30

VSE update

June 1998

In this issue

VSE Update

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

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

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

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

Editor
Fiona Hewitt

Contributions
Articles published in *VSE 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*.

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

Date adjust subroutine

Following on from the convert date subroutine published in the last issue of *VSE Update*, we now present the DPADJ2 date adjust subroutine, which increments or decrements a passed MMDDCCYY date by a given number of days, as indicated by a passed value.

One parameter must be passed, consisting of five fields. A second parameter consisting of two fields is optional.

FIRST PARAMETER

First field

The first field is eight bytes long. If option ‘0’ is not selected, it contains the MMDDCCYY to be incremented or decremented. If option ‘0’ is selected, it contains the low/high date in determining the number of days between two dates. The date must be numeric, and must contain valid month and day numbers.

Second field

The second field is four bytes long.

- If option ‘0’ is not selected, it contains the number of days to be incremented or decremented. This must be a valid signed or unsigned positive or negative number in the range -9999 to +9999. This field may be passed with zeros if you require only the day of the week indicator of the passed MMDDCCYY, or if you simply want to determine whether a passed date is in the external date table.
- If option ‘0’ is selected, it contains the number of days between the dates specified in the first and fourth fields, in packed decimal format. Note that the value returned will be negative if the date specified in the first field is greater than that specified in the fourth field (see Figure 1).

Field 1	Field 4	Result hex
01011995	12311995	X'0000365F'
01011996	12311996	X'0000366F'
12311996	01011996	X'0000365D'
01011997	01011997	X'0000000C'
01011997	01021997	X'0000001F'

Figure 1: Example values

Third field

The third field is a one-byte option field. All options except for option '0' return the result of incrementing or decrementing the passed MMDDCCYY by the number of days specified in the next (fourth) field and a day of the week indicator in this field, where:

1	= Sunday
2	= Monday
3	= Tuesday
	.
	.
	.
7	= Saturday
9	= an error occurred.

If option '0' is not selected, the result date is or is not checked, depending on the following options:

- '0' – This option is used to calculate the number of days between the two dates passed in the first and fourth fields, in MMDDCCYY format. The result is returned in the second field, in packed format, up to a maximum of 9,999,999 days.
- '1' – If the result date is a Saturday or Sunday, it is advanced to the following Monday. The external date table is checked, and, if a match occurs, the result date is replaced by the corresponding table date.

- ‘2’ – The result date is not checked for Saturday or Sunday. The external date table is not checked for a match.
- ‘3’ – If the result date is a Saturday or Sunday, it is advanced to the following Monday. The external date table is not checked for a match.
- ‘4’ – If the result date is a Saturday but not a Sunday, it is advanced to the following Monday. The external date table is not checked for a match.
- ‘5’ – If the result date is a Sunday but not a Saturday, it is advanced to the following Monday. The external date table is not checked for a match.
- ‘6’ – The result date is not checked for a Saturday or Sunday. The external date table is checked, and, if a match occurs, the result date is replaced by the corresponding table date.
- ‘7’ – If the result date is a Saturday but not a Sunday, it is advanced to the following Monday. The external date table is checked, and, if a match occurs, the result date is replaced by the corresponding table date.
- ‘8’ – If the result date is a Sunday but not a Saturday, it is advanced to the following Monday. The external date table is checked, and, if a match occurs, the result date is replaced by the corresponding table date.

Fourth field

The fourth field is eight bytes long. If option ‘0’ is not selected, it contains the MMDDCCYY result of incrementing or decrementing the date passed in the first field. If option ‘0’ is selected, it contains the high/low date in determining the number of days between the date specified in the first field and that specified in this field.

Note that:

- If option ‘0’ is not selected and the first byte of this field contains an X'FF' (high-value), the day of the passed MMDDCCYY in the first field will be replaced by the last day of the month of that field before any incrementing or decrementing.

- If option ‘0’ is not selected and an error occurs, this field will contain blanks on return to the calling program.

Fifth field

The fifth field is one byte long. It contains one of the following values on return to the calling program:

- ‘0’ – All passed fields were correct and no informational messages or errors occurred. Options ‘0’ and ‘2’ always return this value, assuming return codes 6 to 9 weren’t issued.
- ‘1’ – Options ‘1’, ‘3’, ‘5’, or ‘8’ were passed and the result date was a Sunday which was changed to the following Monday.
- ‘2’ – Options ‘1’, ‘3’, ‘4’, or ‘7’ were passed and the result date was a Saturday which was changed to the following Monday.
- ‘3’ – Options ‘1’ or ‘8’ were passed and the result date was a Sunday which was changed to the following Monday. This was replaced because a match was found in the external date table.
- ‘4’ – Options ‘1’ or ‘7’ were passed and the result date was a Saturday which was changed to the following Monday. This was replaced because a match was found in the external date table.
- ‘5’ – Options ‘1’ or ‘6’ were passed and the result date was replaced because a match was found in the external date table.
- ‘6’ – Either the passed option field doesn’t contain the digits 1 to 8 or else the MMDDCCYY in the first field or the increment value in the second field were not numeric.
- ‘7’ – The month number (MM) portion of the first field was not in the range 01 to 12.
- ‘8’ – The day number (DD) portion of the first field was not valid for the given month number (ie the day was 30 and the month was 02, etc) or it was zero.

- ‘9’ – A replacement date, internal logic, or CDLOAD error occurred:
  - A replacement date error occurs when the date to be replaced is not greater than the matched date, or when the day of the week is to be calculated and the day number to be used is higher than that for the given month. This error generally indicates that the replacement date in the external table was incorrectly specified or that an internal logic error occurred.
  - A CDLOAD error occurs when an attempt to load phase DPADT2 (or the alternative) into the GETVIS partition and register 15 didn’t contain a zero return code. A partial storage dump is taken so that you can examine register 15 to resolve this problem.

SECOND PARAMETER

The optional second parameter, which is ignored if the value passed in the option byte (the third field) is ‘0’, contains two fields.

First field

The first field is one byte long, and contains the following on return to the calling program:

- ‘0’ – The day of the result date is not the last day of the month.
- ‘1’ – The day of the result date is the last day of the month.
- ‘2’ – The day of the result date was advanced past the end of the month because it was a Saturday, Sunday, or holiday, but only to the first day of the next month.
- ‘3’ – The day of the result date was advanced past the end of the month because it was a Saturday, Sunday, or holiday, and past the first day of the next month.

Second field

The second field is eight bytes long. It contains the name of the alternative date table to be used for checking (see above). If you don't wish to use an alternative date table, set this field to low-values or spaces; any other value is assumed to be a phase name residing in a LIBDEFed library, and must be in the same format as the default date table, DPADT2 (see below).

If the phase name is shorter than eight characters, it must be left-justified and padded on the right with blanks. If the phase is not found, the fifth field (the return code) is set to '9'.

This subroutine uses the DPDATE subroutine, which is CDLOADed by DPCALL.

CALLING SEQUENCES

The calling sequences are as follows.

COBOL

```
CALL 'DPADJ2' USING PARAM1.
```

Or

```
CALL 'DPADJ2' USING PARAM1, PARAM2.
```

ALC

```
LA    13,SAVEAREA (13 CAN ALSO BE R13 OR RD).
                  CALL  DPADJ2,(PARAM1)
```

Or

```
                  CALL  DPADJ2,(PARAM1,PARAM2)

                     .  (MAINLINE PART OF PROGRAM).
                     .
SAVEAREA DC    18F'Ø'
```

