,Ø(RWKR1)         SET DSORG
         MVC   MDSORG(L'ASDSORG),ASDSORG
         MVI   TEXTDSRG-1,X'FF'        ACTIVE ENTRY
NODSORG  DS    ØH
* MEMBER OF PO
         CLC   =CL2'PO',ASDSORG       PARTITIONED DATASET ?
         BNE   NOMBR                  ... NO
         CLC   ASMEMBER,BLANK         MEMBER SPECIFIED ?
         BE    NOMBR                  ... NO
         CLC   ASMEMBER,ZERIEX        MEMBER SPECIFIED ?
         BE    NOMBR                  ... NO
* COMPUTE MEMBER NAME LENGTH
         LA    RWKR1,ASMEMBER+L'ASMEMBER-1 LAST BYTE
         LA    RWKR2,L'ASMEMBER
LOOPMBR  DS    ØH
         CLI   Ø(RWKR1),C' '
         BE    MBRN
         CLI   Ø(RWKR1),X'Ø'
         BNE   EXLMBR
MBRN     DS    ØH
         SH    RWKR1,=H'1'
         BCT   RWKR2,LOOPMBR
EXLMBR   DS    ØH
         STCM  RWKR2,B'ØØ11',LMEMBER  SET MEMBER NAME LENGTH
         SH    RWKR2,=H'1'            -1 FOR EXECUTE
         BM    NOMBR                  ??? NEGATIVE
         EX    RWKR2,MOVEMBR
         MVI   TEXTMBR-1,X'FF'        ACTIVE ENTRY
         B     NOMBR
MOVEMBR  MVC   MEMBER(Ø),ASMEMBER
NOMBR    DS    ØH
* RECFM
         MVI   RECFM,X'Ø'
         CLC   ASRECFM,BLANK          SPECIFIED ?
         BE    NORECFM                ... NO
         CLC   ASRECFM,ZERIEX         SPECIFIED ?
         BE    NORECFM                ... NO
         LA    RWKR1,TABRECFM         LOAD RECFM TABLE ADDRESS
         LA    RWKR2,ASRECFM
         LA    RWKR15,ASRECFM+L'ASRECFM-1
LOOPRECF DS    ØH
         CR    RWKR2,RWKR15
         BH    FINERECF
         CLI   Ø(RWKR2),C' '
         BE    FINERECF
         CLI   Ø(RWKR1),X'FF'         END OF TABLE ?
         BE    RESETFM                ... YES
```

```
         CLC   Ø(1,RWKR2),1(RWKR1)
         BE    SETRECFM
         LA    RWKR1,L'TABRECFM(RWKR1)
         B     LOOPRECF
RESETFM  DS    ØH
         LA    RWKR1,TABRECFM
         LA    RWKR2,1(RWKR2)
         B     LOOPRECF
SETRECFM DS    ØH
         OC    RECFM,Ø(RWKR1)
         B     RESETFM
FINERECF DS    ØH
         MVC   MRECFM(L'ASRECFM),ASRECFM
         MVI   TEXTRECF-1,X'FF'       ACTIVE ENTRY
NORECFM  DS    ØH
* BLKSIZE
         CLC   ASBLKSIZ,BLANK         SPECIFIED ?
         BE    NOBLKZ                 ... NO
         CLC   ASBLKSIZ,ZERIEX        SPECIFIED ?
         BE    NOBLKZ                 ... NO
         TRT   ASBLKSIZ,TABTRT        NUMERIC ?
         BNZ   NOBLKZ                 ... NO
         CLC   ASBLKSIZ,ZERIZ         VALUE IS ZERO ?
         BE    NOBLKZ                 ... YES
         PACK  DOUBLE,ASBLKSIZ        PREPARE BLOCKSIZE
         CVB   RWKR1,DOUBLE
         STCM  RWKR1,B'ØØ11',BLKSIZE
         MVC   MBLKSIZ(L'ASBLKSIZ),ASBLKSIZ
         MVI   TEXTBLK-1,X'FF'        ACTIVE ENTRY
NOBLKZ   DS    ØH
* LRECL
         CLC   ASLRECL,BLANK          SPECIFIED ?
         BE    NOLRECL                ... NO
         CLC   ASLRECL,ZERIEX         SPECIFIED ?
         BE    NOLRECL                ... NO
         TRT   ASLRECL,TABTRT         NUMERIC ?
         BNZ   NOLRECL                ... NO
         CLC   ASLRECL,ZERIZ          VALUE IS ZERO ?
         BE    NOLRECL                ... YES
         PACK  DOUBLE,ASLRECL         PREPARE LRECL
         CVB   RWKR1,DOUBLE
         STCM  RWKR1,B'ØØ11',LRECL
         MVC   MLRECL(L'ASLRECL),ASLRECL
         MVI   TEXTLRCL-1,X'FF'       ACTIVE ENTRY
NOLRECL  DS    ØH
* BUFNO
         CLC   ASBUFNO,BLANK          SPECIFIED ?
         BE    NOBUFNO                ... NO
         CLC   ASBUFNO,ZERIEX         SPECIFIED ?
         BE    NOBUFNO                ... NO
```

```
         TRT   ASBUFNO,TABTRT          NUMERIC ?
         BNZ   NOBUFNO                 ... NO
         CLC   ASBUFNO,ZERIZ           VALUE IS ZERO ?
         BE    NOBUFNO                 ... YES
         PACK  DOUBLE,ASBUFNO          PREPARE BUFFER NUMBER
         CVB   RWKR1,DOUBLE
         STCM  RWKR1,B'ØØ11',BUFNO
         MVC   MBUFNO(L'ASBUFNO),ASBUFNO
         MVI   TEXTBFNO-1,X'FF'
NOBUFNO  DS    ØH
* OPTCD
         MVI   OPTCD,X'Ø'
         CLC   ASOPTCD,BLANK          OPTION CODE SPECIFIED ?
         BE    NOOPTCD                ... NO
         CLC   ASOPTCD,ZERIEX         SPECIFIED ?
         BE    NOOPTCD                ... NO
         LA    RWKR1,TABOPTCD         LOAD OPTION CODE TABLE
         LA    RWKR2,ASOPTCD
         LA    RWKR15,ASOPTCD+L'ASOPTCD-1
LOOPOPTC DS    ØH
         CR    RWKR2,RWKR15
         BH    FINEOPTC
         CLI   Ø(RWKR2),C' '
         BE    FINEOPTC
         CLI   Ø(RWKR1),X'FF'         END OF TABLE ?
         BE    RESETOP                YES
         CLC   Ø(1,RWKR2),1(RWKR1)
         BE    SETOPTCD
         LA    RWKR1,L'TABOPTCD(RWKR1)
         B     LOOPOPTC
RESETOP  DS    ØH
         LA    RWKR1,TABOPTCD
         LA    RWKR2,1(RWKR2)
         B     LOOPOPTC
SETOPTCD DS    ØH
         OC    OPTCD,Ø(RWKR1)
         B     RESETOP
FINEOPTC DS    ØH
         MVC   MOPTCD(L'ASOPTCD),ASOPTCD
         MVI   TEXTOPTC-1,X'FF'       ACTIVE ENTRY
NOOPTCD  DS    ØH
* DSNAME
         MVC   DSNAME,FILENAME        SET DATASET NAME
         MVC   MSDSN(L'FILENAME),FILENAME
* DDNAME
         MVC   DDNAME,JCLNAME         SET DDNAME
         MVC   MSDDN(L'JCLNAME),JCLNAME
* BUILT TEXT UNIT TABLE FOR SVC 99
         CLI   SFUNC,X'Ø1'            DEALLOCATION REQUIRED ?
         BNE   ALSET                  ... NO
```

```
* DEALLOCATION
         LA    RWKR1,TEXUNITD        SET TABLE
         LA    RWKR2,TEXTPTRD
         B     LOOPTT
* ALLOCATION
ALSET    DS    ØH
         LA    RWKR1,TEXUNITT        SET TABLE
         LA    RWKR2,TEXTPTR
LOOPTT   DS    ØH                    MOVE ACTIVE ENTRIES ADDRESS
         CLI   Ø(RWKR1),X'FF'
         BE    FLOOPT
         L     RWKR15,Ø(RWKR1)
         SH    RWKR15,=H'1'
         CLI   Ø(RWKR15),X'FF'
         BE    MOVET
         LA    RWKR1,L'TEXUNITT(RWKR1)
         B     LOOPTT
MOVET    DS    ØH
         MVC   Ø(L'TEXTPTR,RWKR2),Ø(RWKR1)
         LA    RWKR1,L'TEXUNITT(RWKR1)
         LA    RWKR2,L'TEXTPTR(RWKR2)
         B     LOOPTT
FLOOPT   DS    ØH
         SH    RWKR2,=Y(L'TEXTPTR)
         OI    Ø(RWKR2),X'8Ø'        END OF LIST
         LA    RWKR1,REQBPTR
         SVC   99
         LTR   RWKR15,RWKR15
         BNZ   FAIL
         CLC   REQERR,ZERIEX
         BE    RETURN
         MVI   RES,X'FF'
         B     AFAIL
FAIL     DS    ØH
         STC   RWKR15,RES            SET ERROR RESPONSE
AFAIL    DS    ØH
         ST    RWKR15,DOUBLE
         UNPK  MSER+31(9),DOUBLE(L'DOUBLE+1)
         TR    MSER+31(9),TABEX-24Ø
         MVI   MSER+31+8,C' '
         UNPK  MSER+52(5),REQINF(L'REQINF+1)
         TR    MSER+52(5),TABEX-24Ø
         MVI   MSER+52+4,C' '
         UNPK  MSER+76(5),REQERR(L'REQERR+1)
         TR    MSER+76(5),TABEX-24Ø
         MVI   MSER+76+4,C' '
* SVC 99 ERROR
         CLI   SWTO,X'FF'
         BE    RETURN
* NOTIFY TO OPERATOR
```

