150 MVS update

March 1999

In this issue

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

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

Contributions

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

Editor
Jaime Kaminski

Disclaimer

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

MVS Update on-line

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

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £325.00 in the UK; $485.00 in the USA and Canada; £331.00 in Europe; £337.00 in Australasia and Japan; and £335.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Printed in England.

A copy utility that allows rollover files

INTRODUCTION

COPYFILE (CF) is a utility I use all the time when I am working on an MVS platform. I find it tedious dealing with panels when it is possible to just type TSO CF REXX(COPYFILE) 'SYS.REXX(=)'.

A long time ago I had to move some VM applications to MVS. The problem was that MVS does not support the CMS command set. Using REXX as a foundation, the most critical CMS commands were written to run on MVS. COPYFILE was probably the most successful port. It still can convert filename, filetype, and filemode filenames into MVS equivalents using a number of specially designed subroutines and programs. For instance, the dataset PAY.DB(ID) could be copied using 'TSO CF ID PAY.DB A NEWID = ='.

COPYFILE AND ROLLOVERS

Beyond providing me with a TSO and ISPF copy command that, so far as I know, no one else distributes, COPYFILE does something unprecedented. As a developer I sometimes have to keep track of resources in log files. In MVS I have always been forced to either generate huge extents I will never exceed, or limit the extent definition and purge records when the file reaches the maximum. For instance, I keep a list of every file I touch using cover functions to the editor and browse commands. With COPYFILE I can easily rollover the report at a specified number of lines by executing a command every time I log on as follows:

```
COPYFILE EDITHIST 'SYS.EDITHIST' *MAX(900) *APPEND
```

OTHER COPYFILE FUNCTIONALITY

If you have ever used VM, you already know the bulk of what COPYFILE does:

- Reblocks records during moves if required.
- Copies portions of files using sequence numbers or values.

- Acts as an alternative to IEBGENER down to the SYSUTn DD names.
- Allocates sequential or PDS output files when needed, based on what the input file looks like.
- Allows you to specify directory, space, and unit values on allocations if needed.
- Copies your LMF statistics from one file to another.

Just remember that batch executions of COPYFILE will require that ISPF be started and the appropriate DD names (ie ISPxLIBs) allocated.

WHAT COPYFILE DOES NOT DO

There are two things that COPYFILE does not do:

- It does not copy VSAM files
- It does not change data.

If you want this functionality it is fairly simple to add.

SUBROUTINES

Although most of COPYFILE is self contained, there are some sub-programs that you will need to access. As always, because they are simple on the surface, they can be easily replaced by routines you build or have on your systems already.

- CF – this is the cover function for COPYFILE. If your shop is not already using CF I recommend that this REXX function be included. I avoid aliases because they can be tricky when you want to rearrange things. It is really needed because most MVS command lines are tiny, and you want to be able to issue as many copy commands as possible without resorting to the old‘=6’ panel.

```
/* REXXNAME=(CF); AN MVI EXEC */
PARSE ARG ARGSTRING;DEBUG='';$X=FIND(TRANSLATE(ARGSTRING),'*DEBUG')
IF $X ¬= Ø THEN DO; ARGSTRING = DELWORD(ARGSTRING,$X,1); TRACE I
    DEBUG = '*DEBUG'; END
BEGIN:
'COPYFILE' ARGSTRING DEBUG
EXIT:
EXIT RC
```

- CMSQ – this is a carry-over from VM which suppresses the messages from the commands which it executes.
- STATE – another carry over from VM conversion. This generally can be replaced by SYSDSN or LISTDSI commands. I keep the STATE command around because MVS has some quirky things that can trip you up (VSAM versus PDS versus SMS, etc) when you want to know if a dataset exists or is allocated.
- DSNERASE – this carry over from VM would have gone away except that the TSO delete command also has gotchas.
- FPRT – this is not a VM thing, but an MVS print thing. It seemed a good idea at the time to let the output file go to a print queue that does two-up printing in my style. I recommend replacing this command with a simple PRINTDS.