RPGII

```
     CALL 'DPADJ2'
                  PARM   PARAM1
```

Or

```
                  PARM   PARAM2
```

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

DPADJ2

```
DPAD     TITLE 'DPADJ2 - 1.Ø - DATE ADJUST SUBROUTINE.'
DPADJ2   CSECT Ø
DPADJ2   AMODE 31
DPADJ2   RMODE ANY
         BALR  RF,Ø                 LOAD TEMPORARY BASE.
         USING *,RF                 INFORM ASSEMBLER.
         SAVE  (14,12)
         DROP  RF                   DROP TEMPORARY BASE.
         BALR  R3,RØ
         USING *,R3
         ST    RD,SAVEAREA+4
         LA    RD,SAVEAREA
         B     ADJBEG               BRANCH TO ADJBEG.
*
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
RA       EQU   1Ø
RB       EQU   11
RC       EQU   12
RD       EQU   13
RE       EQU   14
RF       EQU   15
*
         DC    C'DPADJ2 STARTS HERE. ' INSERT EYE CATCHER.
*
```

```
ADJBEG   EQU   *
         L     R4,Ø(R1)            LOAD PASSED PARAMETER TO REG 4.
         MVC   INPDATE(L'SAVEPARM),Ø(R4) MVE IT TO INPDATE.
         MVC   SAVEPARM,INPDATE    MVE IT TO SAVEPARM.
         MVI   DAYWK,C'9'          SET DAY OF WEEK TO '9'.
         MVI   NUMPRM,X'ØØ'        SET NUMBER OF PARAMETERS TO ZERO.
         SR    R6,R6               SET PARAMETER COUNT TO ZERO.
*
ADJARG   EQU   *
         TM    Ø(R1),X'8Ø'         ARE WE DONE.
         BO    ADJLST              YES-BRANCH TO ADJLST.
         LA    R6,4(R6)            INCREMENT REG 6 BY ONE (1).
         LA    R1,4(R1)            INCREMENT REG 1 TO NEXT PARAMETER.
         B     ADJARG              BRANCH TO ADJARG.
*
ADJLST   EQU   *
         SR    R1,R6               RESTORE REG 1.
         SRL   R6,2                DIVIDE REG 6 BY 2.
         LA    R6,1(R6)            BUMP BY ONE FOR FIRST TIME.
         STC   R6,NUMPRM           SAVE NUMBER OF PARAMETERS PASSED.
         MVI   RCDE,C'9'           SET RETURN CODE TO '9'.
         CLI   NUMPRM,1            WAS ONE (1) PARAMETER PASSED.
         BE    DAYADJ              YES-BRANCH TO DAYADJ.
         CLI   NUMPRM,2            WERE TWO (2) PARAMETERS PASSED.
         BNE   RETURN              NO-BRANCH TO RETURN.
         L     R5,4(R1)            GET ADDRESS OF LDAYWK AND ALTERNATE
         MVC   LDAYWK(9),Ø(R5)     MVE IT TO SAVE AREA.
         MVI   LDAYWK,C'Ø'         ASSUME DAY IS NOT LAST FOR MONTH.
*
DAYADJ   EQU   *
         MVI   HOLD,C' '           CLEAR HOLD.
         MVI   RCDE,C'6'           SET RETURN CODE TO '6'.
         LA    RA,11               LOAD BRANCH COUNTER TO REG 1Ø.
         CLI   OPTN,C'Ø'           IS THIS OPTION ZERO.
         BNE   DAYADJ3             NO-BRANCH TO DAYADJ3.
         MVC   INCR,=C'ØØØØ'       SET INCREMENT TO ZERO.
         LA    RA,21               LOAD BRANCH COUNTER TO REG 1Ø.
         B     DAYADJ5             BRANCH TO DAYADJ5.
*
DAYADJ3  EQU   *
         CLI   OPTN,C'1'           IS OPTION LOWER THAN ONE (1).
         BL    RETURN              YES-BRANCH TO RETURN.
*
DAYADJ5  EQU   *
         CLI   OPTN,C'8'           IS OPTION HIGHER THAN EIGHT (8).
         BH    RETURN              YES-BRANCH TO RETURN.
         LA    RC,INPDATE          LOAD ADDRESS OF INPUT TO REG 12.
*
```

```
LOOP1    EQU   *
         CLI   Ø(RC),C'Ø'          IS INPUT NUMERIC.
         BL    RETURN              NO-BRANCH TO RETURN.
         CLI   Ø(RC),C'9'          ...
         BH    RETURN              ...
         LA    RC,1(RC)            INCREMENT REG 12 TO NEXT POSITION.
         BCT   RA,LOOP1            BRANCH TO LOOP1 UNTIL REG 1Ø ZERO.
         CLI   OPTN,C'Ø'           IS THIS OPTION ZERO.
         BE    LOOP1A              YES-BRANCH TO LOOP1A.
         CLI   INCR+3,X'CØ'        IS INPUT NUMERIC.
         BL    RETURN              NO-BRANCH TO RETURN.
         CLI   INCR+3,X'F9'        ...
         BH    RETURN              ...
         CLI   INCR+3,X'CA'
         BL    LOOP1A
         CLI   INCR+3,X'EF'
         BH    LOOP1A
         MVZ   INCR+3(1),=X'DØ'    ASSUME NEGATIVE.
*
LOOP1A   EQU   *
         PACK  DBL1,ICC(4)         PACK ICC AND IYY TO DBL1.
         PACK  DBL,IMM             PACK INPUT MONTH NUMBER.
         CVB   R6,DBL              CONVERT IT TO BINARY.
         PACK  DBL,IDD             PACK INPUT DAY NUMBER.
         CVB   R7,DBL              CONVERT IT TO BINARY.
         PACK  DBL,INCR            PACK INCREMENT DAYS.
         CVB   R8,DBL              CONVERT IT TO BINARY.
         PACK  DBL,IYY             PACK INPUT YEAR NUMBER.
         BAL   RC,SETLEAP          PERFORM SETLEAP ROUTINE.
         MVI   RCDE,C'7'           SET RETURN CODE TO '7'.
         SR    RA,RA               CLEAR REG 1Ø.
         CR    R6,RA               IS MONTH NUMBER LESS THAN ZERO.
         BL    RETURN              YES-BRANCH TO RETURN.
         LA    RA,12               LOAD MAXIMUM MONTH VALUE TO REG 1Ø.
         CR    R6,RA               IS MONTH NUMBER GREATER THAN 12.
         BH    RETURN              YES-BRANCH TO RETURN.
         LTR   R6,R6               IS MONTH NUMBER ZERO.
         BZ    RETURN              YES-BRANCH TO RETURN.
         SR    R9,R9               CLEAR REG 9.
         IC    R9,LIST-1(R6)
         MVI   RCDE,C'8'           SET RETURN CODE TO '8'.
         CR    R7,R9               IS DAY NUMBER VALID FOR MONTH.
         BH    RETURN              NO-BRANCH TO RETURN.
         LTR   R7,R7               IS DAY NUMBER ZERO.
         BZ    RETURN              YES-BRANCH TO RETURN.
         CLI   HOLD,C'Ø'           HAVE WE BEEN HERE BEFORE.
         BE    CALCDAYS            YES-BRANCH TO CALCDAYS.
         CLI   OPTN,C'Ø'           IS THIS OPTION ZERO.
         BNE   LOOP1B              NO-BRANCH TO LOOP1B.
         MVI   HOLD,C'Ø'           SET HOLD.
         MVC   INPDATE,OUTDATE     MVE OUTDATE TO INPDATE.
         B     LOOP1A              BRANCH TO LOOP1A.
*
LOOP1B   EQU   *
         CLI   OUTDATE,X'FF'       DO WE REPLACE DAY NUMBER TO LAST DAY
         BNE   LOOP1D              NO-BRANCH TO LOOP1D.
         LR    R7,R9               REPLACE DAY NUMBER WITH LAST DAY OF
         MVI   OUTDATE,X'FE'
*
LOOP1D   EQU   *
         AR    R7,R8               ADD/SUB INCREMENT DAYS TO/FROM DAY N
         BP    LOOP2C              PLUS-BRANCH TO LOOP2C.
         B     LOOP2B1             BRANCH TO LOOP2B1.
*
LOOP2B   EQU   *
         IC    R9,LIST-1(R6)
         AR    R7,R9               ADD MONTH OVERFLOW TO DAY NUMBER.
         BP    DONE3               PLUS-BRANCH TO DONE3.
*
LOOP2B1  EQU   *
         BCTR  R6,Ø                DECREMENT MONTH NUMBER BY ONE (1).
         LTR   R6,R6               IS IT ZERO.
         BNZ   LOOP2B              NO-BRANCH TO LOOP2B.
         LA    R6,12               SET MONTH NUMBER TO DECEMBER.
         SP    DBL,=P'1'           SUB ONE (1) FROM YEAR NUMBER.
         SP    DBL1,=P'1'          SUB ONE (1) FROM CENTURY/YEAR NUMBER
         BAL   RC,SETLEAP          PERFORM SETLEAP ROUTINE.
         B     LOOP2B              BRANCH TO LOOP2B.
*
LOOP2C   EQU   *
         IC    R9,LIST-1(R6)
         SR    R7,R9               ADD MONTH OVERFLOW TO DAY NUMBER.
         BNP   DONE                NOT PLUS-BRANCH TO DONE.
         LA    R6,1(R6)            INCREMENT MONTH NUMBER BY ONE (1).
         CR    R6,RA               IS IT GREATER THAN 12.
         BNH   LOOP2C              NO-BRANCH TO LOOP2C.
         SR    R6,RA               SET MONTH NUMBER TO JANUARY.
         AP    DBL,=P'1'           ADD ONE (1) TO YEAR NUMBER.
         AP    DBL1,=P'1'          ADD ONE (1) TO CENTURY/YEAR NUMBER.
         BAL   RC,SETLEAP          PERFORM SETLEAP ROUTINE.
         B     LOOP2C              BRANCH TO LOOP2C.
*
DONE     EQU   *
         AR    R7,R9               ADD IT BACK.
*
DONE3    EQU   *
```

```
         UNPK  OUTCC,DBL1+5(3)      UNPACK NEW CENTURY/YEAR NUMBER.
         UNPK  OUTDATE(L'OUTDATE-3),DBL+6(2) UNPACK NEW YEAR NUMBER.
         MVC   OYY,OUTDATE+1        MVE IT TO OUTPUT.
         CVD   R6,DBL               CONVERT NEW MONTH NUMBER TO DECIMAL.
         UNPK  OMM,DBL              UNPACK IT TO OUTPUT.
         CVD   R7,DBL               CONVERT NEW DAY NUMBER TO DECIMAL.
         UNPK  ODD,DBL              UNPACK IT TO OUTPUT.
         MVC   OCCYY,OUTCC+1        MVE IT TO OUTPUT.
         MVZ   OUTDATE,=C'ØØØØØØØØ' MAKE IT ALL NUMERIC.
         BAL   RC,GETDOW            PERFORM GETDOW ROUTINE.
* *********************************************************************
* AT THIS POINT WE'VE ADJUSTED THE DATE. WE MUST NOW CHECK WHETHER  *
* SATURDAY, SUNDAY, OR A HOLIDAY AND RECALCULATE IF REQUESTED.      *
* *******************************************************************
         CLI   OPTN,C'6'            IS THIS OPTION SIX (6).
         BE    LOOP3                YES-BRANCH TO LOOP3.
         CLI   OPTN,C'2'            IS THIS OPTION TWO (2).
         BNE   *+12                 NO-SKIP NEXT TWO (2) INST.
         MVI   RCDE,C'Ø'            SET RETURN CODE TO 'Ø'.
         B     RETURN               BRANCH TO RETURN.
         CLI   OPTN,C'7'            IS THIS OPTION SEVEN (7).
         BE    *+2Ø                 YES-SKIP NEXT FOUR (4) INST.
         CLI   OPTN,C'4'            IS THIS OPTION FOUR (4).
         BE    *+12                 YES-SKIP NEXT TWO (2) INST.
         CLI   DAYWK,C'1'           IS DAY OF WEEK SUNDAY.
         BE    SUN                  YES-BRANCH TO SUN.
         CLI   OPTN,C'8'            IS THIS OPTION EIGHT (8).
         BE    *+2Ø                 YES-SKIP NEXT FOUR (4) INST.
         CLI   OPTN,C'5'            IS THIS OPTION FIVE (5).
         BE    *+12                 YES-SKIP NEXT TWO (2) INST.
         CLI   DAYWK,C'7'           IS DAY OF WEEK SATURDAY.
         BE    SAT                  YES-BRANCH TO SAT.
         CLI   OPTN,C'1'            IS THIS OPTION ONE (1).
         BE    LOOP3                YES-BRANCH TO LOOP3.
         CLI   OPTN,C'6'            IS THIS OPTION SEVEN (7) OR EIGHT (8
         BNH   LOOP4A               NO-BRANCH TO LOOP4A.
*
LOOP3    EQU   *
         LA    RB,DPADT2            LOAD ADDRESS OF DEFAULT PHASE TO REG
         CLI   ALTPHASE,X'41'       WAS ALTERNATE PHASE SPECIFIED.
         BL    *+8                  NO-SKIP NEXT INST.
         LA    RB,ALTPHASE          LOAD ADDRESS OF ALTERNATE PHASE TO R
         CDLOAD (RB),RETPNF=YES     GO CDLOAD PHASE.
         LTR   RF,RF                WAS CDLOAD SUCCESSFUL.
         BNZ   CDLOADE              NO-BRANCH TO CDLOADE.
         LR    R9,R1                LOAD ENTRY POINT TO REG 9.
*
LOOP4    EQU   *
```

```
         CLC   OUTDATE,Ø(R9)       IS DATE IN TABLE.
         BE    FOUND               YES-BRANCH TO FOUND.
         LA    R9,16(R9)           INCREMENT REG 9 TO NEXT TABLE POSITI
         CLI   Ø(R9),X'FF'         ARE WE AT THE END OF THE TABLE.
         BNE   LOOP4               NO-BRANCH TO LOOP4.
*
LOOP4A   EQU   *
         MVI   RCDE,C'Ø'           SET RETURN CODE TO 'Ø'.
         CLI   HOLD,C' '           DID WE REPLACE SATURDAY/SUNDAY.
         BE    RETURN              NO-BRANCH TO RETURN.
         MVC   RCDE,HOLD           SET RETURN CODE TO SATURDAY/SUNDAY.
         B     RETURN              BRANCH TO RETURN.
*
FOUND    EQU   *
         CLC   14(L'OCCYY,R9),OCC  IS REPLACEMENT CENTURY/YEAR LOWER TH
         BL    FOUND3              YES-BRANCH TO FOUND3. (ERROR).
         BH    FOUND5              HIGH-BRANCH TO FOUND5. (CCYY HIGHER)
         CLC   8(L'OMMDD,R9),OMMDD IS REPLACEMENT MMDD LOWER THAN ARGUM
         BNL   FOUND5              NO-BRANCH TO FOUND5.
*
FOUND3   EQU   *
         MVI   RCDE,C'9'           SET RETURN CODE TO '9'.
         B     RETURN              BRANCH TO RETURN.
*
FOUND5   EQU   *
         MVC   OUTDATE,8(R9)       REPLACE OUTPUT DATE WITH NEW TABLE D
         PACK  DBL,OMM             PACK OUTPUT MONTH NUMBER.
         CVB   R6,DBL              CONVERT IT TO BINARY.
         PACK  DBL,ODD             PACK OUTPUT DAY NUMBER.
         CVB   R7,DBL              CONVERT IT TO BINARY.
         PACK  DBL,OYY             PACK OUTPUT YEAR NUMBER.
         PACK  DBL1,OCCYY          PACK OUTPUT CENTURY/YEAR NUMBER.
         BAL   RC,SETLEAP          PERFORM SETLEAP ROUTINE.
         BAL   RC,GETDOW           PERFORM GETDOW ROUTINE.
         CLI   HOLD,C'1'           DID WE REPLACE SUNDAYS DATE.
         BNE   *+12                NO-SKIP NEXT TWO (2) INST.
         MVI   RCDE,C'3'           SET RETURN CODE TO '3'.
         B     RETURN              BRANCH TO RETURN.
         CLI   HOLD,C'2'           DID WE REPLACE SATURDAYS DATE.
         BNE   *+12                NO-SKIP NEXT TWO (2) INST.
         MVI   RCDE,C'4'           SET RETURN CODE TO '4'.
         B     RETURN              BRANCH TO RETURN.
         MVI   RCDE,C'5'           SET RETURN CODE TO '5'.
*
RETURN   EQU   *
         MVC   OPTN,DAYWK          MVE DAY OF WEEK TO OPTION BYTE.
         MVC   INPDATE(12),SAVEPARM RESET INPUT FIELDS.
*
```

```
RETURN3  EQU   *
         CLI   RCDE,C'6'           WAS THERE AN ERROR.
         BL    RETURN4             NO-BRANCH TO RETURN4.
         MVC   OUTDATE,=8C' '      CLEAR OUTDATE.
         MVI   OPTN,C'9'           SET DAY OF WEEK TO '9'.
*
RETURN4  EQU   *
         MVC   Ø(L'SAVEPARM,R4),INPDATE
         CLI   NUMPRM,X'Ø2'        WERE TWO (2) PARAMETERS PASSED.
         BNE   RETURN5             NO-BRANCH TO RETURN5.
         MVC   Ø(L'LDAYWK+L'ALTPHASE,R5),LDAYWK
*
RETURN5  EQU   *
*        PDUMP DPADJ2S,DPADJ2E
         SR    RF,RF               CLEAR REG 15.
         ST    RF,SAVEAREA+16      STORE REG 15 TO SAVEAREA+16.
         L     RD,SAVEAREA+4
         RETURN (14,12)
*
CDLOADE  EQU   *
         LA    RA,2Ø               LOAD 2Ø TO REG 1Ø.
         CR    RF,RA               WAS FAILURE DUE TO PHASE NOT FOUND.
         BE    CDLOADE3            YES-BRANCH TO CDLOADE3.
         MVI   RCDE,C'9'           SET RETURN CODE TO '9'.
         B     RETURN              BRANCH TO RETURN.
*
CDLOADE3 EQU   *
         CLI   ALTPHASE,X'41'      WAS ALTERNATE PHASE SPECIFIED.
         BL    LOOP4A              NO-BRANCH TO LOOP4A.
         MVI   RCDE,C'9'           SET RETURN CODE TO '9'.
         B     RETURN              BRANCH TO RETURN.
*
SUN      EQU   *
         MVC   INCR,=C'ØØØ1'       SET INCREMENT DAYS TO ØØØ1.
         MVI   HOLD,C'1'           SET HOLD TO '1'.
*
SUN3     EQU   *
         MVC   INPDATE,OUTDATE     MVE OUTDATE TO INPDATE.
         B     DAYADJ3             BRANCH TO DAYADJ3.
*
SAT      EQU   *
         MVC   INCR,=C'ØØØ2'       SET INCREMENT DAYS TO ØØØ2.
         MVI   HOLD,C'2'           SET HOLD TO '2'.
         B     SUN3                BRANCH TO SUN3.
*
CALCDAYS EQU   *
         MVI   NUMPRM,1
         MVC   INPDATE(L'SAVEPARM),SAVEPARM RESET INPUT PARAMETERS.
         MVI   RCDE,C'Ø'           SET RETURN CODE TO 'Ø'.
         MVI   DATEWRK,C'2'        INDICATE USER DATE IS TO BE CONVERTE
         MVC   DATEWRK+1(8),INPDATE MVE DATE1.
         MVI   OPTION,C'8'         INDICATE CCYYDDD CONVERSION.
         LA    RD,SAVEAREB         LOAD ADDRESS OF SAVEAREB TO REG 13.
*        CALL  DPDATE,(DATEWRK,OPTION) GO CONVERT TO CCYYDDD FORMAT.
         CALL  DPCALL,(SUBNME,DATEWRK,OPTION) GO CONVERT TO CCYYDDD FOR
         PACK  DATE1P,DATEWRK(7)   PACK DATE1.
         MVI   DATEWRK,C'2'        INDICATE USER DATE IS TO BE CONVERTE
         MVC   DATEWRK+1(8),OUTDATE MVE DATE2.
         LA    RD,SAVEAREB         LOAD ADDRESS OF SAVEAREB TO REG 13.
*        CALL  DPDATE,(DATEWRK,OPTION) GO CONVERT TO CCYYDDD FORMAT.
         CALL  DPCALL,(SUBNME,DATEWRK,OPTION) GO CONVERT TO CCYYDDD FOR
         PACK  DATE2P,DATEWRK(7)   PACK DATE2.
         ZAP   INCR,SIGN           SET RESULT TO ZERO.
         CP    DATE1P,DATE2P       ARE DATE1 AND DATE2 EQUAL.
         BE    SUBTDAYS            YES-BRANCH TO SUBTDAYS.
         BL    DATESOK             LOW-BRANCH TO DATESOK.
         MVI   SIGN,X'ØD'          HIGH-DATE DIFFERENCE WILL BE NEGATIV
         XC    DATE1P,DATE2P       SWAP DATES
         XC    DATE2P,DATE1P         SO DATE1 IS
         XC    DATE1P,DATE2P           LOWER THAN DATE2.
*
DATESOK  EQU   *
         CLC   DATE1P(2),DATE2P    ARE BOTH DATES FOR SAME CENTURY/YEAR
         BE    SUBTDAYS            YES-BRANCH TO SUBTDAYS.
         AP    INCR,=P'365'        ADD 365 DAYS TO RESULT.
         BO    OVRFLOW
         MVC   SVDBL,DBL
         MVC   SVDBL1,DBL1
         MVC   SVLIST,LIST
         UNPK  DATEWRK7,DATE1P
         PACK  DBL,DATEWRK7+2(2)
         PACK  DBL1,DATEWRK7(4)
         BAL   RC,SETLEAP
         MVC   DBL,SVDBL
         MVC   DBL1,SVDBL1
         CLI   LIST+1,29
         MVC   LIST,SVLIST
         BNE   DATESOK3
         AP    INCR,=P'1'          ADD ONE (1) TO RESULT.
         BO    OVRFLOW
*
DATESOK3 EQU   *
         AP    DATE1P,=P'ØØØ1ØØØ'  ADD ONE (1) TO DATE1P CENTURY/YEAR.
         BO    OVRFLOW
         B     DATESOK             BRANCH TO DATESOK.
*
```

```
SUBTDAYS EQU   *
         AP    INCR,DATE2P+2(2)    ADD DATE2 DAYS TO RESULT.
         BO    OVRFLOW             OVERFLOW-BRANCH TO OVRFLOW.
         SP    INCR,DATE1P+2(2)    SUBTRACT DATE1 DAYS.
         MVN   INCR+3(1),SIGN      SET CORRECT SIGN.
         MVI   SIGN,X'ØF'          RESTORE SIGN TO PLUS.
         B     RETURN3             BRANCH TO RETURN3.
*
OVRFLOW  EQU   *
         MVI   RCDE,C'9'           INDICATE DECIMAL OVERFLOW.
         B     RETURN3             BRANCH TO RETURN3.
*
SETLEAP  EQU   *             DETERMINE/SET LEAP YEAR ROUTINE.
         MVC   LIST(12),LISTINT
         CVB   R2,DBL              CONVERT YEAR TO BINARY.
         LTR   R2,R2               IS YEAR ZERO.
         BZ    SETLEAP5            YES-BRANCH TO SETLEAP5.
         LR    RØ,R2               LOAD IT TO REG 2.
         XR    R1,R1               CLEAR REG 1.
         SRDL  RØ,2                SHIFT IT RIGHT 2 POSITIONS (IE DIVI
         LTR   R1,R1               IS IT ZERO.
         BNZ   SETLEAP5            NO-BRANCH TO SETLEAP5.
         MVI   LIST+1,29           MAKE IT A LEAR YEAR.
         BR    RC                  RETURN TO CALLER.
*
SETLEAP5 EQU   *
         MVC   DBL2,DBL1           MVE DBL1 TO DBL2.
         DP    FUL2,=P'4ØØ'        DIVIDE ICC/IYY BY 4ØØ.
         CP    FUL2+2(2),=P'ØØ'    IS THERE A REMAINDER.
         BNER  RC                  YES-RETURN TO CALLER.
         MVI   LIST+1,29           MAKE IT A LEAR YEAR.
         BR    RC                  RETURN TO CALLER.
*
GETDOW   EQU   *              GET DAY OF WEEK ROUTINE.
         SR    R9,R9               CLEAR REG 9.
         IC    R9,LIST-1(R6)
         CR    R7,R9               IS DAY NUMBER VALID FOR MONTH.
         BH    GETDOW9             NO-BRANCH TO GETDOW9.
         BNE   GETDOW3             NOT EQUAL-BRANCH TO GETDOW3.
         MVI   LDAYWK,C'1'         INDICATE DAY IS LAST DAY OF MONTH.
*
GETDOW3  EQU   *
         CLC   SAVEPARM(2),OUTDATE DID WE ADVANCE TO NEXT MONTH.
         BE    GETDOW5             NO-BRANCH TO GETDOW5.
         MVI   LDAYWK,C'2'         INDICATE WE PAST LAST DAY OF THE MON
         CLC   OUTDATE+2(2),=C'Ø2' IS NEW DAY OF MONTH LOWER THAN Ø2.
         BL    GETDOW5             YES-BRANCH TO GETDOW5.
         MVI   LDAYWK,C'3'         INDICATE WE PAST LAST DAY OF THE MON
```

```
*
GETDOW5  EQU   *
         MVI   USRDTE,C'2'
         MVC   USRDTE+1(L'OUTDATE),OUTDATE MVE OUTPUT DATE TO USER DATE
         MVI   OPTION,C'5'          INDICATE LONG FORMAT CONVERSION.
         LA    RD,SAVEAREB          LOAD ADDRESS OF SAVEAREB TO REG 13.
*        CALL  DPDATE,(USRDTE,OPTION) GO GET DAY OF WEEK.
         CALL  DPCALL,(SUBNME,USRDTE,OPTION) GO GET DAY OF WEEK.
         LA    RD,DYOFWKTB          LOAD ADDRESS OF DYOFWKTB TO REG 13.
         LA    RE,3                 LOAD INCREMENT VALUE TO REG 14.
         LA    RF,DYOFWKTB+L'DYOFWKTB-1 LOAD END ADDRESS OF DAYTB1 TO R
*
GETDOW6  EQU   *
         CLC   USRDTE(2),Ø(RD)      IS IT IN THE TABLE.
         BE    GETDOW7              YES-BRANCH TO GETDOW7.
         BXLE  RD,RE,GETDOW6        BRANCH TO GETDOW6 UNTIL DONE.
         B     GETDOW9              BRANCH TO GETDOW9. (DAY OF WEEK ERRO
*
GETDOW7  EQU   *
         MVC   DAYWK,2(RD)          SET DAY OF WEEK CODE.
         BR    RC                   RETURN TO CALLER.
*
GETDOW9  EQU   *
         MVI   RCDE,C'9'            INDICATE DAY OF WEEK ERROR.
         B     RETURN               BRANCH TO RETURN.
*
DPADJ2S  DC    C'DPADJ2 STORAGE HERE. ' INSERT EYE CATCHER.
*
LIST     DS    CL12
SVLIST   DS    CL12
DBL      DS    D                    WORKAREA.
SVDBL    DS    D                    WORKAREA.
DBL1     DS    D                    WORKAREA.
SVDBL1   DS    D                    WORKAREA.
DBL2     DS    ØD                   WORKAREA.
         DS    F
FUL2     DS    F
DAYWK    DS    C
LDAYWK   DS    C                    DAY IS LAST DAY OF MONTH INDICATOR.
ALTPHASE DC    CL8' '               ALTERNATE DATE TABLE.
HOLD     DC    C' '                 HOLD.
USRDTE   DC    CL29' '
OPTION   DC    C'5'
SAVEPARM DS    CL22
INPDATE  DS    ØCL8
IMM      DS    CL2
IDD      DS    CL2
ICC      DS    CL2
```

```
IYY      DS    CL2
INCR     DS    CL4
OPTN     DS    C                   OPTION.
OUTDATE  DS    ØCL8
OMMDD    DS    ØCL4
OMM      DS    CL2
ODD      DS    CL2
OCCYY    DS    ØCL4
OCC      DS    CL2
OYY      DS    CL2
RCDE     DS    C                   RETURN CODE.
OUTCC    DS    CL5
NUMPRM   DC    X'ØØ'               NUMBER OF PARAMETERS PASSED SAVE ARE
SUBNME   DC    C'DPDATE  '
DPADT2   DC    C'DPADT2  '
DATEWRK  DS    CL9
DATEWRK7 DS    CL7
DATE1P   DS    PL4                 WORK AREA FOR LOW/HIGH DATE. (CCYYDD
DATE2P   DS    PL4                 WORK AREA FOR HIGH/LOW DATE. (CCYYDD
SIGN     DC    X'ØF'               USED FOR SIGN IF DATE1 HIGHER THAN
*
LISTINT  DC    AL1(31,28,31,3Ø,31,3Ø,31,31,3Ø,31,3Ø,31)
DYOFWKTB DC    C'SU1MO2TU3WE4TH5FR6SA7'
*
SAVEAREA DC    18F'Ø'
SAVEAREB DC    18F'Ø'
*
DPADJ2E  DC    X'FF'
*
         END
```