```
         MVC   WTOMSG(L'MSER),MSER
         BAL   RBAL2,WTO
         MVC   WTOMSG(L'MSER1),MSER1
         BAL   RBAL2,WTO
         MVC   WTOMSG(L'MSER2),MSER2
         BAL   RBAL2,WTO
         MVC   WTOMSG(L'MSER3),MSER3
         BAL   RBAL2,WTO
         MVC   WTOMSG(L'MSER4),MSER4
         BAL   RBAL2,WTO
         B     RETURN
RETURN   DS    ØH
         L     R13,4(R13)
         LM    R14,R12,12(R13)
         BR    R14
WTO      DS    ØH
         ST    RBAL2,VOXBAL2
         MVC   WTOHD1(L'PGMNAME),PGMNAME  INITIALIZE WITH PROGRAM NAME
         LA    R1,WTOBLK
         SVC   35
         MVC   WTOMSG,BLANK
         L     RBAL2,VOXBAL2
         BR    RBAL2
VOXBAL2  DS    A
SAVEAREA DS    ØD
WORD1    DC    A(Ø)  PL/I ONLY
WORD2    DC    A(Ø)  ADDRESS OF THE CALLER'S SAVE AREA
WORD3    DC    A(Ø)  ADDRESS OF THE SAVEAREA OF THE CALLED PROGRAM
WORD4    DC    A(Ø)  REGISTER 14 RETURN ADDRESS WITHIN THE
*                     CALLING PHASE
WORD5    DC    A(Ø)  REGISTER 15 ENTRY POINT ADDRESS OF THE
*                     CALLED PHASE
WORD6    DC    A(Ø)  REGISTER Ø
WORD7    DC    A(Ø)  REGISTER 1
WORD8    DC    A(Ø)  REGISTER 2
WORD9    DC    A(Ø)  REGISTER 3
WORD1Ø   DC    A(Ø)  REGISTER 4
WORD11   DC    A(Ø)  REGISTER 5
WORD12   DC    A(Ø)  REGISTER 6
WORD13   DC    A(Ø)  REGISTER 7
WORD14   DC    A(Ø)  REGISTER 8
WORD15   DC    A(Ø)  REGISTER 9
WORD16   DC    A(Ø)  REGISTER 1Ø
WORD17   DC    A(Ø)  REGISTER 11
WORD18   DC    A(Ø)  REGISTER 12
* WTO CONTROL BLOCK
         CNOP  Ø,4
WTOBLK   DS    ØH
         DC    Y(WTOBLKE-WTOBLK)
         DC    B'ØØØØØØØØØØØØØØØØ' MCSFLAGS
```

```
WTOHD1   DS    ØCL116
         DC    CL8' ',CL1'-'
WTOMSG   DC    CL1Ø7' '
WTOBLKE  EQU   *
         LTORG
TABEX    DC    256X'Ø'
         ORG   TABEX+X'FØ'
         DC    C'Ø123456789ABCDEF'
         ORG
* WTO INDICATOR
SWTO     DC    X'Ø' X'ØØ'=WTO X'FF'=NO WTO
* FUNCTION INDICATOR
SFUNC    DC    X'Ø' X'ØØ'=ALLOCATION X'Ø1'=DEALLOCATION
* ERROR MESSAGE
MSER     DS    ØCL8Ø
         DC    CL8Ø'ALLOCATION ERROR SVC 99 R15/RC=XXXXXXXX REASON CODE/
               =XXXX ERROR REASON CODE =XXXX',CL1' '
MSER1    DS    ØCL69
         DC    CLØ8'DDNAME ='
MSDDN    DC    8C'X',CL1' '
         DC    CLØ8'DSNAME ='
MSDSN    DC    44C'X'
MSER2    DS    ØCL8Ø
         DC    CLØ6'DISP ='
MSDISP   DC    3C'X',CL1','
MSSTS    DC    7C'X',CL1','
MSSTSC   DC    7C'X',CL1' '
         DC    CLØ7'LABEL ='
MSTAPES  DC    4C'X',CL1','
MSLBL    DC    3C'X',CL1' '
         DC    CLØ6'UNIT ='
MSUNIT   DC    4C'X',CL1' '
         DC    CLØ8'VOLSER ='
MSVOL    DC    6C'X',CL1' '
         DC    CLØ7'DCB R ='
MSDCBR   DC    8C'X'
MSER3    DS    ØCL73
         DC    CLØ5'MBR ='
MMEMBER  DC    8C'X',CL1' '
         DC    CLØ7'DSORG ='
MDSORG   DC    3C'X',CL1' '
         DC    CLØ7'RECFM ='
MRECFM   DC    3C'X',CL1' '
         DC    CLØ7'BLKSZ ='
MBLKSIZ  DC    5C'X',CL1' '
         DC    CLØ7'LRECL ='
MLRECL   DC    5C'X',CL1' '
         DC    CLØ7'BUFNO ='
MBUFNO   DC    3C'X',CL1' '
MSER4    DS    ØCL11
```

```
         DC    CLØ7'OPTCD ='
MOPTCD   DC    3C'X',CL1' '
DOUBLE   DC    D'Ø' WORK AREA
         TITLE 'PARAMETER LIST FOR SVC 99'
         DS    ØA
REQBPTR  DC    XL1'8Ø',AL3(REQBLK) REQUEST BLOCK POINTER
         DS    ØA
REQBLK   DS    ØXL24
REQLEN   DC    HL1'2Ø'          LENGTH REQUEST BLOCK
REQTIP   DC    XL1'Ø1'          VERB CODE: Ø1 = ALLOCATION
*                                        Ø2 = DEALLOCATION
         DC    XL1'Ø'           FLAG11
         DC    XL1'Ø'           FLAG12
REQERR   DC    XL2'Ø'           ERROR CODE
REQINF   DC    XL2'Ø'           INFO  CODE
TXTPP    DC    A(TEXTPTR)       TEXT POINTER
TXTPPE   DC    A(TEXTRBEI)      REQUEST BLOCK EXTENSION POINTER
         DC    XL4'Ø'           FLAGS
* TEXT UNIT TABLE  FOR DEALLOCATION
TEXTPTRD DS    ØA
         DC    A(Ø)             DSNAME
         DC    A(Ø)             UNALLOC OPTION
         DC    X'FF'
* TEXT UNIT TABLE
TEXTPTR  DS    ØA
         DC    A(Ø)             DSNAME
         DC    A(Ø)             STATUS
         DC    A(Ø)             DDNAME
         DC    A(Ø)             NORMAL DISPOSITION
         DC    A(Ø)             CONDITIONAL DISPOSITION
         DC    A(Ø)             LABEL
         DC    A(Ø)             UNIT
         DC    A(Ø)             VOLSER
         DC    A(Ø)             TRACKS
         DC    A(Ø)             CYLINDERS
         DC    A(Ø)             PRIMARY SPACE
         DC    A(Ø)             SECONDARY SPACE
         DC    A(Ø)             UNALLOCATION AT CLOSE
         DC    A(Ø)             MEMBER SELECTION
         DC    A(Ø)             TAPE DATASET SEQUENCE
         DC    A(Ø)             REFERENCE TO DDNAME SPECIFICATION
         DC    A(Ø)             BLOCKSIZE
         DC    A(Ø)             BUFNO
         DC    A(Ø)             DSORG
         DC    A(Ø)             LRECL
         DC    A(Ø)             OPTCD
         DC    A(Ø)             RECFM
         DC    X'FF'
* DSECT S99RBX
TEXTRBEI DS    ØA          REQUEST BLOCK EXTENSION
```