COPYFILE CF simulates the VM/CMS COPYFILE command in a TSO or PC environment. Commands are entered in the format shown below:

```
COPYFILE|CF &INDSN &OUTDSN < *REPlace *APPend *FROM(N) *TO(N) *FOR(N) &OPTS>
                           < *recDLM(nnn) *recMAX(nnn|*) *Print *STats >
```

Where:

- | – means to choose one parameter or the other.
- < > – means keywords or parameters within are optional.
- *REPlace – means the input file will overlay any data in the output file.
- *APPend – means the input file will be appended to any records in output file.
- *FROM(N) – means to start copying records starting at sequence number N. If N is enclosed in quotes or not numeric, N becomes a search value. The copy process starts with the first record containing the search value.
- *TO(N) – means to stop copying records, stopping at sequence number N. If N is enclosed in quotes or not numeric, N becomes a search value. The copy process stops with the first record containing the search value.

- *FOR(N) – means to copy no more than N records to the output file.
- *recDLM(x) – allows input records to be broken up into segments. If 'x' is numeric each input record is broken into records having a length of 'x'. If 'x' is not numeric each input record is divided using 'x' as the separator. This parameter has two uses. First, files with 'UNDEFINED' logical record length can be broken into individual records of any fixed size. Secondly, records containing 'eor' or 'eol' markers can be processed.
- *recMAX(x) – this is making 'ROLLOVER' like files. This field must be numeric, and will limit the number of records that can be written to the output file. If the number of records being written exceeds 'x', that excess number of records is excluded from the beginning of the file. If 'x' is 0 then 'x' is set to the number of input records. This option in conjunction with the *append option can help developers create 'log' files that:

  – Accumulate the most recent information available.

  – Never run out of space.
- &OPTS – if the output file does not exist an output dataset is created modelled after the input file. If you do not want the output dataset modelled after the input file do one of two things:

  1 Allocate the output dataset prior to using this command.

  2 Use one or more of the following optional parameters:

```
*OPMODEL(dataset_name)
*OPDIR(n)      if n is Ø o/p is seq, ¬Ø o/p is PDS.
*OPDASD(unit < volser >)
*OPSPACE(TR|CY|BL priamt < 2ndamt < blksize > >)
```

- *IPDD – if the input and/or output datasets are not standard catalogued *OPDD sequential datasets, you can code the unique ALLOCATE (or JCL) parameters manually. When you invoke COPYFILE, all you need do is refer to the uniquely assigned DD names using the *IPDD and/or*OPDD keyword parameters, in place of the positional input and/or output dataset names. For example, if *IPDD keyword is coded, then COPYFILE reads from the dataset attached to the SYSUT1 DD. While, if the *OPDD keyword is coded, COPYFILE will write to the dataset attached to the SYSUT2 DD. Overriding the default DD is simply

a matter of enclosing the desired DD name in parentheses (ie CF *IPDD(INDD) *OPDD(OUTDD)).

- *Print – after copying the input to output file, print the output file.
- *STats – LMF statistics are not copied along with the data. Enter this keyword to request transferral or creation (if none exist) of the LMF statistics. The user-ids, current date, and time are always reset when *STATS is turned on. This is an MVS-only keyword.

EXAMPLES

To copy one file to another enter:

```
CF 'MHOYY.PRT.COMPARE' SPCL.CMPR
```

If the output dataset does not exist it will be created. To copy the second 500 records to the same output file enter:

```
CF 'MHOYY.PRT.COMPARE' SPCL.CMPR *FROM(500) *FOR(500)
```

To ADD those same 500 records to the same output file enter:

```
CF 'MHOYY.PRT.COMPARE' SPCL.CMPR *FROM(500) *FOR(500) *APPEND
```

To ADD those same 500 records using the VM simulation enter:

```
CF CMSFNI CMSFT * CMSFNO CMSFT * (FROM 500 FOR 500 APPEND)
CF CMSFNI CMSFT * CMSFNO CMSFT * (FROM 500 TO 1000 REPLACE)
```

Note, the MVS dataset names will be ‘CMSFT(CMSFNI)’ and ‘CMSFT(CMSFNO)’. To add new log information to a rollover log file that cannot exceed 3000 records code something similar to what follows. The following example is for a workstation, such as OS/2 or Windows 95:

```
CF CREEPFLG.RPT C:\LOG\CREEPRPT.LOG *APPEND *RECMAX(3000)
```

COPYFILE runs well when submitted to JES as a batch JOB. Below is an example of COPYFILE being used as a replacement to IEBGENER. CF is more flexible than IEBGENER because it handles input/output LRECLs that do not match. An example of the JCL follows, and all the parameters and options listed above can be coded in the PARM= field. Just remember that there are some quite tricky rules governing PARM specification. For instance, the limitation of 100 bytes and that quotation marks must be coded twice.