EXTERNAL DATE TABLE PROGRAM

The following program contains a list of dates used by DPADJ2 (see above). The date in the first eight bytes is compared against an argument date; if a match is found, the date in the second eight bytes replaces the argument date. Note that the replacement date (the second eight bytes) cannot be lower than the date in the first eight bytes.

Although the date entries need not be in any particular sequence, you may find it helpful to add them that way.

This program is CDLOADed by DPADJ2.

NOTES

1 If this table is CDLOADed by programs in long-running jobs such as CICS, the job must be terminated and restarted before the new dates will be recognized.

2 The dates used should be periodically removed once the actual date is past.

3 Although there is no limit to the number of date entries that can be added to the table, it makes sense to remove old entries when new ones are added.

DPADT2

```
DPAD     TITLE 'DPADT2 - 1.Ø - EXTERNAL DATE TABLE PROGRAM.'
DPADT2   CSECT
DPADT2   AMODE 31
DPADT2   RMODE ANY
*
         DC    C'Ø1171994',C'Ø1181994'
         DC    C'Ø2211994',C'Ø2221994'
         DC    C'Ø4Ø11994',C'Ø4Ø41994'
         DC    C'Ø53Ø1994',C'Ø5311994'
         DC    C'Ø7Ø41994',C'Ø7Ø51994'
         DC    C'Ø9Ø51994',C'Ø9Ø61994'
         DC    C'11111994',C'11141994'
         DC    C'11241994',C'11281994'
         DC    C'11251994',C'11281994'
         DC    C'12231994',C'12271994'
         DC    C'12261994',C'12271994'
         DC    C'Ø1Ø21995',C'Ø1Ø31995'
         DC    C'Ø1161995',C'Ø1171995'
         DC    C'Ø22Ø1995',C'Ø2211995'
         DC    C'Ø4141995',C'Ø4171995'
         DC    C'Ø5291995',C'Ø53Ø1995'
         DC    C'Ø7Ø41995',C'Ø7Ø51995'
         DC    C'Ø9Ø41995',C'Ø9Ø51995'
         DC    C'111Ø1995',C'11131995'
         DC    C'11231995',C'11271995'
         DC    C'11241995',C'11271995'
         DC    C'12251995',C'12271995'
         DC    C'12261995',C'12271995'
         DC    C'Ø1Ø11996',C'Ø1Ø21996'
         DC    C'Ø1151996',C'Ø1161996'
         DC    C'Ø2191996',C'Ø22Ø1996'
         DC    C'Ø4Ø51996',C'Ø4Ø81996'
         DC    C'Ø5271996',C'Ø5281996'
         DC    C'Ø7Ø41996',C'Ø7Ø51996'
         DC    C'Ø9Ø21996',C'Ø9Ø31996'
         DC    C'11111996',C'11121996'
         DC    C'11281996',C'12Ø21996'
         DC    C'11291996',C'12Ø21996'
         DC    C'12241996',C'12261996'
         DC    C'12251996',C'12261996'
         DC    C'Ø1Ø11997',C'Ø1Ø21997'
         DC    C'Ø12Ø1997',C'Ø1211997'
         DC    C'Ø2171997',C'Ø2181997'
         DC    C'Ø3281997',C'Ø3311997'
         DC    C'Ø5261997',C'Ø5271997'
         DC    C'Ø7Ø41997',C'Ø7Ø71997'
         DC    C'Ø9Ø11997',C'Ø9Ø21997'
         DC    C'11111997',C'11121997'
         DC    C'11271997',C'12Ø11997'
         DC    C'11281997',C'12Ø11997'
         DC    C'12251997',C'12291997'
         DC    C'12261997',C'12291997'
         DC    X'FF'              END OF TABLE. DO NOT REMOVE.
*
         END
```

Robert Botsis
Senior Systems Programmer (USA) © Xephon 1998

Extended ARIDBS utility

The utility presented here permits the extended use of IBM's ARIDBS. Because it is used without the SQLID password, the security of the SQL/DS environment can be improved without any risk of password violation. All DDL involved with DATA BASE can be stored without having to worry how to keep passwords secret, and can be easily updated at any time with no impact.