```
         DC    CL6'S99RBX' CONTROL BLOCK IDENTIFIER
         DC    XL1'Ø1'     VERSION NUMBER
* ERROR MSG TO CONSOLE
         DC    XL1'84'     FLAGS FOR MSG PROCESSING OPTIONS
         DC    XL1'Ø'      SUBPOOL FOR MESSAGE BLOCK
         DC    XL1'Ø'      STORAGE KEY FOR MESSAGE BLOCK
         DC    XL1'Ø'      SEVERITY LEVEL FOR MESSAGE PROCESSING
* S99ENMSG
         DC    XL1'Ø'      NUMBER OF MSG BLOCK RETURNED
         DC    A(Ø)        CPPL ADDRESS
         DC    XL4'Ø'      MSG SERVICE RETURN CODE
* S99EWRC
         DC    XL4'Ø'      WTO RETURN CODE
* S99EMSGP
         DC    A(Ø)        MSG BLOCK POINTER
         DC    XL4'Ø'      INFORMATION RETRIEVAL R.C.
         DC    XL4'Ø'      SMS ERROR REASON CODE
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'FF'         FF = ACTIVE
TEXTDSN  DS    ØXL5Ø
         DC    XL2'ØØØ2'       DALDSNAM
         DC    XL2'ØØØ1'
         DC    XL2'ØØ2C'       LENGTH (44)
DSNAME   DC    CL44' '         DSNAME
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'FF'         FF = ACTIVE
TEXTSTS  DS    ØXL7
         DC    XL2'ØØØ4'       DALSTATS
         DC    XL2'ØØØ1'
         DC    XL2'ØØØ1'       LENGTH ( 1)
STATUS   DC    XL1'Ø8'         Ø8 = SHR
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'FF'         FF = ACTIVE
TEXTDDN  DS    ØXL14
         DC    XL2'ØØØ1'       DALDDNAM
         DC    XL2'ØØØ1'
         DC    XL2'ØØØ8'       LENGTH ( 8)
DDNAME   DC    CLØ8' '
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'FF'         FF = ACTIVE
TEXTDISP DS    ØXL7
         DC    XL2'ØØØ5'       DALNDISP
         DC    XL2'ØØØ1'
         DC    XL2'ØØØ1'       LENGTH ( 8)
DISP     DC    XL1'Ø8'         Ø8 = KEEP
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'FF'         FF = ACTIVE
TEXCDISP DS    ØXL7
         DC    XL2'ØØØ6'       DALCDISP
         DC    XL2'ØØØ1'
```

```
         DC    XL2'0001'       LENGTH ( 8)
DISPC    DC    XL1'08'         08 = KEEP
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'FF'         FF = ACTIVE
TEXTUC   DS    0XL7
         DC    XL2'001C'       DALCLOSE
         DC    XL2'0000'
         DC    XL2'0'          LENGTH (--)
         DC    XL1'0'
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'0'          FF = ACTIVE
TEXLABEL DS    0XL7
         DC    XL2'001E'       DALLABEL
         DC    XL2'0001'
         DC    XL2'0001'       LENGTH (--)
LABEL    DC    XL1'02'         SL
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'0'          FF = ACTIVE
TEXTUNIT DS    0XL11
         DC    XL2'0015'       DALUNIT
         DC    XL2'0001'
UNITL    DC    XL2'0005'       LENGTH (--)
UNIT     DC    XL5'0'          3480 - 3380 - SYSDA
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'0'          FF = ACTIVE
TEXTVOL  DS    0XL12
         DC    XL2'0010'       DALVLSER
         DC    XL2'0001'       NUMBER OF VOLSER
         DC    XL2'0006'       LENGTH (--)
VOLSER   DC    XL6'0'          SERIAL NUMBER
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'0'          FF = ACTIVE
TEXTTRK  DS    0XL4
TEXTUNL  DS    0XL4
         DC    XL2'0007'       DALTRK
         DC    XL2'0000'
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'0'          FF = ACTIVE
TEXTCYL  DS    0XL4
         DC    XL2'0008'       DALCYL
         DC    XL2'0000'
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'0'          FF = ACTIVE
TEXTSPP  DS    0XL9
         DC    XL2'000A'       DALPRIME
         DC    XL2'0001'
         DC    XL2'0003'       LENGTH (--)
SPACEP   DC    XL3'0'          PRIMARY SPACE
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'0'          FF = ACTIVE
```

```
TEXTSPS  DS    ØXL9
         DC    XL2'ØØØB'       DALSECND
         DC    XL2'ØØØ1'
         DC    XL2'ØØØ3'       LENGTH (--)
SPACES   DC    XL3'Ø'          SECONDARY SPACE
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'Ø'          FF = ACTIVE
TEXTDSS  DS    ØXL8
         DC    XL2'ØØ1F'       DALDSSEQ
         DC    XL2'ØØØ1'
         DC    XL2'ØØØ2'       LENGTH (--)
TAPESEQ  DC    XL2'Ø'          DATASET SEQUENCE
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'Ø'          FF = ACTIVE
TEXTDCB  DS    ØXL14
         DC    XL2'ØØ2D'       DALDCBDD
         DC    XL2'ØØØ1'
LDCBDD   DC    XL2'ØØØ8'       LENGTH (--)
DCBDD    DC    CL8' '          DDNAME
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'Ø'          FF = ACTIVE
TEXTBLK  DS    ØXL8
         DC    XL2'ØØ3Ø'       DALBLKSZ
         DC    XL2'ØØØ1'
         DC    XL2'ØØØ2'       LENGTH (--)
BLKSIZE  DC    XL2'Ø'          BLOCKSIZE
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'Ø'          FF = ACTIVE
TEXTMBR  DS    ØXL14
         DC    XL2'ØØØ3'       DALMEMBR
         DC    XL2'ØØØ1'
LMEMBER  DC    XL2'ØØØ8'       LENGTH (--)
MEMBER   DC    CL8' '          MEMBER NAME
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'Ø'          FF = ACTIVE
TEXTBFNO DS    ØXL7
         DC    XL2'ØØ34'       DALBUFNO
         DC    XL2'ØØØ1'
         DC    XL2'ØØØ1'       LENGTH (--)
BUFNO    DC    XL1'Ø'          BUFFER NUMBER
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'Ø'          FF = ACTIVE
TEXTDSRG DS    ØXL8
         DC    XL2'ØØ3C'       DALDSORG
         DC    XL2'ØØØ1'
         DC    XL2'ØØØ2'       LENGTH (--)
DSORG    DC    XL2'Ø'          DSORG
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'Ø'          FF = ACTIVE
TEXTLRCL DS    ØXL8
```

```
         DC    XL2'0042'       DALLRECL
         DC    XL2'0001'
         DC    XL2'0002'       LENGTH (--)
LRECL    DC    XL2'0'          LRECL
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'0'          FF = ACTIVE
TEXTOPTC DS    0XL7
         DC    XL2'0045'       DALOPTCD
         DC    XL2'0001'
         DC    XL2'0001'       LENGTH (--)
OPTCD    DC    XL1'0'          OPTCD
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         DC    XL1'0'          FF = ACTIVE
TEXTRECF DS    0XL7
         DC    XL2'0049'       DALRECFM
         DC    XL2'0001'
         DC    XL2'0001'       LENGTH (--)
RECFM    DC    XL1'0'          RECFM
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         TITLE 'USER TABLES'
* TEXT UNIT INDEX for deallocation
TEXUNITD DS    0A
         DC    A(TEXTDDN)       DDNAME MANDATORY
         DC    A(TEXTUNL)       UNALLOC OPTION
         DC    X'FF'
* TEXT UNIT INDEX
TEXUNITT DS    0A
         DC    A(TEXTDDN)       DDNAME MANDATORY
         DC    A(TEXTDSN)       DSNAME MANDATORY
         DC    A(TEXTSTS)       STATUS MANDATORY
         DC    A(TEXTDISP)      NORMAL DISPOSITION MANDATORY
         DC    A(TEXCDISP)      CONDITIONAL DISPOSITION MANDATORY
         DC    A(TEXTUC)        UNALLOCATION AT CLOSE MANDATORY
TEXUNITO DS    0A               OPTIONAL ENTRY ADDRESS
         DC    A(TEXLABEL)      LABEL
         DC    A(TEXTUNIT)      UNIT
         DC    A(TEXTVOL)       VOLSER
         DC    A(TEXTTRK)       SPACE SPECIFIED IN TRACKS
         DC    A(TEXTCYL)       SPACE SPECIFIED IN CYLINDERS
         DC    A(TEXTSPP)       PRIMARY SPACE
         DC    A(TEXTSPS)       SECONDARY SPACE
         DC    A(TEXTDSS)       TAPE DATASET SEQUENCE
         DC    A(TEXTMBR)       MEMBER SELECTION
         DC    A(TEXTDCB)       REFERENCE TO DDNAME SPECIFICATION
         DC    A(TEXTBLK)       BLOCKSIZE
         DC    A(TEXTBFNO)      BUFNO
         DC    A(TEXTDSRG)      DSORG
         DC    A(TEXTLRCL)      LRECL
         DC    A(TEXTOPTC)      OPTCD
         DC    A(TEXTRECF)      RECFM
```