BATCHFMT

```
//STEPØØ3Ø EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø,REGION=4Ø96K,
// PARM=('%CF *IPDD *OPDD')
//+*********************************************************
//+ Members can be moved between PDS with unlike attributes
//+*********************************************************
//SYSEXEC  DD   DSN=MIRVI.REXX,DISP=SHR
//SYSUT1   DD   DSN=MIRVI.REXX(AAA),DISP=SHR
//SYSUT2   DD   DSN=MIRVI.PANELS(AAA),DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   DUMMY
```

COPYFILE

```
/*********************************************************************/
/* REXXNAME: COPYFILE                              ALL RIGHTS RESERVED   */
/* FUNCTION: Copy sequential files or PDS members via TO/FROM/FOR        */
/*           options.                                                    */
/*********************************************************************/
SYS=ADDRESS(); IF SYS = 'TSO' | SYS = 'MVS' THEN TSO = 1; ELSE TSO = Ø
IF SYS = 'DOS' | SYS = 'KEDIT' THEN DOS = 1; ELSE DOS = Ø
PARSE ARG ARGSTRING; DEBUG=''; $X = (FIND(TRANSLATE(ARGSTRING),'*DEBUG'))
IF $X ¬= Ø THEN DO; ARGSTRING = (DELWORD(ARGSTRING,$X,1)); TRACE I
    DEBUG = '*DEBUG'; END; CMS=Ø; IF ¬TSO & ¬DOS THEN CMS=1
IF WORD(ARGSTRING,1) = '?' THEN SIGNAL DOC
ARGSTRING = TRANSLATE(ARGSTRING)
IF ARGSTRING = '' THEN SIGNAL DOC
QUIET = Ø; X = FIND(ARGSTRING,'*QUIET')
IF X ¬= Ø THEN DO
    ARGSTRING = DELWORD(ARGSTRING,X,1)
    QUIET = 1         /* ¬ */
    END
STATS = Ø; X = FIND(ARGSTRING,'*STATS')
IF X = Ø THEN X = FIND(ARGSTRING,'*STAT')
IF X = Ø THEN X = FIND(ARGSTRING,'*STA')
IF X = Ø THEN X = FIND(ARGSTRING,'*LMF')
IF X = Ø THEN X = FIND(ARGSTRING,'*ST')
IF X ¬= Ø THEN DO
    ARGSTRING = DELWORD(ARGSTRING,X,1)
    STATS = 1         /* ¬ */
    END
PRINT = Ø; X = FIND(ARGSTRING,'*PRINT')
IF X = Ø THEN X = FIND(ARGSTRING,'*PRT')
IF X = Ø THEN X = FIND(ARGSTRING,'*PR')
IF X = Ø THEN X = FIND(ARGSTRING,'*P')
IF X ¬= Ø THEN DO
    ARGSTRING = DELWORD(ARGSTRING,X,1)
    PRINT = 1         /* ¬ */
    END
PEREN = Ø; X = LASTPOS('(',ARGSTRING)  /* NEED TO HANDLE VM OPTION FORMATS */
```

```
IF X ¬= Ø THEN DO
    Y = SUBSTR(ARGSTRING,X)
    /* IF THE FOLLOWING TEST IS PASSED WE HAVE A VM OPTION FIELD. */
    IF LEFT(Y,9) = '(REPLACE)' |,
       LEFT(Y,9) = '(REPLACE ' |,
       LEFT(Y,8) = '(APPEND)'  |,
       LEFT(Y,8) = '(APPEND '  |,
       LEFT(Y,6) = '(REPL)'    |,
       LEFT(Y,6) = '(REPL '    |,
       LEFT(Y,6) = '(FROM '    |,
       LEFT(Y,5) = '(REP)'     |,
       LEFT(Y,5) = '(REP '     |,
       LEFT(Y,5) = '(APP)'     |,
       LEFT(Y,5) = '(APP '     |,
       LEFT(Y,5) = '(FOR '     |,
       LEFT(Y,4) = '(TO '      |,