XARIDBS

```
* $$ JOB JNM=XARIDBS,CLASS=Ø
  * $$ PUN DISP=I,CLASS=Ø
  //   JOB XARIDBS  *  PGMNAME=XARIDBS *
  /* -------------------------------------------------------------*/
  /* NOTE1: THIS PROGRAM MUST BE CATALOGUED INTO EACH DATABASE     */
  /*        BEFORE BEING USED.                                     */
  /* NOTE2: KEEP THIS PROGRAM SECRET AS THE SQLDBA PASSWORDS ARE   */
            SHOWN.                                                 */
  /* NOTE3: FILL THE SQLDBA PASSWORD AND DBNAME BELOW THE TARGET   */
            DATABASE                                               */
  /* -------------------------------------------------------------*/
  //  OPTION NODUMP
  //  ON     $ABEND  GOTO ERRO
  //  ON     $CANCEL GOTO ERRO
  //  EXEC   COMPUTIL
  // JOB    XARIDBS * PGMNAME=ARIDBS *

  // ON     $ABEND  GOTO ERRO
  // OPTION NODUMP
  // ON     $CANCEL GOTO ERRO
  // LIBDEF PHASE,CATALOG=PRD2.SQL34Ø
  // LIBDEF *,SEARCH=PRD2.SQL34Ø
  // OPTION CATAL
     PHASE  XARIDBS,*
  * ***** ASSEMBLY/XARIDBS ***** COMPILER *
  // EXEC ASSEMBLY
  /*
  * ***** ARIPRPA/XARIDBS  ***** PREPROCESSOR SQL *
  // LIBDEF *,SEARCH=PRD2.SQL34Ø
  // EXEC ARIPRPA,SIZE=AUTO,PARM='USERID=SQLDBA/??????,
      PREP=XARIDBS,    *
                 DBNAME=???????'
           PRINT NOGEN
  XARIDBS  EQU   *
           STM   R14,R12,D12(R13)   SAVE REGISTERS
           BALR  R7,Ø               LOAD BASE REGISTER
           USING *,R7,R11,R12       ESTABLISH ADDRESSABILITY
  *-------------------------------------------------------------------*
  *        SET UP SAVE AREA POINTERS                                  *
  *-------------------------------------- ----------------------------*
           ST    R13,SAVEØ+D4       STORE BACKWARD POINTER TO
                                     SAVEAREA
           LA    R9,SAVEØ           R9:=ADDR(NEW SAVE AREA)
           ST    R9,D8(R13)         STORE FORWARD POINTER TO SAVEAREA
           LR    R13,R9             R13:=ADDR(NEW SAVE AREA)
           LR    R8,R14
           B     INITIAL            BRANCH AROUND CONSTANTS
```

```
*------------------------------------------------------------------*
*        DECLARE HOST VARIABLES                                    *
*------------------------------------------------------------------*
         EXEC  SQL BEGIN DECLARE SECTION
         DS    ØF
DBAID    DC    CL8'SQLPWUSR'      DBA USER ID
PASSW    DC    CL8' '             PASSWORD
SQLID    DC    CL8' '             REQUESTED USERID
HPASSW   DC    CL8' '             PW WILL DROP HERE
         EXEC  SQL END DECLARE SECTION
*------------------------------------------------------------------*
INITIAL  EQU   *
*------------------------------------------------------------------*
WHATVSE  EQU   *                  IS THIS CPU ALLOWED TO RUN
                                   XARIDBS ?
*------------------------------------------------------------------*
         SR    Ø,Ø
         LA    1,=CL8'$$BSUPST'
         SVC   2                   ENTER IN SUPERVISOR STATE
         STIDP CPUID               STOR CPUID
         LA    Ø,1
         LA    1,=CL8'$$BSUPST'
         SVC   2                   ENTER IN PROBLEM STATE
         LA    6,CPUID             MAP CPUID INFO
         USING LAYOUT,6
VSEPROD  CLC   VSEID,=X'ØØØØØØ'    THIS PROGRAM...
         BE    VSEOK               MAY...
VSETESTE CLC   VSEID,=X'111111'    RUN...
         BE    VSEOK               ONLY...
VSESUPT  CLC   VSEID,=X'222222'    IN...
         BE    VSEOK               THESE MACHINES.
         B     VSEWRONG            IF NOT, WARN...
VSEOK    EQU   *

*------------------------------------------------------------------*
         MVC   PASSW,THEPW        MOVE THE PW
*------------------------------------------------------------------*
PROCESS  EQU   *
         MVC   PARM,2(R4)         MOVE SID= TO PARM
         CLC   PARM,SID           COMPARE FORMAT
         BNE   ERROSID            IF WRONG, WARN
         MVC   SQLID,6(R4)        MOVE SQL-ID TO BE CONNECTED
         OC    SQLID,=X'4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø' MOVE BLANKS TO BINARIES
*------------------------------------------------------------------*
         LA    R11,XFFF(R7)       LOAD BASE REGISTER
         LA    R11,D1(R11)        LOAD BASE REGISTER
         LA    R12,XFFF(R11)      LOAD BASE REGISTER
         LA    R12,D1(R12)        LOAD BASE REGISTER
```

```
*-------------------------------------------------------------------*
*        GET VIRTUAL STORAGE FOR SQL/DS AND INITIALIZE IT TO ZERO *
*-------------------------------------------------------------------*
         L     RØ,SQLDSIZ         RØ:=LENGTH OF DSECT FOR DS2
         GETVIS ADDRESS=(1),LENGTH=(Ø)
         LTR   R15,R15            TEST IF RETURN CODE
         BZ    GO                 NO RETURN CODE
         B     GETVISER           TERMINATE
GO       EQU   *
         LR    R6,R1              R6:=ADDR(DMSFREE AREA)
         USING SQLDSECT,R6        ESTABLISH ADDRESSABILITY
         LR    R4,RØ              R4:=LENGH OF DMSFREE SPACE
LOOP     EQU   *
         MVC   DØ(D8,R1),=8XL1'ØØ' CLEAR THE AREA
         LA    R1,D8(R1)          INCREMENT R1
         BCT   R4,LOOP            LOOP
*-------------------------------------------------------------------*
* PROGRAM WILL IGNORE WARNINGS SINCE THEY WILL NOT AFFECT RESULTS *
*-------------------------------------------------------------------*
         EXEC  SQL WHENEVER SQLWARNING CONTINUE
         EXEC  SQL WHENEVER SQLERROR   GOTO SQLERR
*-------------------------------------------------------------------*
CONNECT  EQU   *                  CONECT WITH THE WILD KEY
         EXEC  SQL CONNECT :DBAID IDENTIFIED BY :PASSW
LOCUSER  EQU   *
         EXEC  SQL DECLARE C1 CURSOR FOR                            *
                   SELECT  PASSWORD FROM SYSTEM.SYSUSERAUTH         *
                   WHERE   NAME = :SQLID
OPENC1   EQU   *
         EXEC  SQL OPEN C1
FETCHTAB EQU   *
         EXEC  SQL FETCH C1 INTO :HPASSW
         CLC   SQLCODE,FD1ØØ      TEST IF END OF DATA
         BE    IDNOTFND           ID NOT FOUND
*-------------------------------------------------------------------*
OK       EQU   *
         MVC   SAVCODE,=F'Ø'
COMMIT   EQU   *
         EXEC  SQL COMMIT WORK
*-------------------------------------------------------------------*
CONNECTA EQU   *                  CONNECT AS REQUESTED
         EXEC  SQL CONNECT :SQLID IDENTIFIED BY :HPASSW
*-------------------------------------------------------------------*
         L     RØ,DSIZE
         LR    R1,R6
         LA    R3,MSGØ1           LOAD MSGØ1 ADDR

         MVC   21(8,R3),SQLID     MOVE SQL-ID TO OUT
         MVC   ACNM,MSGØ1         MOVE MSG TO OUT
         PUT   CONSM              DISPLAY ON CONSOLE...
```

```
*        MVC   ACNM,HPASSW         ... PW GOT...
*        PUT   CONSM               ... FOR DEBUG ONLY
CALLARI  EQU   *
         CDLOAD IBMDBS             CALL IBM ARIDBS TO WORK
         LR     15,1
         CALL  (15)
EOJ      EQU   *
EOJERR   EOJ   RC=(15)             PASS RC ON RETURN TO VSE...
*-------------------------------------------------------------------*
* SUBROUTINES                                                       *
*-------------------------------------------------------------------*
ERROSID  EQU   *
         MVC   ACNM,MSGØ2          MOVE MSGØ2
         PUT   CONSM               DISPLAY ON CONSOLE
         B     CALLARI             CALL THAT GUY
*-------------------------------------------------------------------*
IDNOTFND EQU   *
         LA    R3,MSGØ3            LOAD MSGØ3 ADDR
         MVC   1Ø(8,R3),SQLID      MOVE SQL-ID TO OUT
         MVC   ACNM,MSGØ3          MOVE MSG TO OUT
         PUT   CONSM               DISPLAY ON CONSOLE
         MVC   ACNM,MSGØ7          MOVE MSG TO OUT
         PUT   CONSM               DISPLAY ON CONSOLE
         CANCEL ALL                CANCEL
*-------------------------------------------------------------------*
GETVISER EQU   *
         MVC   ACNM,MSGØ4          PUT MSG ON CONSOLE
         PUT   CONSM               DISPLAY ON CONSOLE
         B     EOJERR              PASS SQLCODE AS RC
*-------------------------------------------------------------------*
SQLERR   EQU   *
*-------------------------------------------------------------------*
         LA    R3,MSGØ5            LOAD MSGØ5 ADDR
         L     R1,SQLCODE          R1:= ERROR CODE
         CVD   R1,CVDFLD           CONVERT SQLCODE TO DECIMAL
         UNPK  MSG1ØB,CVDFLD       UNPACK SQLCODE FOR PRINT
         TM    MSG1ØB+L'MSG1ØB-D1,X1Ø  TEST IF NEGATIVE
         BZ    POSITIVE            POSITIVE ERROR CODE
         MVI   MSG1ØB,MINUS        MOVE MINUS SIGN BEFORE SQL CODE
POSITIVE OI    MSG1ØB+L'MSG1ØB-D1,XFØ  ERASE ORIGINAL SIGN
         MVC   2Ø(4,R3),MSG1ØB     MOVE MESSAGE TO OUTPUT AREA
*-------------------------------------------------------------------*
         MVC   ACNM,MSGØ5          PUT MSG ON CONSOLE
         PUT   CONSM               DISPLAY ON CONSOLE
         SR    R15,R15             ADJUST R15 TO ZERO
         L     R15,SQLCODE         LOAD SQLCODE ON R15
         B     EOJERR              PASS SQLCODE AS RC
*-------------------------------------------------------------------*
VSEWRONG EQU   *
         MVC   ACNM,MSGØ6
         PUT   CONSM              DISPLAY ON CONSOLE
         SR    R15,R15            ZERO R15
         L     R15,SQLCODE        LOAD SQLCODE ON R15
         B     EOJERR
*------------------------------------------------------------------*
* WORKING ...                                                      *
*------------------------------------------------------------------*

         DS    H
CVDFLD   DC    PL8'Ø'             FIELD USED FOR CVD INSTRUCTION
MSG1ØB   DS    CL4
MINUS    EQU   C'-'               CHAR
SAVEØ    DS    18F                SAVE AREA
SAVCODE  DS    F
SAVR14   DS    F
CPUID    DS    D
DSIZE    DS    F
THEPW    DC   X'FFFFFFFFFFFFFFFF' BIG PASSWORD
*------------------------------------------------------------------*
CONSM    DTFCN BLKSIZE=55,DEVADDR=SYSLOG,IOAREA1=ACNM,TYPEFLE=OUTPUT
ACNM     DC    CL55' '                  OUT AREA ON CONSOLE
PARM     DC    CL4' '                   WHAT IS ON // EXEC ?
SID      DC    CL4'SID='                PARAMETER FORMAT
USER     DC    CL8' '                   SQLID NAME
SAVE     DS    18F                      R13 SAVE AREA
*----------------123456789+123456789+123456789+123456789+123456789+12*
MSGØ1    DC CL55'..... CONNECT SQLID:                              '
MSGØ2    DC CL55'..... PARM SID=  NOT FOUND, CONNECT NOT DONE      '
MSGØ3    DC CL55'..... SID=         NOT FOUND, CONNECT NOT DONE    '
MSGØ4    DC CL55'..... GETVIS ERROR, ARIDBS NOT CALLED             '
MSGØ5    DC CL55'..... WARNING: DATABASE ERROR                     '
MSGØ6    DC CL55'..... WARNING: CPUID NOT ALLOWED                  '
MSGØ7    DC CL55'..... JOB CANCELLED                               '
*------------------------------------------------------------------*
*                      EQUATES AND CONSTANTS                       *
*------------------------------------------------------------------*
         EXEC SQL INCLUDE SQLCA
*------------------------------------------------------------------*
RØ       EQU   Ø                  REGISTER
R1       EQU   1                  REGISTER
R2       EQU   2                  REGISTER
R3       EQU   3                  REGISTER
R4       EQU   4                  REGISTER
R5       EQU   5                  REGISTER
R6       EQU   6                  REGISTER
R7       EQU   7                  REGISTER
R8       EQU   8                  REGISTER
R9       EQU   9                  REGISTER
R1Ø      EQU   1Ø                 REGISTER
R11      EQU   11                 REGISTER
```

```
R12      EQU   12                  REGISTER
R13      EQU   13                  REGISTER
R14      EQU   14                  REGISTER
R15      EQU   15                  REGISTER
*------------------------------------------------------------------*
DØ       EQU   Ø                   DISPLACEMENT
D1       EQU   1                   DISPLACEMENT
D2       EQU   2                   DISPLACEMENT
D3       EQU   3                   DISPLACEMENT
D4       EQU   4                   DISPLACEMENT
D5       EQU   5                   DISPLACEMENT
D8       EQU   8                   DISPLACEMENT
D12      EQU   12                  DISPLACEMENT
*------------------------------------------------------------------*
FØ       DC    F'Ø'                RETURN CODE
FD1ØØ    DC    A(1ØØ)              RETURN CODE
XFFF     EQU   X'FFF'              HEX NUMBER, MASK
X1Ø      EQU   X'1Ø'               HEX NUMBER, MASK

XFØ      EQU   X'FØ'               HEX NUMBER, MASK
*------------------------------------------------------------------*
LAYOUT   DSECT                     STIDP CPUID LAYOUT
FF       DS    XL1                 ALWAYS FF
VSEID    DS    XL3                 CPUID ON DIRECT
MODEL    DS    XL2                 CPU MODEL
REST     DS    XL2                 RESERVED
         END   XARIDBS
/*
IF $RC > 8 OR $RC = 8 THEN
// GOTO ERRO
// EXEC COMPUTIL
/*
IF $RC > 8 OR $RC = 8 THEN
// GOTO ERRO
* ***** LNKEDT/XARIDBS   ***** CATALOG *
// EXEC LNKEDT,PARM='MSHP'
// GOTO $EOJ
/. ERRO
* ERROR/ABEND...
/&
/*
// GOTO $EOJ
/. ERRO
* ERROR/ABEND...
// EXEC COMPUTIL
/*
/&
/*
/&
* $$ EOJ
```

$$BSUPST SUBROUTINE

This multipurpose subroutine will make things easier for those using VSE under VM.

```
* $$ JOB    JNM=$$BSUPST,CLASS=Ø
  //   JOB    $$BSUPST
  //   LIBDEF PHASE,CATALOG=IJSYSRS.SYSLIB
  //   OPTION CATAL,NODECK
       PHASE  $$BSUPST,S,NOAUTO,SVA,PBDY
  //   EXEC   ASSEMBLY,SIZE=128K
  $BSP     TITLE '$$BSUPST - TRANSIENT TO ENTER SUPERVISOR STATE'
  ***************************************************************
  *                                                             *
  * $$BSUPST                                                    *
  *                                                             *
  * FUNCTION: CHANGE FROM PROBLEM TO SUPERVISOR STATE OR        *
  *           VICE-VERSA                                        *
  *                                                             *
  * HOW TO USE:                                                 *
  *                                                             *
  *          SR    Ø,Ø    -OR-     LA    Ø,1                    *
  *          LA    1,=CL8'$$BSUPST'                             *
  *          SVC   2                                            *
  *                                                             *
  *      REG Ø HAS A CODE INDICATING THE OPERATION REQUESTED:   *
  *                                                             *
  *           Ø: FROM PROBLEM TO SUPERVISOR                     *
  *                                                             *
  *           1: FROM SUPERVISOR TO PROBLEM                     *
  *                                                             *
  * NOTE: IN CASE OF INVALID CODE OR IMPROPER CHANGE            *
  *       PROGRAM WILL BE CANCELLED.                            *
  *                                                             *
  ***************************************************************
           EJECT
           SPACE
           PRINT NOGEN
  $$BSUPST START Ø
           USING *,R15                R15 - BASE
  TRANSTRT DC    CL8'$$BSUPST'        TRANSIENT NAME
           B     COMECA
           DC    C'1'                 VERSION
           DC    C'Ø'                 LEVEL
           SPACE 2
  COMECA   L     R1,2Ø                R1 -> COMREG
           USING COMREG,R1            COMREG BASE
           LH    R2,PIBTAB            GET PIB TABLE ADDR
           L     R2,8(R2)             PARTITION SAVE AREA ADDR
           USING SAVEAREA,R2          PART.SAVE AREA - BASE
```

```
         LTR   RØ,RØ               PGM -> SUP ?
         BZ    PROGSUP             YES, GO TO PROPRER ROUTINE
         BCT   RØ,CANCEL           SUP -> PGM ? ABEND IF NOT AND
         OI    STATE,1             TURN ON PROBL/STATE ON PSW
         MVZ   STATE(1),X'2F'(1)   TURN ON PARTITION KEY ON PSW
         SVC   11                  RETURN TO USER PROGRAM
         EJECT
PROGSUP  DS    ØH
         NI    STATE,255-1         TURN ON SUPERV/STATE ON PSW
         NI    STATE,15            TURN ON KEY ZERO ON PSW
         SVC   11                  RETURN TO USER/PROGRAM
         SPACE 4
CANCEL   CANCEL ALL
         EJECT
COMREG   DSECT
         ORG   COMREG+X'5A'
PIBTAB   DS    H                   PIB TABLE ADDR
         SPACE 4
SAVEAREA DSECT
PGMNAME  DS    CL8                 PROGRAM NAME
OLDPSW   DS    CL8                 OLD PSW WHEN WAS INTERR.
STATE    EQU   OLDPSW+1            KEY + AMWP
REGS     EQU   *
*
*        REG   EQUATES
*
RØ       EQU   Ø                   REGISTER
R1       EQU   1                   REGISTER
R2       EQU   2                   REGISTER
R3       EQU   3                   REGISTER
R4       EQU   4                   REGISTER
R5       EQU   5                   REGISTER
R6       EQU   6                   REGISTER
R7       EQU   7                   REGISTER
R8       EQU   8                   REGISTER
R9       EQU   9                   REGISTER
R1Ø      EQU   1Ø                  REGISTER
R11      EQU   11                  REGISTER
R12      EQU   12                  REGISTER
R13      EQU   13                  REGISTER
R14      EQU   14                  REGISTER
R15      EQU   15                  REGISTER
         EJECT
         END
/*
// EXEC LNKEDT,PARM='MSHP'
// EXEC LNKEDT
/&
* $$ EOJ
```