```
         DC    X'FF'
* PARAMETERS DECODING TABLES
TABSTS   DS    ØXL4  THE LAST ELEMENT IS THE DEFAULT
         DC    CL3'MOD',XL1'Ø2'
         DC    CL3'NEW',XL1'Ø4'
         DC    CL3'OLD',XL1'Ø1'
         DC    CL3'SHR',XL1'Ø8' DEFAULT
         DC    X'FF'
TABDISP  DS    ØXL8  THE LAST ELEMENT IS THE DEFAULT
         DC    CL7'UNCATLG',XL1'Ø1'
         DC    CL7'CATLG',XL1'Ø2'
         DC    CL7'DELETE',XL1'Ø4'
         DC    CL7'KEEP',XL1'Ø8'  DEFAULT
         DC    X'FF'
TABLBL   DS    ØXL4  THE LAST ELEMENT IS THE DEFAULT
         DC    CL3'NL',X'Ø1'
         DC    CL3'NSL',X'Ø4'
         DC    CL3'SUL',X'ØA'
         DC    CL3'BLP',X'1Ø'
         DC    CL3'LTM',X'21'
         DC    CL3'AL',X'4Ø'
         DC    CL3'AUL',X'48'
         DC    CL3'SL',X'Ø2' DEFAULT
         DC    X'FF'
TABDSRG  DS    ØXL5
         DC    XL2'ØØØ8',CL3'VS'
         DC    XL2'Ø2ØØ',CL3'PO'
         DC    XL2'Ø3ØØ',CL3'POU'
         DC    XL2'2ØØØ',CL3'DA'
         DC    XL2'21ØØ',CL3'DAU'
         DC    XL2'4ØØØ',CL3'PS'
         DC    XL2'41ØØ',CL3'PSU'
         DC    X'FF'
TABOPTCD DS    ØXL2
         DC    XL1'Ø1',CL1'R'
         DC    XL1'Ø1',CL1'J'
         DC    XL1'Ø2',CL1'T'
         DC    XL1'Ø4',CL1'Z'
         DC    XL1'Ø8',CL1'A'
         DC    XL1'Ø8',CL1'Q'
         DC    XL1'4Ø',CL1'B'
         DC    XL1'8Ø',CL1'W'
         DC    X'FF'
TABRECFM DS    ØXL2
         DC    XL1'Ø2',CL1'M'
         DC    XL1'Ø4',CL1'A'
         DC    XL1'Ø8',CL1'S'
         DC    XL1'1Ø',CL1'B'
         DC    XL1'2Ø',CL1'D'
         DC    XL1'4Ø',CL1'V'
```

```
         DC    XL1'8Ø',CL1'F'
         DC    XL1'CØ',CL1'U'
         DC    X'FF'
TABTRT   DS    ØXL256
BLANK    DC    CL(C'Ø')' '
ZERIEX   DC    1ØX'Ø'
         DC    CL(X'FF'-C'9')' '
ZERIZ    DS    ØZL1Ø
         DC    1ØC'Ø'
* CALLING AREA
AS       DSECT
JCLNAME  DS    CL8
FILENAME DS    CL44
ASDISP   DS    CL7 CATLG DELETE KEEP UNCATLG
ASDISPC  DS    CL7 CATLG DELETE KEEP UNCATLG
ASSTS    DS    CL3 OLD-MOD-NEW-SHR
ASLABEL  DS    CL3 NL SL NSL SUL BLP LTM AL AUL
ASUNIT   DS    CL5 EG 348Ø..
ASVOLSER DS    CL6 VOLUME ID
ASSPACET DS    CL1 EG C = CYLINDERS(DEFAULT) T = TRACKS
ASSPACEP DS    ZL3 PRIMARY SPACE EG Ø2Ø
ASSPACES DS    ZL3 SECONDARY SPACE EG Ø1Ø
ASTAPES  DS    ZL4 TAPE DATASET SEQUENCE
ASDCBR   DS    CL8 REFERENCE TO DDNAME FOR DCB PARAMETERS
ASDSORG  DS    CL3 DATASET ORGANIZATION EG PS PO VS
ASMEMBER DS    CL3 MEMBER FOR DS PARTITIONED
ASRECFM  DS    CL3 RECORD FORMAT EG F  FB  FBS
ASBLKSIZ DS    ZL5 BLOCKSIZE
ASLRECL  DS    ZL5 LRECL
ASBUFNO  DS    ZL3 BUFNO
ASOPTCD  DS    CL3 OPTCD
RES      DS    X   X'ØØ' = ALLOCATION OK:
*                          ON CALL: N = NO WTO FOR SVC99 ERROR AND
*                                       ALLOCATION
*                                   X = NO WTO FOR SVC99 ERROR AND
*                                       DEALLOCATION
*                                   Y = WTO FOR SVC99 ERROR AND
*                                       DEALLOCATION
*                                   OTHERWISE
*                                       WTO AND ALLOCATION
         LTORG
         CSSET
* DYNAMIC ALLOCATION PARM LIST
         IEFZB4DØ
* DYNAMIC ALLOCATION DEFINITION KEY TABLE
         IEFZB4D2
         END    DYNALLOC
```

SAMPLE JCL

```
Create DFHCSDUP listing dataset
//CSDLIST  EXEC PGM=DFHCSDUP,REGION=5M,
//           PARM='CSD(READONLY)'
//*
//STEPLIB  DD DSN=CICS.SDFHLOAD,DISP=SHR
//DFHCSD   DD DSN=MY.DFHCSD,DISP=SHR
//SYSPRINT DD DSN=CSD.LIST,DISP=SHR         <   List dataset
//SYSIN    DD *
 LIST GROUP(CICSFPL) OBJECTS
//*

Execute CSDVER
//CSDVER   EXEC PGM=CSDVER,REGION=7M
//STEPLIB  DD DSN=MY.LOADLIB,DISP=SHR
//CSDPRT   DD DSN=CSD.LIST,DISP=SHR          <  List dataset
//CSDVARY  DD DSN=CSD.SYSIN,DISP=SHR         <  Control Card Dataset
//PRINT    DD SYSOUT=*                       <  Report
//TRACE    DD DUMMY                       <  No Trace Report is required
//*
```

Sample output control statements: input to the next DFHCSDUP job

```
*-------------------------------------------------------------------
* CXW.CODICE.UFFICI
ALTER FILE(UFFILE  ) GROUP(CICSFPL ) KEYLENGTH(ØØ3)
ALTER FILE(UFFILE  ) GROUP(CICSFPL ) RECORDSIZE(ØØØ81)

Modify DFHCSD
//CSDALT   EXEC PGM=DFHCSDUP,REGION=5M
//*
//STEPLIB  DD DSN=CICS.SDFHLOAD,DISP=SHR
//DFHCSD   DD DSN=MY.DFHCSD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DSN=CSD.SYSIN,DISP=SHR      <  Control statements dataset
//*
```

Giuseppe Rallo
Senior Technical Analyst
Sicilcassa spa (Italy)

REXX extensions for VSAM

Editor's note: This article, which will be serialized over several issues, focuses on VSAM file processing, but also extends to cover date processing and data conversion – activities in which data administrators will frequently be engaged.

REXXPLUS/MVS is a set of functions and subroutines that extend IBM REXX. These functions are built on the standard facilities of REXX, and they have the same syntax as REXX.

REXXPLUS/MVS is useful in the following areas:

- *Date processing*. REXXPLUS/MVS converts dates, adds days to a date, or tests the validity of a given date.
- *File processing*. REXXPLUS/MVS provides a complete set of functions for VSAM, QSAM, and BPAM processing. VSAM files can be processed with all the options available in the Assembler language (except CI processing), including backward read, higher or equal key, and so on.
- *Data conversion*. REXXPLUS/MVS allows you to convert extended numeric data into packed decimal data and floating point data, and vice versa.

INSTALLATION

In order to install REXXPLUS/MVS, follow the instructions below:

1 Assemble (or compile, for the COBOL program) all the programs in a library (eg MY.LIB) with the RENT attribute, except for IRXFLOC and $IRXCNV2, which must be non-reentrant, non-reusable, and except for $IRXTERM which must be link-edited reentrant, AC=1, in your linklist. The $IRXTERM program is a clean-up routine.

2 Copy the SYS1.SAMPLIB(TSOREXX1) member into one of your own libraries, and update it by replacing the following line:

```
+------------------------------------------------------------------+
! MODNAMET_EXECTERM DC CL8'        '                               !
+------------------------------------------------------------------+
```

with:

```
+------------------------------------------------------------------+
! MODNAMET_EXECTERM DC CL8'$IRXTERM'                               !
+------------------------------------------------------------------+
```

Then assemble this new member, and put the load module into your library, MY.LIB, with the name IRXPARMS (this name is fixed by IBM), with the AC=0 and RENT attributes.

3 Include the library MY.LIB in the STEPLIB of your TSO log-on procedure and the STEPLIB of all batch jobs calling REXXPLUS/MVS.

Note that the $IRXTERM clean-up routine is called only by those address spaces that have the modified IRXPARMS in their STEPLIB. $IRXTERM is not called from standard IBM REXX routines that have the standard IRXPARMS, so installing $IRXTERM in your linklist does not have any effect on your standard REXX.

DATA CONVERSION FUNCTIONS

The data conversion functions of REXXPLUS/MVS are described below.

$DF2D

```
+------------------------------------------------------------------+
! $DF2D(string)                                                    !
+------------------------------------------------------------------+
```

This function returns the decimal value of the floating point representation of the string, whose length must be eight. For example:

```
$DF2D('4112000000000000'X) -> +.1125000000000000E+01
```

$D2DF

```
+------------------------------------------------------------------+
! $D2DF(number)                                                    !
+------------------------------------------------------------------+
```

This function returns a character string that is the double precision floating point representation of the decimal number. For example:

```
$D2DF(1.125) -> '4112000000000000'X
```

$D2P

```
+-----------------------------------------------------------------+
! $D2P(number, length)                                            !
+-----------------------------------------------------------------+
```

This function returns a character string that is the packed decimal representation of the decimal number. The length of the result is specified by ‘length’, which must represent a whole number from 0 to 16. If the length is greater than 16, it is set to 16. If the length is not high enough to accommodate the result, it is truncated on the left.

Two examples of the $D2P function are given below:

```
$D2P(12,4) -> '0000012F'X
$D2P(-12345,2) -> '345D'X (left truncation)
```

$D2SF

```
+-----------------------------------------------------------------+
! $D2SF(number)                                                   !
+-----------------------------------------------------------------+
```

This function returns a character string that is the single precision floating point representation of the decimal number. For example:

```
$D2SF(1.125) -> '41120000'X
```

$P2D

```
+-----------------------------------------------------------------+
! $P2D(string)                                                    !
+-----------------------------------------------------------------+
```

This function returns the decimal value of the packed decimal representation of the string. The string must have a length of less than or equal to 16. The resulting number has 31 digits at the most, and a sign.

Two examples of the $P2D function are given below:

```
$P2D('Ø12F'X) -> 12
$P2D('1D'X)  -> -1
```

$SF2D

```
+-------------------------------------------------------------------+
! $SF2D(string)                                                     !
+-------------------------------------------------------------------+
```

This function returns the decimal value of the floating point representation of the string. The string length must be four. For example:

```
$SF2D('41120000'X) -> +.1125000000E+01
```

$SHIFT

```
+-------------------------------------------------------------------+
! $SHIFT(string<,n<,pad>>)                                          !
+-------------------------------------------------------------------+
```

This function returns a character string with the same length as ‘string’, where the bits have been shifted *n* bits to the left.

Before shifting, the padding character, comprising the first byte of ‘pad’, is concatenated on the right of the ‘string’, so that the freed bits on the right of the ‘string’ are replaced by the first bits of the ‘pad’. The ‘*n*’ first bits of the ‘string’ are lost.

The ‘string’ length must be between 0 and 32. ‘N’ must represent a whole number between zero and eight, and the default value is four. ‘Pad’ is any expression from which only the first byte is used. The default value for ‘pad’ is 'FF'X.

If required, you can prevent these bits being lost by concatenating a '00'X in front of ‘string’ (see below).

$SHIFT('00'X||STRING)

This function is used for adding a sign to a string containing a signless packed decimal value. For example:

```
$SHIFT('Ø123'X) -> '123F'X
```

adds a ‘+’, and

```
$SHIFT('Ø123'X,,'DO'X) -> '123D'X
```

adds a ‘-’.

DATE AND TIME CONVERSION FUNCTIONS

The date and time conversion functions of REXXPLUS/MVS are described below.

$CLOCK

```
+-------------------------------------------------------------+
! $CLOCK(expression)                                          !
+-------------------------------------------------------------+
```

This function returns the computer’s TOD clock. See IBM documentation for details on the ‘TOD CLOCK’. Although ‘expression’ must be a valid expression, it is not actually used.

An example of the $CLOCK function is given below:

```
$CLOCK(Ø) -> 'AA64338DCAF6E311'X /* MAY BE */
```

$CLOCKDT

```
+-------------------------------------------------------------+
! $CLOCKDT(string)                                            !
+-------------------------------------------------------------+
```

This function returns the ISO date and time value of the string, which must be a TOD clock. The length of the string must be eight.

An example of the $CLOCKDT function is given below:

```
$CLOCKDT('AA64338DCAF6E311'X) -> 1994-12-27-22.5Ø.14.113134
```

$DATEREF subroutine

```
+-------------------------------------------------------------+
! $DATEREF number                                             !
+-------------------------------------------------------------+
```

This subroutine fixes a reference year that will be used in subsequent date processing functions. The dates will have to be supplied relative

to this reference, and the functions will return dates relative to this reference (except for the N and ISO formats). When this subroutine is not called, the effect is the same as calling $DATEREF 0.

For example:

```
CALL $DATEREF 0 : 19971231
```

represents 31 December 1997, while:

```
CALL $DATEREF 1900 :
```

fixes the reference year to 1900, so that 971231 represents 31 December 1997.

$DATE

```
+-----------------------------------------------------------------+
! $DATE(date, input_format, output_format)                        !
+-----------------------------------------------------------------+
```

Converts the ‘date’ from the ‘input_format’ to the ‘output_format’.

The input formats are as follows:

- ‘YMD’ or ‘AMJ’. Year, month, day (or année, mois, jour). The input date must be in the format YYYYMMDD. For example, 19950131 represents 31 January 1995. Alternatively, if CALL $DATEREF 1900 has been executed, 950131 represents 31 January 1995.
- ‘YQ’ or ‘AQ’. Year, day in the year. The input date must be in the format YYYYQQQ, so that 1995123 represents the 123rd day in the year 1995.
- ‘N’. Number of days since 1 January 1601 – for example, 145200 represents the 145200th day since 1 January 1601. Note that because this format is independent of the $DATEREF subroutine, the resulting date is always relative to 1 January 1601, whatever $DATEREF subroutine has been executed.

The output formats are as follows:

- ‘YMD’ or ‘AMJ’. The date is returned in the format YYYYMMDD.

- ‘YQ’ or ‘AQ’. The date is returned in the format YYYYQQQ.
- ‘N’. Number of days since 1 January 1601.
- ‘YWD’ or ‘ASJ’. Year, week, day in the week (or année, semaine, jour dans la semaine). The date is returned in the format YYYYWWD.

  According to the French Bureau des Longitudes (and I think these are international norms):

  – The week begins on Monday.

  – Weeks are numbered from 1 to 52, or to 53 if the last week ends on a Thursday, or on a Thursday or a Friday if the year is a leap year (although this rule might seem complex, if you look at a calendar, you will see it is not!).

  – Week number 1 is the first week containing a Thursday – that is, the first week having at least four days.

    According to this norm, some days at the beginning and end of the year may have no week number. Because REXXPLUS/MVS must always return a valid number for a valid date, it returns 0 for any ‘short’ first week with less than four days, and 53 for any ‘short’ last week of the year. This ensures that all computations are both valid and consistent.

    Note that a leap year is a year that is divisible by four, except when it is divisible by 100, unless it is divisible by 400. For example, 1904, 1600, and 2000 are all leap years, whereas 1700, 1800, and 1900 are not.

- ‘EASTER’ or ‘PAQUES’. Date of Easter Monday. The date is returned in the same format as the input format.
- ‘ASCENSION’. Date of Ascension Thursday. The date is returned in the same format as the input format.
- ‘PENTECOST’ or ‘PENTECOTE’. Date of Pentecost Monday (not the Sunday). The date is returned in the same format as the input format.

- ‘LDM’ or ‘DJM’. Last day of the month (or dernier jour du mois). Returns the date of the last day in the month of the input date, in the same format as the input format.
- ‘FDM’ or ‘PJM’. First day of the month (or premier jour du mois). Returns the date of the first day in the month of the input date, in the same format as the input format.
- ‘DW’ or ‘JS’. Day of the week (or jour de la semaine). Returns the number of the day in the week of the input date (with 1 for Monday, and so on)
- ‘ISO’. Returns the date in the ISO format YYYY-MM-DD, whatever $DATEREF subroutine has been executed.
- ‘TEST’. Returns 1 if the date is valid and 0 if not.
- ‘M’. Month (or mois). Returns the number of the month of the input date (with 1 for January, and so on).
- ‘DM’ or ‘JM’. Day of the month (or jour du mois). Returns the day of the month of the input date, from 1 to 28, 29, 30, or 31.
- ‘Y’ or ‘A’. Year (or année). Returns the year of the input date.
- ‘DY’ or ‘JA’. Day of the year (or jour de l’année). Returns the number of the day of the input date, from 1 to 365 or 366.

Note that the input and output formats must be in upper case.

Some examples of the $DATE function are given below:

- Convert the date 19920102 from format YYMMDD to format YYQQQ:

```
$DATE(19920102,'YMD','YQ') -> 1992002 /* 2ND DAY OF 1992 */
```

- What date is obtained by adding 15 days to 20 December 1995?

```
$DATE($DATE(19951220,'YMD','N')+15,'N','YMD') -> 19960104
                                    /* JANUARY 4TH, 1996 */
```

- Is the date 19930229 (format YYMMDD) valid?

```
$DATE(19930229,'YMD','TEST') -> 0 /*FEB 29TH, 1993 DOES NOT
EXIST*/
```

$DATE with 'ADD'

```
+-------------------------------------------------------------+
! $DATE(date, input_format, 'ADD', number)                     !
+-------------------------------------------------------------+
```

This function returns the date in the ‘input-format’ obtained by adding the ‘number’ of days to the ‘date’ in the ‘input_format’. The ‘input_format’ must be one of the input formats described for the $DATE function (see above). ‘Number’ must represent a positive, negative, or null whole number.

The $DATE(date,format,'ADD',n) function is equivalent to the $DATE($DATE(date,format,'N')+n,'N',format) function.

For example:

What date is obtained by subtracting one day from 1 January 1995?