       LEFT(Y,2) = '( '
       THEN DO
           /* ISOLATE THE OPEN PARENTHESIS FOR VM OPTION FIELDS. */
           ARGSTRING = LEFT(ARGSTRING,X)||' '||SUBSTR(ARGSTRING,X+1)
           PEREN = X  /* SAVE WHERE THE VM PARENTHESIS LIES */
           PERENPFX = WORDS(LEFT(ARGSTRING,X-1)) /* WORDS BEFORE
           PARENTHEIS */
           /* IF WE HAVE OPEN PEREN OPTION, GOTTA DO CLOSE PEREN.  */
           Z = LASTPOS(')',ARGSTRING)
           IF Z = Ø | Z < X THEN LEAVE /* WATCH OUT FOR PDS MEM NAMES */
           ARGSTRING = LEFT(ARGSTRING,Z-1)||' '||SUBSTR(ARGSTRING,Z)
           END
    END
REPLACE = Ø; X = FIND(ARGSTRING,'*REPLACE')
IF X = Ø THEN X = FIND(ARGSTRING,'*REP')
IF X = Ø THEN X = FIND(ARGSTRING,'*R')
IF X = Ø & PEREN > Ø THEN DO
    X = FIND(SUBSTR(ARGSTRING,PEREN),'REPLACE')
    IF X = Ø THEN X = FIND(SUBSTR(ARGSTRING,PEREN),'REPL')
    IF X = Ø THEN X = FIND(SUBSTR(ARGSTRING,PEREN),'REP')
    /* COMPENSATE FOR THE WORDS IN FRONT OF PEREN */
    IF X > Ø THEN X = X + PERENPFX
    END
IF X ¬= Ø THEN DO
    ARGSTRING = DELWORD(ARGSTRING,X,1)
    REPLACE = 1
    END
APPEND = Ø; X = FIND(ARGSTRING,'*APPEND')
IF X = Ø THEN X = FIND(ARGSTRING,'*APP')
IF X = Ø THEN X = FIND(ARGSTRING,'*A')
IF X = Ø & PEREN > Ø THEN DO
    X = FIND(SUBSTR(ARGSTRING,PEREN),'APPEND')
    IF X = Ø THEN X = FIND(SUBSTR(ARGSTRING,PEREN),'APP')
    /* COMPENSATE FOR THE WORDS IN FRONT OF PEREN */
    IF X > Ø THEN X = X + PERENPFX
```

```
    END
IF X ¬= Ø THEN DO
    ARGSTRING = DELWORD(ARGSTRING,X,1)
    APPEND = 1
    END
FOR = ''; EMPTY = Ø; X = FIND(ARGSTRING,'*EMPTY')
IF X ¬= Ø THEN DO
    ARGSTRING = DELWORD(ARGSTRING,X,1)
    EMPTY = 1; FOR = Ø
    END
RECMAX = ''; X = POS('*RECMAX(',ARGSTRING); U = X + Ø8; W = U
IF X = Ø THEN DO
    X = POS('*MAX(',ARGSTRING); U = X + Ø5; W = U
    END
IF X ¬= Ø THEN DO FOREVER    /* PARMREXX */
    Y = POS(')',ARGSTRING,U); IF Y = Ø THEN LEAVE
    Z = POS('(',ARGSTRING,W)  /* CHK FOR *VAL1(*SUB1(X) *SUB2(VAL)) */
    IF Z ¬= Ø & Z < Y & LENGTH(ARGSTRING) > Y
       THEN DO; W = Z+1; U = Y+1; ITERATE; END
    ZS = X + Ø8; ZL = Y - X - Ø8
    RECMAX = STRIP(SUBSTR(ARGSTRING,ZS,ZL))
    IF RECMAX = '*' THEN RECMAX = Ø  /*JUST DON'T MAKE FILE ANY BIGGER*/
    IF DATATYPE(RECMAX) ¬= 'NUM' THEN RECMAX = '' /*DEFENSIVE CODE*/
    ZL = Y - X + 1
    ARGSTRING = DELSTR(ARGSTRING,X,ZL)
    LEAVE
    END
RECDLM = ''; X = POS('*RECDLM(',ARGSTRING); U = X + Ø8; W = U
IF X = Ø THEN DO
    X = POS('*DLM(',ARGSTRING); U = X + Ø5; W = U
    END
IF X ¬= Ø THEN DO FOREVER    /* PARMREXX */
    Y = POS(')',ARGSTRING,U); IF Y = Ø THEN LEAVE
    Z = POS('(',ARGSTRING,W)  /* CHK FOR *VAL1(*SUB1(X) *SUB2(VAL)) */