INSTALLATION INSTRUCTIONS

1 Catalog XARIDBS main program.

2 Catalog $$BSUPST subroutine.

3 DBBASE must be started with parms below to permit SYSTEM.SYSUSERAUTH to be updated.

```
SQLSTART DB(DBPROD) PARM(PARMID=SQLSTARW,SERVAIDS=0000010,
LOGMODE=Y)
```

4 Create an SQLPWUSR SQLID with a dummy password as DBA user.

5 Use the DDL shown below to update the SQLPWUSR password. But first, use an editor to change '++++++++' to high-values (x'FF'), and change ??????? to the current password.

```
CONNECT SQLDBA IDENTIFIED  BY  ???????                 ; UPDATE
SYSTEM.SYSUSERAUTH SET PASSWORD =  '++++++++' WHERE   NAME =
'SQLPWUSR'                               ;
```

6 Shut down database and restart as usual

XARIDBS is now ready to be used – and you can say goodbye to SQLID passwords.

XARIDBS EXAMPLE JOB

```
* $$   JOB    JNM=XARITEST,CLASS=Ø
  //     JOB    XARITEST
  //     OPTION NODUMP
  //     EXEC   XARIDBS,SIZE=XARIDBS,PARM='SID=SQLDBA'
  SET    ERRORMODE CONTINUE;
  DROP   DBSPACE   ANYDBSPACE;
  SELECT * FROM SYSTEM.SYSPROGAUTH
         WHERE PROGNAME = 'XARIDBS';
  /*
  /*
  /&
  * $$ EOJ
```

Ricardo Quintal Reiche
Senior Systems Programmer (Brazil)

Transferring code from the Web to a mainframe

Editor's note: although this article was written by an MVS Update *subscriber using an ISPF edit macro, the same method can overcome a problem experienced 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:

```
IRXØØ13I 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'BØ' 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)

Macro processing when submitting files from CMS to VSE

OVERVIEW

The SUBVSE EXEC command for sending a CMS file to a VSE guest virtual machine is delivered with every VSE system and can be transferred to CMS. The command transfers the given file one-to-one to the specified VSE guest machine. If there are two or more VSE guest machines running under VM, there are usually different job files for the different VSE guest machines doing the same task. Because these job files often differ in a few strings, it is useful to hold one job file for all VSE guest machines and to replace the specifics of a VSE guest machine when submitting the job file. This is done by the J2VSE procedure. This procedure can also process POWER and VSE jobs, and can be extended to process other job files. J2VSE uses XEDIT with the MPP XEDIT macro, which is part of the J2VSE package.

A job can be submitted directly from XEDIT by issuing the J2VSE command which calls the J macro. J XEDIT is part of the J2VSE package.

SYNTAX

The CMS command:

```
J2VSE vse fn ft fm [partition] [values] ( options )
```

submits the job file fn ft fm to the VSE guest machine specified by VSE. The optional ‘partition’ parameter specifies the partition in which the job will be executed if the job file describes a VSE job. The optional ‘values’ parameter specifies the values of the arguments of the job file. If you choose the REPL option, an additional POWER job will be sent to the VSE guest machine, which will delete the POWER job with the same name from the POWER reader queue. With the HOLD option, the job to be sent will be browsed and not sent to the VSE guest machine. With the EDIT option, the job can be edited with

XEDIT and will be sent to the VSE guest machine when editing is complete. The record format may no longer be FIXED 80, and storing the job files with variable record format will save disk space.

A job file will be sent to the VSE guest machine only if all macros have been substituted. Otherwise, the macros which have not been expanded will be reported.

The XEDIT command:

```
J2VSE vse [partition] [values] ( options )
```

submits the job file currently edited to the VSE guest machine specified by vse. All parameters have the same meaning as described above. Macro J saves the actual file data into a temporary file which is submitted by executing the CMS command J2VSE.

MACRO PREPROCESSOR

The XEDIT-Makro MPP implements a macro preprocessor and will be called whenever a job file is to be sent to a VSE guest machine. It processes macro directives and replaces macros.

Macro directives are as shown in Figure 1.

The macro definitions can be specified in several ways; some macros (eg date) are defined by the system, others by the VSE guest machine the file will be submitted to. A macro definition has the following syntax:

%% DEFINE macrokey=text	define macro
%% UNDEF macrokey	undefine macro
%% INCLUDE fn ft fm	include specified file
%% IFDEF macrokey	if macro is defined (top of conditional processing)
%% IFNDEF macrokey	if macro is not defined (top of conditional processing)
%% ELSIFDEF macrokey	otherwise if macro is defined
%% ELSIFNDEF macrokey	otherwise if macro is not defined
%% ELSE	otherwise
%% ENDIF	end of conditional processing

Figure 1: Macro directives

```
macrokey=macrovalue
```

Because %macrokey% will be substituted by the macro value, neither the '%' character nor the '=' character are valid within the macrokey. %% cannot be used as a macro.

The following macros are defined by the parameter values within the command line:

1	value 1
2	value 2
3	value 3
. . .	further values

Figure 2 shows the macros defined by the system.

The macros defined by the VSE guest machine are specified in the file vse MPPDEF * (see Figure 3).

The value of a macro may be enclosed in quotes (see the WORK macro).

A log file can be generated containing all macros not expanded by the

```
USERID                      eg. CMS1
FNAME                       eg. PLOGDATE
FTYPE                       eg. JOB
THISDAY.DDMMYY              eg. 170396
THISDAY.DD/MM/YYYY          eg. 17/03/1996
THISDAY.DD.MM.YYYY          eg. 17.03.1996
THISMONTH.MM/YY             eg. 03/96
PREVMONTH.MM/YY             eg. 02/96
NEXTMONTH.MM/YY             eg. 04/96
TWOMONTHS.MMYY,MMYY         eg. 0396,0496
```

Figure 2: System-defined macros

```
TAPE181=600
TAPE182=602
SAP.SORT.VOLUME=VSE00C
SAVS50P1=VSE00D
VSE.NODE.ICCF=BC16
VSE.NODE.SAP=BC18
```

Figure 3: Macros defined by VSE guest machine

macro preprocessor. This log file will be displayed by the procedure J2VSE.

EXAMPLE

The following job file PLOGDATE JOB contains macro directives and macros:

```
* $$ JOB JNM=%FNAME%,CLASS=Ø,DISP=D
* $$ LST CLASS=R,DISP=K,DEST=(,ACCOUNT)
// JOB PRINTLOG ALLKOST=132ØØØØØ      PRINT HARDCOPY FILE
%% IFNDEF 1
%% DEFINE 1=%THISDAY.DD/MM/YYYY%
%% ENDIF
// EXEC PRINTLOG,PARM='%1%'
/*
/&
* $$ EOJ
```

The macro processing of the command

J2VSE VSEPROD 17/03/1996

sends the following data to the VSE guest machine VSEPROD:

```
* $$ JOB JNM=PLOGDATE,CLASS=Ø,DISP=D
* $$ LST CLASS=R,DISP=K,DEST=(,ACCOUNT)
// JOB PRINTLOG ALLKOST=132ØØØØØ      PRINT HARDCOPY FILE
// EXEC PRINTLOG,PARM='17/Ø3/1996'
/*
/&
* $$ EOJ
```

CONCLUSION

Using macro processing when submitting a CMS file to a VSE guest virtual machine has the following advantages:

- Maintainance of only one job file for many VSE guest machines.
- Data consistency between job file and VSE guest machine.
- Data consistency between different job files by using INCLUDE directives.
- Saving disk data space by using variable record format.

Since the only disadvantage is an increase in the execution time in CMS when submitting the file, you should always use macro processing when submitting a CMS file to a VSE guest virtual machine.

J2VSE EXEC

```
/* Datei:                J2VSE    EXEC          User:  VSE         */
/* Author:               Abstreiter, Franz      Date:  2Ø Aug 1997 */
/* Version: 1.ØØ                                                   */
/* (x) compiled by       Abstreiter, Franz             2Ø Aug 1997 */
/* (x) released by       Abstreiter, Franz             2Ø Aug 1997 */
/*                                                                 */
/* Call:                                                           */
/*   J2VSE machine fn ft fm partition values ( options )           */
/* Parameter:                                                      */
/*   machine   VSE guest virtual machine the job will be sent to   */
/*   partition Partition in which the job will be executed         */
/*   values    Values of the parameters of the job file            */
/* Options:                                                        */
/*   REPL      send an additional POWER job to the VSE guest       */
/*             machine for deleting a job from the POWER reader    */
/*             queue with the same name                            */
/*   HOLD      browses the job file and does not send it to        */
/*             the VSE guest machine                               */
/*   EDIT      opens the job file with XEDIT before sending it to  */
/*             the VSE guest machine                               */
/*   COPIES n  sends the job file n times to the VSE guest machine */
/*   ORIGIN fn ft  original filename and filetype of the job file  */
/*                                                                 */
Trace 'o'
Parse Source . . thisfn .
'IDENTIFY ( LIFO'
Parse Pull userid .

Parse Arg machine fn ft fm argument'('options
Upper machine fn ft fm
file = fn' 'ft' 'fm
argument = Strip(argument, 'B')
options = Strip(options, 'B')

/* check options ... */
replace = Ø
nCopies = 1
hold = Ø
edit = Ø
echo = Ø
szOrigin = fn' 'ft
```

```
DO FOREVER
  Parse Upper Var options name value remain
  IF (name = "") THEN LEAVE
  if (name = 'REPL') THEN DO
    replace = 1
    remain = value' 'remain
    END
  if (name = 'COPIES') THEN nCopies = value
  if (name = 'HOLD') THEN DO
    hold = 1
    remain = value' 'remain
    END
  if (name = 'EDIT') THEN DO
    edit = 1
    remain = value' 'remain
    END
  if (name = 'ORIGIN') THEN DO
    Parse Upper Var remain value2 remain
    szOrigin = value' 'value2
    END
  options = remain
  END

'ESTATE 'file' ( LIFO'
IF (rc <> Ø) THEN DO
  Say 'Error: File 'file' not found!'
  EXIT
  END
Pull file

Say 'Submitting a job file to the VSE guest machine 'machine' ...'

jobfile = thisfn' $$PJOB$$ A'
'ESTATE 'jobfile' ( LIFO'
IF (rc = Ø) THEN DO
  Pull jobfile
  IF (file <> jobfile) THEN 'ERASE 'jobfile
  END

Say '- Determine class of job file ...'
file_class = ''
jobmachs = 'VSEPROD VSETEST'
/* the first job class is the default job class ... */
jobclass = 'Ø123456789ABCDEF'
DO FOREVER
  'EXECIO 1 DISKR 'file' ( FIFO'
  IF (rc <> Ø) THEN LEAVE
  Parse Pull line
` IF (index(line, '%% MACHINE ') = 1) THEN DO
    Parse Var line '%% MACHINE ' jobmachs
    END
  IF (file_class = '') THEN DO
    IF (index(line, '* $$ JOB') = 1) THEN DO
      Parse Var line . 'JNM=' jobname ',' .
      jobname = Strip(jobname, 'B')
      file_class = 'POWER-Job'
      END
    IF (index(line, '// JOB') = 1) THEN DO
      Parse Var szOrigin jobname .
      file_class = 'VSE-Job'
      END
    END
  END
'FINIS 'file

IF (WordPos(machine, jobmachs) = Ø) THEN DO
  nErrorCount = nErrorCount + 1
  Say 'Error: File 'file
  Say '       cannot be sent to VSE guest machine 'machine'!'
  END
IF (nErrorCount > Ø) THEN EXIT

Say '- Building the job file ...'
nErrorCount = Ø
IF (file_class = 'POWER-Job') THEN DO
  IF (file <> jobfile) THEN 'COPYFILE 'file' 'jobfile' ( REPL'
  msg = 'PWR Job 'jobname' file 'file' sent to 'machine'!'
  END
ELSE IF (file_class = 'VSE-Job') THEN DO
  Parse Var argument partition argument
  partitions = 'BG F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF'
  classes =    ' Ø  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F'
  n = wordpos(partition, partitions)
  IF (n = Ø) THEN DO
    class = SubStr(jobclass, 1, 1)
    n = WordPos(class, classes)
    argument = partition' 'argument
    END
  partition = word(partitions, n)
  class = word(classes, n)
  IF (Pos(class, jobclass) = Ø) THEN DO
    nErrorCount = nErrorCount + 1
    Say 'Error: VSE job 'jobname' file 'file
    Say '       cannot be executed in partition 'partition'!'
    END
  ELSE DO
    /* record format VARIABLE ... */
    'EXECIO Ø DISKW 'jobfile' 1 V'
    'MAKEBUF'
    IF (echo = Ø) THEN Queue '* $$ JOB
```

```
       JNM='jobname',CLASS='class',DISP=D'
    ELSE Queue '* $$ JOB
       JNM='jobname',CLASS='class',DISP=D,ECHO=(ALL,'userid')'
    Queue '* $$ LST DISP=D,CLASS=R,DEST=(,'userid')'
    Queue '* $$ PUN DISP=D,CLASS=R,DEST=(,'userid')'
    'EXECIO 3 DISKW 'jobfile
    'DROPBUF'
    'FINIS 'jobfile
    'COPYFILE 'file' 'jobfile' ( APPEND'
    'MAKEBUF'
    Queue '* $$ EOJ';
    'EXECIO 1 DISKW 'jobfile
    'DROPBUF'
    'FINIS 'jobfile
    msg = 'VSE job 'jobname' file 'file' sent to 'machine
   ' and will run in partition 'partition'!'
    END
  END
ELSE DO
  'COPYFILE 'file' 'jobfile' ( REPL'
  msg = 'Job 'jobname' file 'file' sent to 'machine'!'
  END