```
$DATE(19950101,'YMD','ADD',-1) -> 19941231 /*DECEMBER 31ST, 1994*/
```

$DATE with 'DIFF'

```
+-------------------------------------------------------------+
! $DATE(date1, input_format, 'DIFF', date2)                    !
+-------------------------------------------------------------+
```

This function returns the number of days between ‘date1’ and ‘date2’, both in the ‘input_format’. The ‘input_format’ must be one of the input formats described for the $DATE function (see above).

If ‘date2’ is later than ‘date1’, the result is negative.

The $DATE(date,format,'DIFF',date2) function is equivalent to the $DATE(date,format,'N')-$DATE(date2,format,'N') function.

For example:

How many days are there between 1 January 1995, and 1 February 1995?

```
$DATE(19950201,'YMD','DIFF',19950201) -> 31
```

$DATE with 'NBRDM'

```
+-------------------------------------------------------------+
! $DATE(date, input_format, 'NBRDM', day)                      !
+-------------------------------------------------------------+
```

('NBRDM' means 'number of days in the month' (the synonymous 'NBRJM', for 'nombre de jours dans le mois', may also be used)).

This function returns the number of days of the 'day' kind in the month of the date 'date' in the format 'input_format'. The 'input_format' must be one of the input formats described for the $DATE function (see above). 'Day' must represent a whole number from one for Monday to seven for Sunday. The input date must be a valid date, but its day is not used.

For example:

How many Tuesdays were there in January 1995 ?

```
$DATE(19950101,'YMD','NBRDM',2) -> 5
```

$DATE with 'DATEDM'

```
+-------------------------------------------------------------+
! $DATE(date, input_format, 'DATEDM', day, n)                  !
+-------------------------------------------------------------+
```

(The synonymous 'DATEJM' may also be used for 'DATEDM'.)

This function returns the date in the 'input_format' of the 'N'th day of the 'day' kind, in the month of the 'date' in the format 'input_format'. The 'input_format' must be one of the input formats described for the $DATE function (see above).

Note that 'day' must represent a whole number, from one for Monday to seven for Sunday. 'N' must represent a whole number from 1 to 4 or 5, depending on the number of days of the 'day' kind there are in the month of the input date. The input date must be a valid date, but its day is not used.

For example:

What is the date of the third Monday in January 1995?

```
$DATE(19950101,'YMD','DATEDM',1,3) -> 19950116 /*JAN 16TH, 1995*/
```

Sample program for dates processing

A sample program using dates processing functions is shown below.

```
/* REXX - DATES */
CALL $DATEREF 1900
SAY 'GIVE THE INPUT FORMAT FOR YOUR DATE'; PULL FMT
DO WHILE FMT¬=''
  SAY 'GIVE A DATE IN FORMAT '||FMT; PULL REP
  TST=$DATE(REP,FMT,'TEST')
  IF TST=1
  THEN DO
       SAY 'DAY IN THE WEEK=       ' $DATE(REP,FMT,'DW')
       SAY 'EASTER MONDAY=         ' $DATE(REP,FMT,'EASTER')
       SAY 'FIRST DAY IN THE MONTH=' $DATE(REP,FMT,'FDM')
       SAY 'YYQQQ=                 ' $DATE(REP,FMT,'YQ')
  END
  ELSE SAY 'THIS DATE IS INVALID'
  SAY 'GIVE THE INPUT FORMAT FOR YOUR DATE'; PULL FMT
END
```

FILES PROCESSING FUNCTIONS

The files processing functions of REXXPLUS/MVS are described below.

$CLOSE

```
code=$CLOSE(ddname)
```

This function closes the file with the DDNAME ‘ddname’.

Note that this function returns the return code of the IBM macro used by REXXPLUS/MVS. When this return code is not 0, the REASON variable contains the reason code, and the FUNCTION variable contains the name of the IBM macro in error. See IBM documentation for these return and reason codes.

An example of the $CLOCK function is given below:

```
CODE=$CLOSE('SYSUT1')
```

$ERASE

```
+-------------------------------------------------------------+
! code=$ERASE(ddname)                                         !
+-------------------------------------------------------------+
```

This function deletes a record in the VSAM dataset with the DDNAME 'ddname'. For the returned value, see the note in the $CLOSE function (above).

An example of the $ERASE function is given below:

```
CODE=$ERASE('F1')
```

$FIND

```
+-------------------------------------------------------------+
! code=$FIND(ddname, member)                                  !
+-------------------------------------------------------------+
```

This function moves to the 'member' in the partitioned dataset (or datasets) with the DDNAME 'ddname', so that the next read instruction will read the first record of this member.

It returns one of the following return codes:

- 0 if OK
- 4 if the member does not exist
- 8 for any other error (I/O error, insufficient memory, etc).

An example of the $FIND function is given below:

```
CODE=$FIND('PDS','MEMBER1')
```

$GET

```
+-------------------------------------------------------------+
! code=$GET(ddname<,key_for_VSAM>)                            !
+-------------------------------------------------------------+
```

This function reads the dataset with the DDNAME 'ddname', and returns the record in the <ddname>_RECORD variable (eg the SYSUT1_RECORD for the SYSUT1 file). For the returned value, see the note in $CLOSE function (above).

For a VSAM dataset, 'key_for_VSAM' allows the key of the desired

record, or its RBA (relative byte address), or its RRN (relative record number) to be supplied, depending on the options specified in the $MODRPL function. If the key is an RRN (for an RRDS), it must be four bytes long, in binary. For example, the following instruction reads the 35th record of the 'F1' RRDS:

```
CODE=$GET('F1',D2C(35,4))
```

At end of the file, the <ddname>_EOF variable (eg SYSUT1_EOF for the SYSUT1 file) is set to 1. Note that this variable is not set to 0 when the file is opened, nor when the $GET does not reach the end of the file.

Two examples of the $GET function are given below:

```
CODE=$GET('SYSUT1')
CODE=$GET('VSAM1','0000001')
```

$GETA

```
+-----------------------------------------------------------------+
! code=$GETA(ddname<,key_for_VSAM>)                                !
+-----------------------------------------------------------------+
```

This function works like $GET, but returns the record address in the <ddname>_RECORD variable (eg SYSUT1_RECORD for the SYSUT1 file), instead of the record itself. The length of the record is returned in the <ddname>_RECORDL variable (eg SYSUT1_RECORDL for the SYSUT1 file).

Two examples of the $GETA function are given below:

```
CODE=$GETA('SYSUT1')
CODE=$GETA('VSAM1','0000001')
```

$LOCATE

```
+-----------------------------------------------------------------+
! code=$LOCATE(dsn, variable)                                      !
+-----------------------------------------------------------------+
```

This function assigns the list of the volumes on which the dataset 'dsn' resides to the variable 'variable'. The data is formatted by the LOCATE macro. LOCATE returns as many entries as there are volumes where the dataset resides, and one entry is formatted as follows:

- Four bytes for the type of unit ('3010200E' for 3380, X'3010200F' for 3390, etc).
- Six bytes for the VOLSER.
- Two bytes for the sequence number, in binary.

One entry is twelve bytes long, so the variable is (12*number of VOLSER) bytes long. It returns the code returned by the LOCATE macro, namely 0 if OK, or 8 if the dataset does not exist (see IBM documentation for further details).

An example of the $LOCATE function is given below:

```
CODE=$LOCATE('MY.DATASET','VOLS')
```

$MODRPL

```
+-------------------------------------------------------------+
! code=$MODRPL(ddname, parameters<,length_of_record>)          !
+-------------------------------------------------------------+
```

This function modifies the parameters of a request on the VSAM dataset with the DDNAME ‘ddname’. For the returned value, see the note in the section on the $CLOSE function (above).

Note that ‘parameters’ must contain one or more of the following strings, separated by commas:

- ‘KEY’ for an access by key or RRN.
- ‘ADR’ for an addressed access.
- ‘DIR’ for a direct access.
- ‘SEQ’ for a sequential access.
- ‘SKP’ for a skip sequential access.
- ‘ARD’ for an access according to the supplied arguments.
- ‘LRD’ for an access to the last record.
- ‘FWD’ for a forward access.
- ‘BWD’ for a backward access.

- ‘NSP’ for VSAM to remember the positioning.
- ‘NUP’ if a record is not to be updated.
- ‘UPD’ if a record may be deleted or updated.
- ‘KEQ’ for equal key.
- ‘KGE’ for higher or equal key.
- ‘FKS’ for a full key.
- ‘GEN’ for a generic key.

‘Length_of_record’ must represent a whole number which, for output files, will be the length of the next records to be written; for input files with the GEN option, it will be the length of the generic key.

An example of the $MODRPL function is given below:

```
CODE=$MODRPL('F1','ADR,DIR')
```

$OBTAIN

```
+-------------------------------------------------------------------+
! code=$OBTAIN(dsn, volume, variable <, 'D'>)                       !
!     or                                                            !
! code=$OBTAIN(seek, volume, variable, 'S')                         !
+-------------------------------------------------------------------+
```

This function assigns a DSCB of the dataset ‘dsn’ that resides on the volume ‘volume’ to the variable ‘variable’.

For the first format of this function (with ‘D’ in the fourth argument), format 1 DSCB is returned. For the second format, (with ‘S’ in the fourth argument), format 3 DSCB is returned. In this case, ‘seek’ must be five bytes long, and must be the address of the DSCB to be read. This address is located in DS1PTRDS of the format 1 DSCB for the dataset.

An example of the $OBTAIN function is given below:

```
CODE=$OBTAIN('MY.DATASET','VOL001','VARDSCB')
```

$OPEN

```
+-------------------------------------------------------------+
! code=$OPEN(ddname, open_type <,parameters_for_VSAM>)         !
+-------------------------------------------------------------+
```

This function opens the file with the ddname ‘DDNAME’. For the returned value, see the note in the section on the $CLOSE function (above).

Note that ‘open_type’ must be one of the following:

- ‘INPUT’ for an OS sequential input dataset.
- ‘OUTPUT’ for an OS sequential output dataset.
- ‘BPAMI’ for a partitioned input dataset, where RECFM must be F(B) or U.
- ‘BPAMO’ for a partitioned output dataset, where RECFM must be F(B) or U.

  Note that because REXXPLUS/MVS transforms the BPAM access method into a QPAM access method, we are interested in records only, and not in block management.
- ‘VSAM’ for a VSAM file.

The argument ‘parameters_for_VSAM’ must contain one or more of the following keywords, separated by commas (more details of these values are supplied in the IBM documentation for the GENCB ACB= macro, MACRF=keyword):

- ‘KEY’ for an access by key or RRN.
- ‘ADR’ for an addressed access.
- ‘DIR’ for a direct access.
- ‘SEQ’ for a sequential access.
- ‘SKP’ for a skip sequential access.
- ‘IN ’ for an input access.
- ‘OUT’ for an output access.

- ‘NRM’ for an access to the dataset itself.
- ‘AIX’ for an access to the alternate index.
- ‘NIS’ for a normal insert.
- ‘SIS’ for a sequential insert.
- ‘NRS’ for a non reusable dataset.
- ‘RST’ for a reusable dataset.

Because these options must be three bytes long, the ‘IN’ in the ‘IN ’ option must be followed by a blank (eg ‘IN ,OUT,DIR’).

Two examples of the $OPEN function are given below:

```
RC=$OPEN('SYSUT1','INPUT')
RC=$OPEN('F','VSAM','IN ,OUT,DIR')
```

$OPTION subroutine

```
+-----------------------------------------------------------------+
! CALL $OPTION parameter                                          !
+-----------------------------------------------------------------+
```

This subroutine allows you to specify options for file processing. The parameter must be one of the following:

- ‘LLZZ’. Data for an output file or from an input file with RECFM V(B) beginning with four bytes (LLZZ) containing the record length.
- ‘NLLZZ’. Data for an output file or from an input file with RECFM V(B) which does not contain the LLZZ field.

If this subroutine is not called, the ‘LLZZ’ option is in effect. Your records must therefore begin with LLZZ, and records returned by REXXPLUS/MVS begin with LLZZ (this applies only for V(B) files).

An example of the $OPTION subroutine is given below:

```
CALL $OPTION 'NLLZZ'
```

$POINT

```
+-------------------------------------------------------------+
! code=$POINT(ddname <, parameter>)                           !
+-------------------------------------------------------------+
```

This function allows you to position on the VSAM file with the DDNAME ‘ddname’. The parameter allows you to specify the record on which you want to position. The parameter can be an RRN, an RBA, or a key, depending on the access mode. More details of these values are supplied in the IBM documentation for the GENCB macro, ARG= keyword.

For the returned value, see the note in the section on the $CLOSE function (above).

An example of the $POINT function is given below:

```
CODE=$POINT('F1',KEY)
```

$PUT

```
+-------------------------------------------------------------+
! code=$PUT(ddname, record, <rrn>)                            !
+-------------------------------------------------------------+
```

This function writes the record ‘record’ to the file with the DDNAME ‘ddname’.

The argument ‘rrn’ is valid (and mandatory) only for VSAM RRDS files. It must represent the Relative Record Number, in binary.

An example of the $PUT function is given below:

```
CODE=$PUT('F1',RECORD,D2C(8,4))
```

For details on the returned value, see the note in the section on the $CLOSE function (above).

Another example is as follows:

```
CODE=$PUT('SYSUT1','LINE 1')
```

$RPL subroutine

```
+-------------------------------------------------------------+
! CALL $RPL ddname, rpl_number                                !
+-------------------------------------------------------------+
```

This subroutine allows you to specify the RPL (Request Parameter List) that will be used for the next operation on the VSAM dataset with the DDNAME ‘ddname’. The argument ‘rpl_number’ must represent a whole number between one and four. This means that you have four different RPLs for one VSAM file at your disposal.

When this subroutine is not called, the RPL number 1 is used.

An example of the $RPL function is given below:

```
CALL $RPL 'SYSUT1', 2
```

$SHOWACB

```
+-------------------------------------------------------------+
! code=$SHOWACB(ddname, variable, parameter <, codef>)        !
+-------------------------------------------------------------+
```

This function assigns an area of the ACB of the VSAM file with the DDNAME ‘ddname’ to the variable ‘variable’.

For details on the returned value, see the note in the section on the $CLOSE function (above).

The argument ‘codef’ allows you to specify which part of the CLUSTER you want to process. Its value must be either ‘D’ (the default value) if you want to process the data part, or ‘I’ if you want to process the index part.

The argument ‘parameter’ specifies which area of the ACB you want to obtain. Some possible values are given below:

- ‘AVSPAC’ – free space.
- ‘BFRFND’ – number of anticipated reads made successfully.
- ‘BUFNO’ – number of buffers.
- ‘BUFRDS’ – number of reads made in the buffers.
- ‘CINV’ – CISIZE.
- ‘ENDRBA’ – last RBA of used space.
- ‘FS’ – number of free CI by CA.

- ‘HALCRBA’ – highest allocated RBA.
- ‘KEYLEN’ – key length.
- ‘LRECL’ – record length.
- ‘NCIS’ – number of CI splits.
- ‘NDELR’ – number of deleted records.
- ‘NEXCP’ – number of EXCP.
- ‘NEXT’ – number of extents.
- ‘NINSR’ number of inserted records.
- ‘NIXL’ – number of index levels.
- ‘NLOGR’ – number of records.
- ‘NRETR’ – number of read records.
- ‘NSSS’ – number of CA splits.
- ‘NUIW’ – number of writes not initiated by user.
- ‘NUPDR’ – number of updated records.
- ‘RKP’ – key offset in the record.
- ‘UIW’ – number of writes initiated by user.
- ‘ERROR’ – return code from OPEN.

More details can be obtained from the IBM documentation for the SHOWCB ACB=macro, FIELDS= keyword.

An example of the $SHOWACB function is given below:

```
code=$SHOWACB('F1','RESULT','LRECL')
```

Editor’s note: This article will be continued in the next and subsequent issues.

Patrick Leloup
System Engineer
Credit Agricole de Loire Atlantique (France)

Transferring code from the Web to a mainframe

Editor's note: although this article was written by an MVS Update *subscriber, the method followed by the ISPF edit macro can be used by others downloading* Update *code to a mainframe.*

When a colleague of mine recently downloaded an *MVS Update* article from the Xephon Web site to his PC and then uploaded it to his MVS system, he found to his disappointment that the program code would not run properly.

It was a REXX program, and, when he executed it, he received the following message:

```
IRX0013I Error running XXXXXXXX, line nn: Invalid character in program
```

This was rather puzzling, but a quick look at the code revealed that the offending character was a REXX 'not' (that is ^, in a ^= expression), which should be a hex value X'5F', but was instead a X'B0'. The REXX interpreter was rejecting this value. Another odd character turned out to be the '|' operator, which should be X'4F', but was X'6A'.

Having discovered this, it was trivial to code an ISPF edit macro to fix this and to cater for it in future uploads:

```
ISREDIT MACRO
ISREDIT CHANGE ALL X'B0' X'5F'
ISREDIT CHANGE ALL X'6A' X'4F'
EXIT
```

The PC was running IBM Personal Communications 3270 Version 4.1 for Windows with an IEEE 802.2 connection to the host, code page 037. The upload was achieved using the IBM 3270 PC File Transfer Program for MVS/TSO Release 1.1.1 using the following command:

```
IND$FILE PUT XEPHFILE.TEXT ASCII CRLF RECFM(V) LRECL(133)
```

It seems that the ASCII to EBCDIC conversion taking place works fine for alphanumeric characters, but is suspect for unusual ones. Readers should be aware of this when transferring code.

Patrick Mullen
MVS Systems Consultant (Canada)

Increasing file space allocation

There are often times when you need to increase the size of a file being edited. For example:

- The directory of a PDS can be filled up, and using COMPRESS will not create more space.
- When copying the members from one PDS to another, and the receiving PDS has insufficient space.

The following REXX program increases file space allocation for VSAM files.

INVSAM