    IF Z ¬= Ø & Z < Y & LENGTH(ARGSTRING) > Y
       THEN DO; W = Z+1; U = Y+1; ITERATE; END
    ZS = X + Ø8; ZL = Y - X - Ø8
    RECDLM = STRIP(SUBSTR(ARGSTRING,ZS,ZL))
    ZL = Y - X + 1
    ARGSTRING = DELSTR(ARGSTRING,X,ZL)
    LEAVE
    END
IF REPLACE THEN APPEND = Ø
IF APPEND THEN REPLACE = Ø
IF ¬REPLACE & ¬APPEND THEN REPLACE = 1   /* REPLACE IS DEFAULT */
FROM = 1; FROM_IS_KEY = Ø; XFROM = ''
X = POS('*FROM(',ARGSTRING)
IF X ¬= Ø THEN DO 1     /* PARMREXX */
    Y = POS(')',ARGSTRING,X); IF Y = Ø THEN LEAVE
    ZS = X + Ø6; ZL = Y - X - Ø6
    FROM = STRIP(SUBSTR(ARGSTRING,ZS,ZL))
```



```
    IF FROM = '' THEN FROM = 1
    IF DATATYPE(FROM) ¬= 'NUM'
       THEN DO
        XFROM = FROM
        IF LEFT(FROM,1) = "'" THEN PARSE VAR FROM "'" FROM "'"
        IF LEFT(FROM,1) = '"' THEN PARSE VAR FROM '"' FROM '"'
        FROM_IS_KEY = 1
        END
    ZL = Y - X + 1
    ARGSTRING = DELSTR(ARGSTRING,X,ZL)
    END
IF PEREN > Ø THEN DO
    X = FIND(SUBSTR(ARGSTRING,PEREN),'FROM')
    IF X > Ø THEN DO
        X = X + PERENPFX
        FROM = WORD(ARGSTRING,X+1)   /* PICK UP THE ARGUMENT */
        IF FROM = '' THEN FROM = 1
        IF DATATYPE(FROM) ¬= 'NUM'
           THEN DO
            IF LEFT(FROM,1) = "'" THEN PARSE VAR FROM "'" FROM "'"
            IF LEFT(FROM,1) = '"' THEN PARSE VAR FROM '"' FROM '"'
            FROM_IS_KEY = 1
            END
        ARGSTRING = DELWORD(ARGSTRING,X,2)
        END
    END
TO = 1ØØØØØ; XTO = ''
TO_IS_KEY = Ø; X = POS('*TO(',ARGSTRING)
IF X ¬= Ø THEN DO 1     /* PARMREXX */
    Y = POS(')',ARGSTRING,X); IF Y = Ø THEN LEAVE
    ZS = X + Ø4; ZL = Y - X - Ø4
    TO = STRIP(SUBSTR(ARGSTRING,ZS,ZL))
    IF TO = '' THEN TO = 1ØØØØØ
    IF DATATYPE(TO) ¬= 'NUM'
       THEN DO
        XTO = TO
        IF LEFT(TO,1) = "'" THEN PARSE VAR TO "'" TO "'"
        IF LEFT(TO,1) = '"' THEN PARSE VAR TO '"' TO '"'
        TO_IS_KEY = 1
        END
    ZL = Y - X + 1
    ARGSTRING = DELSTR(ARGSTRING,X,ZL)
    END
IF PEREN > Ø THEN DO
    X = FIND(SUBSTR(ARGSTRING,PEREN),'TO')
    IF X > Ø THEN DO
        X = X + PERENPFX
        TO = WORD(ARGSTRING,X+1)   /* PICK UP THE ARGUMENT */
        IF TO = '' THEN TO = 1ØØØØØ
        IF DATATYPE(TO) ¬= 'NUM'
           THEN DO
```

```
            IF LEFT(TO,1) = "'" THEN PARSE VAR TO "'" TO "'"
            IF LEFT(TO,1) = '"' THEN PARSE VAR TO '"' TO '"'
            TO_IS_KEY = 1
            END
        ARGSTRING = DELWORD(ARGSTRING,X,2)
        END
    END
IF FOR = '' THEN FOR = 1ØØØØØ; X = POS('*FOR(',ARGSTRING)
IF X ¬= Ø THEN DO 1     /* PARMREXX */
    Y = POS(')',ARGSTRING,X); IF Y = Ø THEN LEAVE
    ZS = X + Ø5; ZL = Y - X - Ø5