IF (nErrorCount > Ø) THEN EXIT

/* insert PDELETE if option REPL has been specified ... */
IF (replace) THEN DO
  Say '- process option REPL ...'
  Queue 'TOP'
  Queue 'GET PDELETE MPPINCL * QUIET ARG RDR 'jobname
  Queue ''
  Queue 'FFILE'
  'XEDIT 'jobfile' ( NOPROFIL'
  END

'ESTATE J2VSE $$9999$$ A1 ( ERASE'
Say '- call macro preprocessor...'
'MAKEBUF'
Queue 'MPP'
Queue 'LOG J2VSE $$9999$$ A1'
IF (argument <> '') THEN Queue 'ARG 'argument
Queue 'SYS 'szOrigin
Queue 'VSE 'machine
Queue ''
Queue 'FFILE'
'XEDIT 'jobfile' ( NOPROFIL'
'DROPBUF'

'ESTATE J2VSE $$9999$$ A1'
IF (rc = Ø) THEN DO
```

```
  DO FOREVER
    'EXECIO 1 DISKR J2VSE $$9999$$ A1 ( LIFO'
    IF (rc <> Ø) THEN LEAVE
    Parse Pull szLine
    Say szLine
    END
  Say 'Error: file contains unresolved preprocessor directives
       and/or macros!'
  nErrorCount = nErrorCount + 1
  END

IF (nErrorCount = Ø) THEN DO
  /* check if the record format is FIXED 8Ø ... */
  'EFLIST 'jobfile' ( LIFO'
  Parse Pull . . . recfm lrecl .
  IF (lrecl > 8Ø) THEN DO
    Say 'Error: record length must be lower or equal 8Ø bytes! '
    nErrorCount = nErrorCount + 1
    END
  END

IF (nErrorCount = Ø) THEN DO
  IF (recfm''lrecl <> 'F8Ø') THEN DO
    Queue 'RECFM F'
    Queue 'FFILE'
    'XEDIT 'jobfile' ( NOPROFIL WIDTH 8Ø'
    END
  END

IF (edit) THEN DO
  'XEDIT 'jobfile' ( NOCLEAR'
  END

IF (hold) THEN DO
  'BROWSE 'jobfile' ( NOCLEAR'
  END

IF (nErrorCount = Ø) THEN DO
  IF (hold = Ø) THEN DO
    Say msg
    'CP SPOOL PUNCH TO 'machine
    DO nCopies
      'PUNCH 'jobfile' ( NOH'
      END
    'CP CLOSE PUNCH'
    'CP SPOOL PUNCH TO *'
    IF (file <> jobfile) THEN 'ERASE 'jobfile
    'ESTATE J2VSE $$9999$$ A1 ( ERASE'
    END
  END

EXIT
```

J XEDIT

```
/* Datei:               J        XEDIT         User:  AB          */
/* Autor:               Abstreiter, Franz      Date:  11 Aug 1997 */
/* Version: 1.00                                                  */
/* (x) compiled by      Abstreiter, Franz             13 Aug 1997 */
/* (x) released by      Abstreiter, Franz             20 Aug 1997 */
/*                                                                */
/* Call:                                                          */
/*   J2VSE machine partition values ( options )                   */
/* Parameter:                                                     */
/*   machine    VSE guest virtual machine the job will be sent to */
/*   partition  Partition in which the job will be executed       */
/*   values     Values of the parameters of the job file          */
/* Options:                                                       */
/*   same as J2VSE EXEC                                           */
/*                                                                */
/* Comments:                                                      */
/* - If calling J2VSE within XEDIT does not work it will be       */
/*   necessary to insert the following line into your XEDIT       */
/*    profile:                                                    */
/*     SET SYNONYM JOIN 4 JOIN                                    */
/*   or just install JOIN XEDIT.                                  */
/*                                                                */
Trace 'o'
Parse Source . . thisfn .

Parse Arg szSuffix szArgument
Upper szSuffix

IF (WordPos(szSuffix, "2VSE") = 0) THEN DO
  'COMMAND MSG Error: Macro 'thisfn''szSuffix' not defined!'
  EXIT
  END
szFile = 'J2VSE $$0000$$ A1'
'ESTATE 'szFile' ( ERASE'

'EXTRACT /FNAME/FTYPE'
szOrigin = 'ORIGIN 'fname.1' 'ftype.1
IF (Pos('(', szArgument) = 0) THEN szArgument = szArgument' ('
szArgument = szArgument' 'szOrigin

'EXTRACT /LINE'
nLineNo = line.1
```

```
'COMMAND TOP'
'COMMAND PUT * 'szFile
'COMMAND LOCATE :'nLineNo

Address CMS 'EXEC 'thisfn''szSuffix' 'szFile' 'szArgument

'ERASE 'szFile

EXIT
```

JOIN XEDIT

```
/* Datei:                JOIN     XEDIT         User:  AB           */
/* Autor:                Abstreiter, Franz      Date:  20 Aug 1997 */
/* Version: 1.00                                                   */
/* (x) compiled by       Abstreiter, Franz             20 Aug 1997 */
/* (x) released by       Abstreiter, Franz             20 Aug 1997 */
/*                                                                 */
Trace 'o'
Parse Arg szArgument
Parse Var szArgument szKey szRemain
Upper szKey
IF (WordPos(szKey, '2VSE') > 0) THEN DO
  'MACRO J 'szKey' 'szRemain
  END
ELSE DO
  'COMMAND JOIN 'szArgument
  END
EXIT
```

MPP XEDIT

```
/* Datei:                MPP      XEDIT         User:  AB           */
/* Autor:                Abstreiter, Franz      Date:  20 Aug 1997 */
/* Version: 1.00                                                   */
/* (x) compiled by       Abstreiter, Franz             20 Aug 1997 */
/* (x) released by       Abstreiter, Franz             20 Aug 1997 */
/*                                                                 */
/* Macro PreProcessor                                              */
/*                                                                 */
/* The macro preprocessor interprets directives and substitutes    */
/* macros by their values.                                         */
/* Syntax of macro definition:                                     */
/*   macrokey=macrovalue                                           */
/* Syntax of macro application:                                    */
/*   %macrokey%                                                    */
/* Syntax of macro directives:                                     */
/*   %% DEFINE macrokey=macrovalue                                 */
/*   %% UNDEF macrokey                                            */
/*   %% INCLUDE fn ft fm                                          */
/*   %% IFDEF macrokey                                            */
/*   %% IFNDEF macrokey                                           */
/*   %% ELSIFDEF macrokey                                         */
/*   %% ELSIFNDEF macrokey                                        */
/*   %% ELSE                                                      */
/*   %% ENDIF                                                     */
/* The macrokey must not contain the characters '%' and '='.      */
/* The macro directive %% INCLUDE takes 'MPPINCL' as default      */
/* filetype                                                       */
/*                                                                */
/* Parameter:                                                     */
/*   ARG arg1 arg2 ...                                            */
/*     The macros %1%, %2%, ... will be substituted by values     */
/*     arg1, arg2, ...                                            */
/*   DEF fn ft fm                                                 */
/*     Macro definitions are read from file fn ft fm.             */
/*   SYS fn ft                                                    */
/*     Macros defined by the system will be used. fn and ft       */
/*     specify the origin file.                                   */
/*   VSE fn                                                       */
/*     Macro definitions specific to a VSE guest machine will be  */
/*     used and macro definitions will be read from file          */
/*     fn MPPDEF *.                                               */
/*   MAC macrokey=value                                           */
/*     The given macro definition will be used.                   */
/*   GET fn ft fm argument                                        */
/*     The file fn ft fm will be included and the macro           */
/*     preprocessor will be called with the given arguments.      */
/*   LOG fn ft fm                                                 */
/*     The macros not expanded will be logged into the given file. */
/*   QUIET                                                        */
/*     The substitutions will not be displayed at the terminal.   */
/*                                                                */
Trace 'o'

Parse Source . . thisfn .

Parse Arg key params
IF (key <> '' & key <> 'STACK') THEN DO
  Push ''
  Push key' 'params
  END

'COMMAND EXTRACT /ARBCHAR/LINE/MSGMODE/WRAP'
szSetArbchar = 'SET ARBCHAR 'arbchar.1' 'arbchar.2
szLocateLine = 'LOCATE :'line.1
szSetMsgmode = 'SET MSGMODE 'msgmode.1' 'msgmode.2
szSetWrap = 'SET WRAP 'wrap.1
```

```
'SET MSGMODE OFF'
'SET WRAP OFF'
'SET ARBCHAR OFF'

bQuiet = Ø
szFileLog = ''
szOrigin = ''
DO Queued()
  Parse Pull line
  IF (line = '') THEN LEAVE
  bMacro = 1
  DO FOREVER
    Parse Var line szCommand szOption
    Parse Var szOption szValue szRemain
    szCommand = Strip(szCommand, 'B')
    IF (szCommand = '') THEN LEAVE
    Upper szCommand
    IF (szCommand = 'ARG') THEN DO
      DO i=1
        param = Word(szOption, i)
        IF (param = '') THEN Leave
        Call DefineMacro i'='param
        END
      line = ''
      bMacro = Ø
      END
    IF (szCommand = 'VSE') THEN DO
      machine = szValue
      machname = machine
      Call DefineMacrosByFile 'VSE'
      Call DefineMacro 'VSE.MACHINE='machname
      Call DefineMacrosByFile machine
      line = szRemain
      bMacro = Ø
      END
    IF (szCommand = 'DEF') THEN DO
      Call DefineMacrosByFile Words(szOption, 1, 3)
      line = Words(szOption, 4)
      bMacro = Ø
      END
    IF (szCommand = 'SYS') THEN DO
      fn = szValue
      Parse Var szRemain ft szRemain
      szOrigin = Strip(fn' 'ft, 'B')
      IF (Words(szFileLog) < 2) THEN szOrigin = ''
      Call DefineMacrosBySystem
      line = szRemain
      bMacro = Ø
      END
    IF (szCommand = 'QUIET') THEN DO
```

```
      bQuiet = 1
      line = szOption
      bMacro = Ø
      END
    IF (szCommand = 'MAC') THEN DO
      Call DefineMacro szOption
      line = ''
      bMacro = Ø
      END
    IF (szCommand = 'LOG') THEN DO
      fn = szValue
      Parse Var szRemain ft fm szRemain
      szFileLog = Strip(fn' 'ft' 'fm, 'B')
      IF (Words(szFileLog) < 3) THEN szFileLog = ''
      line = szRemain
      bMacro = Ø
      END
    IF (szCommand = 'GET') THEN DO
      Call IncludeFile szOption
      line = ''
      bMacro = Ø
      END
    IF ((bMacro) & (Pos('=', line) > Ø)) THEN DO
      Call DefineMacro line
      LEAVE
      END
    END
  END

nCountUndefs = Ø
ifskips = Ø
ifexecs = Ø
ifstate.ifexecs = 1
'TOP'
DO FOREVER
  IF (ifstate.ifexecs = 1) THEN DO
    'COMMAND LOCATE /%/'
    IF (rc <> Ø) THEN Leave
    END
  'EXTRACT /LINE'
  lineno = line.1
  'EXTRACT /CURLINE'
  line = curline.3
  uline = line
  Upper uline
  DO WHILE (pos('%% ', uline) = 1)
    /* Expand include-Macros ... */
    IF (pos('%% INCLUDE ', uline) = 1) THEN DO
      IF (ifstate.ifexecs <> 1) THEN DO
        'DELETE +1'
        IF (rc <> Ø) THEN Leave
        END
      ELSE DO
        include_file = SubStr(uline, 12)
        IF (Words(include_file) = 1) THEN include_file =
         include_file' MPPINCL'
        IF (Words(include_file) = 2) THEN include_file =
         include_file' *'
        'ESTATE 'include_file' ( LIFO'
        IF (rc <> Ø) THEN DO
          'LOCATE +1'
          Push 'Datei 'include_file' not found]'
          IF (szFileLog <> '') THEN 'EXECIO 1 DISKW 'szFileLog
          END
        ELSE DO
          Pull include_file
          'GET 'include_file
          IF (bQuiet = Ø) THEN Say 'Include 'include_file
          'LOCATE :'lineno
          'DELETE +1'
          END
        END
      END
    IF (pos('%% DEFINE ', uline) = 1) THEN DO
      IF (ifstate.ifexecs = 1) THEN DO
        Call DefineMacro SubStr(line, 11)
        END
      'DELETE +1'
      IF (rc <> Ø) THEN Leave
      END
    IF (pos('%% UNDEF ', uline) = 1) THEN DO
      IF (ifstate.ifexecs = 1) THEN DO
        key = SubStr(line, 1Ø)
        param.key = 'PARAM.'key
        END
      'DELETE +1'
      IF (rc <> Ø) THEN Leave
      END
    IF (pos('%% IFDEF ', uline) = 1) THEN DO
      IF (ifstate.ifexecs <> 1) THEN ifskips = ifskips + 1
      ELSE DO
        pattern = SubStr(line, 1Ø)
        ifexecs = ifexecs + 1
        IF (param.pattern = 'PARAM.'pattern) THEN ifstate.ifexecs = Ø
        ELSE ifstate.ifexecs = 1
        END
      'DELETE +1'
      IF (rc <> Ø) THEN Leave
      END
    IF (pos('%% IFNDEF ', uline) = 1) THEN DO
```