```
/* REXX */
/*********************************************************************/
/*********************************************************************/
/*********************************************************************/
/**       INCREASE VSAM FILE                                 *********/
/*********************************************************************/
SEC = Ø
SEC2 = Ø
PRI = Ø
PRI2 = Ø
FILEN = ''
FILE  = ''
IF ARG()  = Ø   THEN CALL MAP
                ELSE DO
                PARSE ARG FILEN
                IF FILEN = '' THEN EXIT 4
ADDRESS TSO     "LISTCAT ENTRY("FILEN") ALL "
                  IF RC = Ø
                  THEN CALL PROC
                      ELSE EXIT RC
                     END
PROC:
ADDRESS TSO     "FREE FILE(FILOUT)"
ADDRESS TSO     "FREE FILE(EXIT)"
SAY FILEN         'FILE NAME OF DATASET TO BE INCREASE '
    ADDRESS TSO "DELETE '"USERID()".LISTCAT' "
          IF RC > 8 THEN EXIT RC
                    ELSE NOP
    ADDRESS TSO "ALLOCATE DATASET('"USERID()".LISTCAT')
```

```
     FILE(EXIT)",
                         "DSORG(PS) SPACE(1,Ø) TRACKS RELEASE",
                         "LRECL(125) BLKSIZE(629) RECFM(V,B)",
                         "MGMTCLAS(NMIGNSAV) STORCLAS(BASE)",
                         "NEW CATALOG"
ADDRESS TSO     "LISTCAT ENTRY("FILEN") ALL    OUTFILE(EXIT)"
          IF RC > 8 THEN EXIT RC
                    ELSE NOP
ADDRESS MVS "EXECIO * DISKR EXIT (STEM T. FINIS"
 I=1
 DO UNTIL I = T.Ø
 SAY T.I
 A=LEFT(T.I,33)
 A=STRIP(A)
 SAY A
 L = LENGTH(A)
 RESTE = L - 1Ø
 IF SUBSTR(A,1,1Ø) = 'SPACE-TYPE' THEN TYPE = SUBSTR(A,11,2Ø)
                           ELSE NOP
 IF SUBSTR(A,1,9) = 'SPACE-PRI' THEN PRI = SUBSTR(A,1Ø,2Ø)
                           ELSE NOP
 IF SUBSTR(A,1,9) = 'SPACE-SEC' THEN DO
                    SEC = SUBSTR(A,1Ø,2Ø) END ='OK'
                    END ='OK'
                    END
                           ELSE NOP
 IF END = 'OK' THEN LEAVE
 I = I + 1
  END
 SEC = STRIP(SEC,,'-')
 PRI = STRIP(PRI,,'-')
 TYPE = STRIP(TYPE,,'-')
 TYPE = STRIP(TYPE)
 SAY TYPE PRI SEC
 PRI2 = PRI * 2
 SEC2 = SEC * 2
ADDRESS TSO     "FREE FILE(EXIT)"
X = LISTDSI(FILEN "DIRECTORY" "RECALL")
FILE = SYSDSNAME
/*********************************************/
/* USE OF TEMPORARY FILE                     */
/*            USERID.DSNAME                  */
/*********************************************/
    ADDRESS TSO "DELETE '"USERID()"."FILE"'"
          IF RC > 8 THEN EXIT RC
                    ELSE NOP
/*********************************************************************/
/**                                                                 **/
/**     ALLOCATION AFTER DELETE OF THE TEMPORARY DATASET            **/
/**                                                                 **/
/*********************************************************************/
```

```
 ADDRESS TSO "ALLOCATE DATASET('"USERID()"."FILE"') FILE(FILOUT)
       LIKE("FILEN")
          SPACE("PRI2","SEC2") "TYPE"
       NEW CATALOG"
          IF RC > 8 THEN EXIT RC
                    ELSE NOP
/*********************************************************************/
/*********************************************************************/
/**                                                                 **/
/**     COPY A VSAM FILE                                            **/
/**                                                                 **/
/*********************************************************************/
ADDRESS TSO     "FREE FILE(SYSIN)"
ADDRESS TSO     "FREE FILE(SYSPRINT)"
ADDRESS TSO "ALLOCATE FILE(SYSPRINT) DSN(*) REUSE "
ADDRESS TSO "ALLOCATE FILE(SYSOUT) DSN(*) REUSE "
ADDRESS TSO "ALLOCATE FILE(SYSIN)
     SPACE(1,1) TRACK LRECL(8Ø) RECFM(F) BLKSIZE(8Ø) REUSE"
UPPER FILEN
PUSH " REPRO IDS("FILEN") ODS('"USERID()"."FILE"')"
ADDRESS MVS "EXECIO 1 DISKW  SYSIN ( FINIS"
/* CALL IDCAMS */
ADDRESS TSO "TSOEXEC CALL 'SYS1.LINKLIB(IDCAMS)'"
 SELECT
   WHEN RC=Ø  THEN
                DO
                   ZEDLMSG="THE FILE"FILEN" HAS BEEN COPIED"
                   ZEDSMSG="OK"
                   "SETMSG MSG(ISRZØØ1)"
                    EXIT RC
                END
   WHEN RC=4  THEN
                DO
                   ZEDLMSG="THE FILE "FILEN" DOES NOT EXIST OR FILE",
                   " VIDE ; RC="RC
                   ZEDSMSG="EMPTY FILE !! "
                   "SETMSG MSG(ISRZØØ1)"
                    EXIT RC
                END
   WHEN RC=8  THEN
                DO
                   ZEDLMSG="THE FILEN "FILEN" DOES NOT EXIST;RC="RC
                   ZEDSMSG="FILEN NOT FOUND !! "
                   "SETMSG MSG(ISRZØØ1)"
                    EXIT RC
                END
   OTHERWISE
                DO
                   ZEDLMSG="ERROR SMCOPY ; RC="RC
                   ZEDSMSG="ERROR"
```

```
                   "SETMSG MSG(ISRZØØ1)"
                    EXIT RC
                END
 END
ADDRESS TSO     "FREE FILE(FILOUT)"
ADDRESS TSO     "FREE FILE(SYSIN)"
ADDRESS TSO     "FREE FILE(SYSPRINT)"
ADDRESS TSO     "FREE FILE(SYSOUT)"
 EXIT RC
RETURN
MAP:
PARSE  EXTERNAL FILEN
SAY FILEN         'FILE NAME OF DATASET TO BE INCREASED '
ADDRESS TSO     "LISTCAT ENTRY("FILEN") ALL "
                  IF RC = Ø
                  THEN CALL PROC
                      ELSE EXIT RC
RETURN RC
EXIT   RC
```

Claude Dunand
(France)

Contributing to *VSAM Update*

Although the articles published in *VSAM Update* are of a very high standard, the vast majority are not written by professional writers, and we rely heavily on our readers themselves taking the time and trouble to share their experiences with others. Many have discovered that writing an article is not the daunting task that it might appear to be at first glance.

If you have ever experienced any difficulties with VSAM or made an interesting discovery, you could receive a cash payment, a free subscription to any of our *Updates*, or a credit against any of Xephon's wide range of products and services, simply by telling us all about it. For a copy of our *Notes for Contributors*, which explains the terms and conditions under which we publish articles, please write to the editor, Fiona Hewitt, at any of the addresses shown on page 2, or e-mail her on 100336.1412@compuserve.com

VSAM news

Sterling Software has announced that its Vision:Solutions 2000 suite of tools, methodologies, education, and consulting services now includes Vision:Dager, a data ageing utility for VSAM datasets.

Vision:Dager, which also includes date windowing facilities, is designed to allow VSAM KSDS datasets with two-digit years in the key portion to sequence the dates correctly when the year 99 rolls over to 00. It supports all date formats and is claimed to eliminate the need to make time-consuming changes to application programs that use a VSAM-keyed dataset.

The windowing operation is transparent to programs, with no impact on applications except to provide records in the correct sequence.

For more information, contact:
Sterling Software, 1800 Alexander Bell Drive, Reston, VA 22091, USA.
Tel: (703) 264 8000.
Sterling Software Ltd, 75 London Road, Reading, Berks, RG1 5BS, UK.
Tel: (01734) 391139.
URL: www.sterling.com.

* * *

IBM has announced Release 2 of SmartBatch for OS/390, both an enhancement to and a replacement for BatchPipes/MVS.

The Data Accelerator performance component uses memory and CPU to improve the efficiency of application I/O requests to VSAM and non-VSAM datasets. It's designed to reduce the number of accesses to disk, and reduces or eliminates unnecessary wait times. The component provides both basic and advanced I/O performance processing, with the former automatically determining and setting the optimum buffering values for the utility or application.

At the advanced level, it optimizes I/O by using buffer adjustments and a combination of acquiring and managing its own buffers, decreasing I/O path lengths, optimizing channel programs, controlling overlapping I/O, providing larger data transfers per I/O operation, and learning an application's access patterns to restructure I/O requests. Enhancements are geared to provide improved VSAM optimization by maintaining a history of prior optimizations, improved diagnostics, and better Resource Usage Bias (RUB) selection.

For more information, contact your local IBM representative.

* * *