```
    IF (ifstate.ifexecs <> 1) THEN ifskips = ifskips + 1
    ELSE DO
      pattern = SubStr(line, 11)
      ifexecs = ifexecs + 1
      IF (param.pattern = 'PARAM.'pattern) THEN ifstate.ifexecs = 1
      ELSE ifstate.ifexecs = Ø
      END
    'DELETE +1'
    IF (rc <> Ø) THEN Leave
    END
  IF (pos('%% ELSIFDEF ', uline) = 1) THEN DO
    IF (ifskips = Ø) THEN DO
      ifstate.ifexecs = ifstate.ifexecs + 1
      IF (ifstate.ifexecs = 1) THEN Do
        pattern = SubStr(line, 13)
        IF (param.pattern = 'PARAM.'pattern) THEN ifstate.ifexecs =
         Ø
        ELSE ifstate.ifexecs = 1
        END
      END
    'DELETE +1'
    IF (rc <> Ø) THEN Leave
    END
  IF (pos('%% ELSIFNDEF ', uline) = 1) THEN DO
    IF (ifskips = Ø) THEN DO
      ifstate.ifexecs = ifstate.ifexecs + 1
      IF (ifstate.ifexecs = 1) THEN Do
        pattern = SubStr(line, 14)
        IF (param.pattern = 'PARAM.'pattern) THEN ifstate.ifexecs =
         1
        ELSE ifstate.ifexecs = Ø
        END
      END
    'DELETE +1'
    IF (rc <> Ø) THEN Leave
    END
  IF (pos('%% ELSE', uline) = 1) THEN DO
    IF (ifskips <> 1) THEN ifstate.ifexecs = ifstate.ifexecs + 1
    'DELETE +1'
    IF (rc <> Ø) THEN Leave
    END
  IF (pos('%% ENDIF', uline) = 1) THEN DO
    IF (ifskips > Ø) THEN ifskips = ifskips - 1
    ELSE DO
      IF (ifexecs > Ø) THEN ifexecs = ifexecs - 1
      END
    'DELETE +1'
    IF (rc <> Ø) THEN Leave
    END
  'EXTRACT /CURLINE'
  line = curline.3
  uline = line
  Upper uline
  END
/* substitute macros ... */
IF (ifstate.ifexecs <> 1) THEN DO
  'COMMAND DELETE +1'
  END
ELSE DO
  szWork = line
  DO FOREVER
    bChanged = Ø
    text = ''
    pospos = 1
    DO FOREVER
      patpos1 = pos('%', szWork, pospos)
      IF (patpos1 = Ø) THEN parpos = Ø
      ELSE DO
        patpos2 = pos('%', szWork, patpos1+1)
        IF (patpos2 = Ø) THEN parpos = Ø
        ELSE DO
          pattern = substr(szWork, patpos1+1, patpos2-patpos1-1)
          IF (pattern = '') THEN newtext = '%'pattern'%'
          ELSE IF (param.pattern <> 'PARAM.'pattern)
          THEN newtext = param.pattern
          ELSE DO
            newtext = '%'pattern'%'
            IF (szFileLog <> '') THEN DO
              DO n=1 TO nCountUndefs
                IF (undefs.n = newtext) THEN LEAVE
                END
              IF (n > nCountUndefs) THEN DO
                nCountUndefs = nCountUndefs + 1
                undefs.nCountUndefs = newtext
                END
              END
            END
          pattern = '%'pattern'%'
          parpos = pos(pattern, szWork, pospos)
          END
        END
      IF (parpos = Ø) THEN DO
        parpos = length(szWork) + 1
        pattern = ''
        newtext = ''
        END
      chars = parpos - pospos
      IF (chars > Ø) THEN text = text''substr(szWork, pospos,
       chars)
      IF (pattern <> newtext) THEN bChanged = 1
```

```
        text = text''newtext
        pospos = parpos + length(pattern)
        IF (pospos > length(szWork)) THEN Leave
        END
      IF (bChanged = 0) THEN LEAVE
      szWork = text
      END
    IF (line <> text) THEN DO
      'REPLACE 'text
      IF (bQuiet = 0) THEN Say 'Change 'strip(line, 'T')
      IF (bQuiet = 0) THEN Say '    to 'strip(text, 'T')
      END
    END
  END

'COMMAND 'szSetArbchar
'COMMAND 'szLocateLine
'COMMAND 'szSetMsgmode
'COMMAND 'szSetWrap

IF ((nCountUndefs > 0) & (szFileLog <> '')) THEN DO
  'MAKEBUF'
  nentries = Queued()
  DO n=1 TO nCountUndefs
    Queue 'Macro 'undefs.n' not defined!'
    END
  'EXECIO 'Queued()-nentries' DISKW 'szFileLog
  'DROPBUF'
  END

EXIT

DefineMacrosByFile:
  Parse Arg szFileDef
  szFileDef = Strip(szFileDef, 'B')
  IF (Words(szFileDef) = 0) THEN RETURN
  IF (Words(szFileDef) = 1) THEN szFileDef = szFileDef' MPPDEF'
  IF (Words(szFileDef) = 2) THEN szFileDef = szFileDef' *'
  'ESTATE 'szFileDef' ( LIFO'
  IF (rc = 0) THEN DO
    Pull szFileDef
    DO FOREVER
      'EXECIO 1 DISKR 'szFileDef' ( LIFO'
      IF (rc <> 0) THEN LEAVE
      Parse Pull line
      Call DefineMacro line
      END
    END
  RETURN
```

```
DefineMacro:
  Parse Arg szMacroDefinition
  IF (pos('*', szMacroDefinition) = 1) THEN RETURN
  IF (pos('=', szMacroDefinition) = Ø) THEN RETURN
  Parse Var szMacroDefinition key'='text
  param.key = text
  IF (pos('''', text) = 1) THEN DO
    n = Length(text)
    c = SubStr(text, n, 1)
    IF (c = '''') THEN param.key = SubStr(text, 2, n-2)
    END
  Return

DefineMacrosBySystem:
  /* macros defined by the system ... */
  'IDENTIFY ( LIFO'
  Parse Pull szUserId .
  zeit = time()
  datum = date('O')
  thisdd = substr(datum, 7, 2)
  thismm = substr(datum, 4, 2)
  thisyy = substr(datum, 1, 2)
  thisyyyy = 19ØØ + thisyy
  IF (thisyyyy < 1997) THEN thisyyyy = thisyyyy + 1ØØ
  /* previous month ... */
  prevmonth.mm = thismm - 1
  prevmonth.yy = thisyy
  IF (prevmonth.mm = Ø) THEN DO
    prevmonth.mm = 12
    prevmonth.yy = prevmonth.yy - 1
    END
  prevmonth.mm = right(prevmonth.mm, 2, 'Ø')
  prevmonth.yy = right(prevmonth.yy, 2, 'Ø')
  /* following month ... */
  nextmonth.mm = thismm + 1
  nextmonth.yy = thisyy
  IF (nextmonth.mm = 13) THEN DO
    nextmonth.mm = 1
    nextmonth.yy = nextmonth.yy + 1
    END
  nextmonth.mm = right(nextmonth.mm, 2, 'Ø')
  nextmonth.yy = right(nextmonth.yy, 2, 'Ø')
  /* define macros ... */
  Call DefineMacro 'USERID='szUserId
  IF (szOrigin <> '') THEN DO
    Call DefineMacro 'FNAME='Word(szOrigin, 1)
    Call DefineMacro 'FTYPE='Word(szOrigin, 2)
    END
  Call DefineMacro 'THISDAY.DDMMYY='thisdd''thismm''thisyy
  Call DefineMacro 'THISDAY.DD/MM/YYYY='thisdd'/'thismm'/'thisyyyy
  Call DefineMacro 'THISDAY.DD.MM.YYYY='thisdd'.'thismm'.'thisyyyy
  Call DefineMacro 'THISMONTH.MM/YY='thismm'/'thisyy
  Call DefineMacro 'NEXTMONTH.MM/YY='nextmonth.mm'/'nextmonth.yy
  Call DefineMacro 'PREVMONTH.MM/YY='prevmonth.mm'/'prevmonth.yy
  Call DefineMacro 'THISMONTH.MMYY='thismm''thisyy
  Call DefineMacro 'NEXTMONTH.MMYY='nextmonth.mm''nextmonth.yy
  Call DefineMacro 'PREVMONTH.MMYY='prevmonth.mm''prevmonth.yy
  IF (thisdd < 16) THEN Call DefineMacro 'TWOMONTHS.MMYY,MMYY='
   prevmonth.mm''prevmonth.yy','thismm'/'thisyy
  ELSE Call DefineMacro 'TWOMONTHS.MMYY,MMYY='
   nextmonth.mm''nextmonth.yy','thismm''thisyy
  RETURN

IncludeFile:
  Parse Arg fn ft fm szIncludeOption
  szIncludeOption = Strip(szIncludeOption, 'B')
  'ESTATE 'fn ft fm' ( LIFO'
  IF (rc <> Ø) THEN RETURN
  Pull fn ft fm
  'EXTRACT /LINE'
  nLineNo = line.1
  'GET 'fn ft fm
  'LOCATE :'nLineNo
  IF (szIncludeOption <> '') THEN DO
    'EXTRACT /LINE/RANGE'
    nLines = line.1 - nLineNo + 1
    'SET RANGE :'nLineNo' +'nLines
    'MACRO 'thisfn' 'szIncludeOption
    'SET RANGE :'range.1' :'range.2
    END
  RETURN
```

Franz Abstreiter
(Germany)

Synchronizing a batch job with a given fixed time

We had been having problems synchronizing a batch job with a given fixed time, despite using the two options available to resolve this problem, namely:

- The new keyword operands like DUETIME and DUEDATE, which had been added to the VSE/POWER * $$ JOB statement to support time event scheduling for VSE/POWER jobs.

- The IBM-supplied program IESWAIT, which waits for the number of seconds specified as the PARM value of the EXEC statement.

For our time-critical application, the last steps of the corresponding job could be executed only after a given absolute time. Splitting the application into two separate jobs was not an option, because the last steps needed information – such as logical assignments and the values of symbolic parameters – from the first steps.

I therefore wrote the small program presented here, which allows a job to synchronize with a given absolute time. It can be called in one of the following three formats:

```
•      // EXEC B252SYN,PARM='HH'
•      // EXEC B252SYN,PARM='HH:MM'
•      // EXEC B252SYN,PARM='HH:MM:SS'
```

NOTES

- The PARM value of the EXEC statement specifies the absolute time using the 24-hour clock, with HH for the hour (from 00 through to 23), MM for the minutes, and SS for the seconds (both from 00 through to 59). Leading zeroes must be specified. Omitted values for minutes and seconds are defaulted to 00.
- The program waits only if the computed interval is less than twelve hours.
- If the format of the PARM value is wrong, no wait occurs, and return code 9 is issued to job control.
- Because of a restriction in the EOJ macro with the RC keyword, this program can only be executed below the 16MB line (RMODE 24).

EXAMPLE

In the following example, the program waits until 6 pm if the step is executed after 6 am and before 6 pm.

```
* $$ JOB JNM=TIMECRIT,...
  // JOB TIMECRIT ...
  ...
  Steps that can be executed before 6 pm
  ...
  // EXEC B252SYN,PARM='18'
  ...
  Steps that must be executed after 6 pm
  ...
  /&
  * $$ EOJ
```

SOURCE OF B252SYN

```
TITLE 'B252SYN - WAIT UNTIL A SPECIFIED ABSOLUTE TIME'
B252SYN  CSECT
B252SYN  AMODE 24
B252SYN  RMODE 24
         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
         EJECT
*********************************************************************
* REGISTER USAGE:
*
*    R15 PROGRAM ENTRY POINT, RETURN CODE
*    R14 RETURN ADDRESS
*    R13
*    R12
*    R11
*    R1Ø
*    R9  BASE REGISTER
*    R8
*    R7
*    R6
*    R5
*    R4
*    R3  WORK REGISTER
*    R2  WORK REGISTER, LENGTH OF PARM STRING (ABSOLUTE TIME)
*    R1  ADDRESS OF PARM STRING (INPUT PARAMETER), USED BY IBM MACROS
*    RØ  USED BY IBM MACROS
**********************************************************************
         EJECT
**********************************************************************
*        TEST INPUT PARAMETER, INITIALIZE REGISTERS AND STORAGE
**********************************************************************
         BALR  R9,Ø                     LOAD BASE REGISTER
         USING *,R9                     ESTABLISH ADDRESSABILITY
         CR    R1,R15                   PARM STRING EXISTS
         BE    RETURN9                  NO, INFORM JOB CONTROL
         TM    Ø(R1),X'8Ø'              HIGH ORDER BIT OK
         BNO   RETURN9                  NO, INFORM JOB CONTROL
         L     R1,Ø(,R1)                ADDRESS OF PARM STRING
         LH    R2,Ø(,R1)                LOAD LENGTH OF PARM STRING
         CH    R2,=H'8'                 TEST LENGTH OF PARM STRING
         BH    RETURN9                  TOO LONG, INFORM JOB CONTROL
         BL    NOSECSP                  NO SECONDS SUPPLIED
         CLI   7(R1),C':'               DELIMITER COLON
         BNE   RETURN9                  NO, INFORM JOB CONTROL
         MVC   ABSTIME+4(2),8(R1)       MOVE SECONDS
         B     MOVEMIN                  MOVE MINUTES
NOSECSP  DS    ØH
         CH    R2,=H'5'                 TEST LENGTH OF PARM STRING
         BL    NOMINSP                  NO MINUTES SUPPLIED
         BNE   RETURN9                  WRONG LENGTH, INFORM JOB
                                         CONTROL
MOVEMIN  DS    ØH
         CLI   4(R1),C':'               DELIMITER COLON
         BNE   RETURN9                  NO, INFORM JOB CONTROL
         MVC   ABSTIME+2(2),5(R1)       MOVE MINUTES
         B     MOVEHOUR                 MOVE HOURS
NOMINSP  DS    ØH
         CH    R2,=H'2'                 TEST LENGTH OF PARM STRING
         BNE   RETURN9                  WRONG LENGTH, INFORM JOB
                                         CONTROL
MOVEHOUR DS    ØH
         MVC   ABSTIME(2),2(R1)         MOVE HOURS
         EJECT
**********************************************************************
*        CHECK SUPPLIED ABSOLUTE TIME
**********************************************************************
         LA    R3,L'ABSTIME             LENGTH OF ABSOLUTE TIME
TSTNXT   DS    ØH
         LA    R2,ABSTIME-1(R3)         NEXT CHARACTER
         CLI   Ø(R2),C'Ø'               CHARACTER LESS THAN Ø
         BL    RETURN9                 YES, INFORM JOB CONTROL
         CLI   Ø(R2),C'9'              CHARACTER GREATER THAN 9
         BH    RETURN9                 YES, INFORM JOB CONTROL
         BCT   R3,TSTNXT               TEST NEXT CHARACTER
         CLI   ABSTIME+2,C'5'          MINUTES GREATER THAN 59
         BH    RETURN9                 YES, INFORM JOB CONTROL
         CLI   ABSTIME+4,C'5'          SECONDS GREATER THAN 59
         BH    RETURN9                 YES, INFORM JOB CONTROL
         CLI   ABSTIME,C'2'            HOURS GREATER THAN 29
         BH    RETURN9                 YES, INFORM JOB CONTROL
         BL    HOUROK                  HOURS LESS THAN 2Ø, HOURS O.K.
         CLI   ABSTIME+1,C'3'          HOURS GREATER THAN 23
         BH    RETURN9                 YES, INFORM JOB CONTROL
HOUROK   DS    ØH
         EJECT
**********************************************************************
*        CONVERT SUPPLIED ABSOLUTE TIME TO BINARY SECONDS
**********************************************************************
         PACK  CVBTIME,ABSTIME(2)      PACK HOURS
         CVB   R2,CVBTIME              STORE HOURS IN REGISTER 2
         MH    R2,=H'6Ø'               CONVERT HOURS TO MINUTES
         PACK  CVBTIME,ABSTIME+2(2)    PACK MINUTES
         CVB   R3,CVBTIME              STORE MINUTES IN REGISTER 3
         AR    R2,R3                   TOTAL AMOUNT OF MINUTES IN
                                        REG.2
         MH    R2,=H'6Ø'               CONVERT MINUTES TO SECONDS
         PACK  CVBTIME,ABSTIME+4(2)    PACK SECONDS
         CVB   R3,CVBTIME              STORE SECONDS IN REGISTER 3
         AR    R2,R3                   TOTAL AMOUNT OF SECONDS IN
                                        REG.2
         EJECT
**********************************************************************
*        GET TIME OF DATE IN REGISTER 1 AS BINARY NUMBER OF SECONDS
**********************************************************************
         GETIME BINARY
         EJECT
**********************************************************************
*        COMPUTE DIFFERENCE BETWEEN TIME OF DATE AND SUPPLIED TIME
**********************************************************************
         SR    R2,R1                   COMPUTE DIFFERENCE
         BZ    RETURN                  NO DIFFERENCE, DO NOT WAIT
         BP    PARMLATE                SUPPLIED TIME IS LATER
         A     R2,=F'864ØØ'            ADD 24 HOURS
PARMLATE DS    ØH
         C     R2,=F'432ØØ'            INTERVAL LESS THAN 12 HOURS
         BNL   RETURN                  NO, DO NOT WAIT
         EJECT
**********************************************************************
*        WAIT UNTIL THE SUPPLIED ABSOLUTE TIME
**********************************************************************
```

```
         LR    R1,R2                    LOAD TIME INTERVAL
         SETIME (1),TECB                SET INTERVAL TIMER
         WAIT  TECB                     WAIT TIMER EVENT CONTROL BLOCK
         EJECT
**********************************************************************
*        TERMINATE PROGRAM WITH RETURN CODE
**********************************************************************
         SR    R15,R15                  SET RETURN CODE TO Ø
RETURN   EOJ   RC=(15)                  RETURN TO JOB CONTROL
         EJECT
**********************************************************************
*        SET JCL RETURN CODE FOR MISSING OR WRONG PARAMETER
**********************************************************************
RETURN9  DS    ØH
         LA    R15,9                    SET RETURN CODE TO 9
         B     RETURN                   TERMINATE PROGRAM
         EJECT
**********************************************************************
*        WORKING STORAGE
**********************************************************************
CVBTIME  DS    D                        STORAGE TO PACK SUPPLIED TIME
TECB     TECB                           TIMER EVENT CONTROL BLOCK
ABSTIME  DC    CL6'ØØØØØØ'              SUPPLIED ABSOLUTE TIME
         END   B252SYN
```

Walter Richters
(Germany)

VSE to MVS conversion

With the recent level of enthusiasm for ‘rightsizing’ and distributed client/server architectures, it would be easy to believe that medium-sized mainframe users never upgrade to more powerful operating systems. But for many existing VM/VSE users, MVS can be the best route forward. This chapter considers the practical implications of VSE to MVS conversion.

THE DRIVERS – WHO AND WHY?

Over the past nine years, we have performed VSE to MVS conversions at over 60 sites in seven countries. The businesses which have undergone these conversions can be divided into the categories shown in Figure 1.

Sector	Percent
Financial sector	45%
Manufacturing	25%
Government	10%
Oil/Gas/Transport	10%
Pharmaceutical	5%
Entertainment	5%

Figure 1: Businesses which have undertaken VSE to MVS conversions

The major reason for migrating is economic rather than technical. Data centre consolidation and the economies of scale which can be achieved as a consequence provide the justification. One large insurance company consolidated five VM/VSE sites scattered around Europe into a single data centre in Paris. Another example is a bank which is predominantly an MVS user that migrated the single European VSE site into the main data centre in Germany.

Legislative changes, in particular the single European market and privacy laws, have acted as drivers in a small number of cases and have tended to be country-specific.

Whatever provides the driver and business justification for the migration, IT has its own set of perceptions about the benefits of a migration. At a personal level, most people see it as an opportunity to learn more skills and increase their market value. The commonest view is that MVS will offer the promised improved reliability, availability, and serviceability (just like IBM says) along with the ability to handle large workloads with a high throughput. Replacing several VSE systems with a single, more manageable MVS system is seen as an attraction. Having justified a conversion, many sites use the opportunity to correct or bury some of the mistakes of the past and bundle this into the cost of the migration.

For most software development and maintenance departments, the opportunity to convert modern COBOL to ANSI ’85 standard is seen as a positive move. New language features such as EVALUATE, in-line PERFORMs and scope terminators allow for better, safer, more structured code. Better testing facilities exist under MVS. For CICS users (ie almost everyone), the arcane fiddling with COBOL BLL cells is replaced with the much clearer use of the ADDRESS OF special register. The ability to access a four-digit year in dates will also be handy very soon!

But so far, I have been looking back into the past, with the erosion over the past ten years of the VSE market, as the smaller users downsized to other platforms and the larger users gradually upsized and consolidated. What are the current and future justifications for migrating? Taking the long-term view, just how much longer is VSE going to be a viable platform? Sentiment also plays a part: all the exciting things seem to be happening elsewhere. But any migration is an expensive exercise and is not without risk, so, without a real business case, it isn't going to happen. Major business reorganization will continue to drive some migrations. Changes in the costs of hardware and software and staff will provide opportunities regarding the long-term cost of ownership of VSE compared to MVS.

MOVING TO MVS

Undertaking a move from VSE to MVS is a large project – probably the largest ever in most IT departments. Let's get some terminology sorted out first. Many people refer to a migration, which implies that systems will be moved one by one in a gentle and controlled manner. At first sight, it looks a simple and low-risk approach (apart from the risk of never moving all the systems). But is it feasible? When you consider the data shared by these different systems, which systems update the data and which systems use it, and the need to keep the whole thing in step, you soon realize that it is very unmanageable and carries a high risk. It also increases the effort and cost by introducing the need for backward and forward bridges between VSE and MVS at different stages of the migration process.

The alternative is to refer to a conversion. Here, we are looking at cutting over production and development to MVS in one move – usually referred to as a 'big bang' or 'sudden death' approach, depending on how brave you are feeling. In almost all cases, conversion is less expensive and carries less risk than migration.

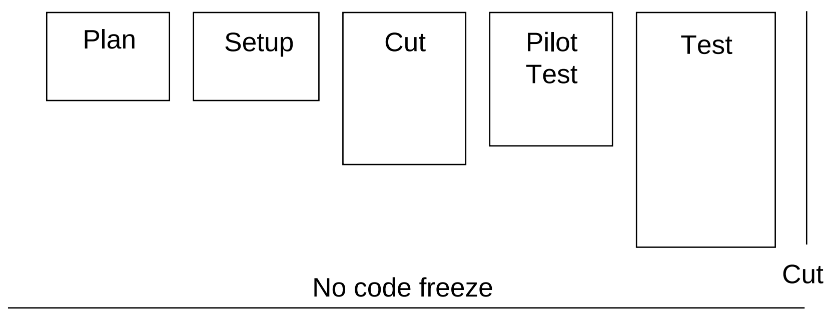

Figure 2: VSE to MVS conversion plan

The general shape of the conversions we have done is shown in Figure 2.

The following sections discuss each phase of the project in turn.

Plan

Every project needs a plan, so what's special about this one? To get a view on the size of the project, the first step is to decide what to convert. Unfortunately, this presents the first problem because most sites are not as well organized as they might like to believe. If you start by looking at libraries of programs, you will probably end up converting programs which are no longer used. Converting unnecessary code means unnecessary effort and expense – it also proves a big problem in testing.

A better place to start is from the production JCL. First, the JCL itself will have to be converted. Second, the JCL will execute lots of utilities whose control statements will need conversion. Finally, it invokes packaged and user-written programs. By following the program executions, you can then go to the libraries and start looking for the subprograms and copybooks that they use, and so on down the program structure. By starting from the JCL, we also ensure that we can find all the source code. Missing source needs to be identified as soon as possible and plans formulated to rebuild or recreate it.

Already, we are beginning to see that this is a tedious and time-consuming process. Searching through source code to find CALLs and COPYs is a job better done by a tool than a person. In fact, the whole conversion process is better carried out by a specialized conversion tool. So, plan to select one; preferably one which is useful right from the start in identifying what needs to be converted.

While a conversion is going on, the business will not take it kindly if development and on-going support are disrupted. In practice, your operations and support staff will be involved in the testing phase, so all you can do is to keep disruption to a minimum rather than avoid it completely. Plan to acquire the necessary human resources to cope with the extra workload. Ideally, this can be achieved by buying in services to complement your chosen conversion tool – preferably people who have used the tool before and understand the processes involved in a conversion project.

Plan the new environment: languages, utilities, tools, databases, and finally hardware. You will need to change and update your standards for the new environment. In particular, areas such as libraries, JCL, and security are very different. Also plan training for the various groups within IT, and decide whether any user education will be necessary.

If you use packages, plan their replacement in MVS. If you have taken a package and modified it, it may be better to convert the package rather than to acquire the MVS version and modify it. Either way, you will have to check with your supplier that MVS versions are available or discuss the contractual implications of conversion to MVS.

Finally, select the cutover date. A long weekend may be necessary. Since cutover is so critical, plan to test the whole cutover process about two weeks before it really happens.

Set-up

The set-up process deals with the installation and customization of the conversion tool. The area where most tailoring takes place relates to standards to be used in the MVS JCL. Most VSE users barely understand MVS JCL at this stage and are not the best people to set the standards. Get some advice from experienced MVS users here.

Conversion

The conversion process is iterative. In order to ensure that no code freeze is necessary during the process, we must be able to pick up the current version of the VSE material at any stage right up to just before the cutover. Various conversion problems will need to be resolved. The best way to achieve this is to build the application of the solution into the conversion process. For example, if it is necessary to make changes to a piece of source code, this would be done by applying a verify/replace into the process either just before or just after translation. That way, every time we fetch updated code from VSE, the change is re-applied.

The main deliverables at the end of the conversion are libraries of converted, compiled, and link-edited programs, libraries of translated utility control statements, and the JCL necessary to run it. Planning should ensure that, as this phase ends, the MVS environment has been set up ready for testing to commence.

Pilot test

Before the main testing commences, we need a pilot. This has two objectives. The first is to ensure that the new environment has been set up correctly – if we see errors during testing, they should be real errors which we can fix as part of the conversion process. Errors caused by a bad MVS set-up or missing facilities can be very disruptive during the main test phase. The second objective of the pilot is to verify the conversion process.

Testing

Testing is not everybody's favourite task, but it has to be done. On most conversions, testing normally involves a full parallel run on every system, and is carried out system by system. You can't risk the consequences of the worst-case scenario, which is that source existed for a program on VSE but was out of date and does not have a fix for a major bug.

The conversion process can deliver the JCL and program and utility libraries that are necessary. Getting hold of the necessary data can be more problematic. If existing operations documentation is good, this may be enough to identify the data sources that need to be copied to MVS before the test commences. Some tools will produce documentation as part of the conversion process to help in identifying the required data, and some will even generate the necessary back-up and restore JCL.

Before this stage starts, it is normal to have the operations support staff trained. It is here that they start to gain experience of using their systems in an MVS environment.

Cutover

I laboured the point earlier of the importance of planning, and cutover is no exception. Cutover requires a very detailed fine-grain plan. Here, we are planning to the hour, not in days or weeks. JCL and program and utilities can be produced in good time by freezing them one week before the cutover and then doing a final conversion. The biggest task of the cutover is to ensure that all the VSE data is transferred to the new environment ready for the start-up of production MVS. This normally involves logical copies on VSE after the last production job has run and reloads on MVS.

In practice, all the data will probably have been moved at some stage of the testing phase, so this should provide useful information about how long the unloads and reloads take.

A good plan is useless if it isn’t followed, so frequent reviews of the actual cutover are necessary to ensure that things are running as scheduled, or that you have time to correct them or, in the worst case, put the fallback plan into effect.

Before cutover, the development staff will need to have been trained in the COBOL language differences and in the new development environment, as well as probably in some basic MVS JCL knowledge.

Life after big bang

If everything has gone according to plan, the effect on the end users will be minimal. Except for probably having to log on fewer times, their systems will appear to them to run just the same as usual – and maybe just a little faster.

The conversion is now complete except that we should probably review it.

But at the start of that first day in MVS, all your JCL and program sources will be current and synchronized. You will carry no extra baggage of bits of source code or files that have been lying around for years but nobody dared to delete. This would be a great time to keep things in step by implementing a rigorous change control/configuration management system.

Mike Godman
ECsoft (UK)

VSE news

Platinum Technology has begun shipping InfoSession for the Web, providing direct access to mainframe applications via a Web browser, without rewriting any code.

The client components run on Windows 95 and NT, plus OS/2, AIX, HP-UX, and Solaris. The mainframe component runs on OS/390 and VSE.

For further information, contact:
Platinum Technology, 1815 S Meyers Road, Oakbrook Terrace, IL 60181-5241, USA.
Tel: (714) 453 4000.
Platinum Technology, Turnberry House, 30 Caldecote Lake Drive, Milton Keynes, Bucks, MK7 8LE, UK.
Tel: (01908) 274777.

* * *

IBM is expanding its VisualAge 2000 portfolio with Millennium Language Extensions, available on VSE/ESA in June.

For further information, contact your local IBM representative.

* * *

Sterling Software has expanded its Vision:Solutions 2000 suite of Year 2000 tools, methodologies, education, and consulting services. The enhancements include Vision:Simulate, which allows testing at the program level for batch (MVS/ESA, OS/390, and VSE), CICS, and IMS/DC/TM without disrupting the normal operation of other programs on the system.

For VSE batch testing, it allows a simulated system date to be applied to all jobs run in any VSE partition and/or VSE dynamic class. Included is a program date/time analyser for locating date/time routines in batch and CICS load modules. It supports COBOL, PL/I, Assembler, and Natural, and includes an optional add-on for testing DB2 and other applications.

For further 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.

* * *

Macro 4 has launched Version 4.3 of its VTAMPRINT/VSE with TCP/IP support for printing networks within VSE installations. The new version enables native VSE and VM/VSE installations to print any VSE output to any printer in the TCP/IP network directly, including printers attached to other platforms such as Unix and NT.

The company has also begun shipping Version 3.7 of its EnterWEB software, with VSE support. EnterWEB gives 3270 users direct access to the Internet and intranets from 3270 terminals.

For further information, contact:
Macro 4, The Orangery, Turners Hill Road, Worth, Crawley, West Sussex, RH10 4SS, UK.
Tel: (01293) 886060.
Macro 4, 35 Waterview Blvd, PO Box 292, Parsippany, NJ 07054-0292, USA.
Tel: (201) 402 8000.

* * *

xephon