    FOR = STRIP(SUBSTR(ARGSTRING,ZS,ZL))
    ZL = Y - X + 1
    ARGSTRING = DELSTR(ARGSTRING,X,ZL)
    IF FOR = '' THEN LEAVE
    IF DATATYPE(FOR) ¬= 'NUM' THEN SIGNAL ERR1ØØ
    END
IF PEREN > Ø THEN DO
    X = FIND(SUBSTR(ARGSTRING,PEREN),'FOR')
    IF X > Ø THEN DO 1
        X = X + PERENPFX
        FOR = WORD(ARGSTRING,X+1)   /* PICK UP THE ARGUMENT */
        ARGSTRING = DELWORD(ARGSTRING,X,2)
        IF FOR = '' THEN LEAVE
        IF DATATYPE(FOR) ¬= 'NUM' THEN SIGNAL ERR1ØØ
        END
    END
IPDD = ''; X = FIND(ARGSTRING,'*IPDD')
IF X ¬= Ø THEN DO
    ARGSTRING = DELWORD(ARGSTRING,X,1)
    IPDD = 'SYSUT1'    /* USE SAME I/P DDNAME AS IEBGENER */
    END
X = POS('*IPDD(',ARGSTRING)
IF X ¬= Ø THEN DO 1
    Y = POS(')',ARGSTRING,X)    /* FIND THE END OF THE INPUT PARM */
    ZS = X + 6; ZL = Y - X - 6  /* CALC START & LENGTH OF IP PARM */
    IPDD = SUBSTR(ARGSTRING,ZS,ZL)
    IF IPDD = '' THEN IPDD = 'SYSUT1'
    ZL = Y - X + 1              /* CALC LENGTH OF IP PARM TO DROP */
    ARGSTRING = DELSTR(ARGSTRING,X,ZL) /* DROP THE INPUT PARM FLD */
    END
OPDD = ''; X = FIND(ARGSTRING,'*OPDD')
IF X ¬= Ø THEN DO
    ARGSTRING = DELWORD(ARGSTRING,X,1)
    OPDD = 'SYSUT2'    /* USE SAME O/P DDNAME AS IEBGENER */
    END
X = POS('*OPDD(',ARGSTRING)
IF X ¬= Ø THEN DO 1               /* IS THE *OPDD() OPTION USED?  */
    Y = POS(')',ARGSTRING,X)    /* FIND THE END OF THE INPUT PARM */
    ZS = X + 6; ZL = Y - X - 6  /* CALC START & LENGTH OF IP PARM */
    OPDD = SUBSTR(ARGSTRING,ZS,ZL) /* SET THE OPDD PARM VALS      */
```

```
    IF OPDD = '' THEN OPDD = 'SYSUT2'   /* SET DEFAULT IF *OPDD() */
    IF OPDD = '=' THEN OPDD = IPDD
    ZL = Y - X + 1              /* CALC LENGTH OF IP PARM TO DROP */
    ARGSTRING = DELSTR(ARGSTRING,X,ZL) /* DROP THE INPUT PARM FLD */
    END
OPMODEL = ''; X = POS('*OPMODEL(',ARGSTRING); U = X + Ø9; W = U
IF X ¬= Ø THEN DO FOREVER    /* PARMREXX */
    Y = POS(')',ARGSTRING,U); IF Y = Ø THEN LEAVE
    Z = POS('(',ARGSTRING,W)  /* CHK FOR *VAL1(*SUB1(X) *SUB2(VAL)) */
    IF Z ¬= Ø & Z < Y & LENGTH(ARGSTRING) > Y
       THEN DO; W = Z+1; U = Y+1; ITERATE; END
    ZS = X + Ø9; ZL = Y - X - Ø9
    OPMODEL = STRIP(SUBSTR(ARGSTRING,ZS,ZL))
    ZL = Y - X + 1
    ARGSTRING = DELSTR(ARGSTRING,X,ZL)
    LEAVE
    END
OPDIR = ''; X = POS('*OPDIR(',ARGSTRING)
IF X ¬= Ø THEN DO 1               /* IS THE *OPDIR() OPTION USED? */
    Y = POS(')',ARGSTRING,X)    /* FIND THE END OF THE INPUT PARM */
    ZS = X + 7; ZL = Y - X - 7  /* CALC START & LENGTH OF IP PARM */
    OPDIR = SUBSTR(ARGSTRING,ZS,ZL)   /* SET THE OPDIR PARM VALS  */
    IF OPDIR = '' THEN LEAVE
    IF DATATYPE(OPDIR,'NUM') = Ø THEN SIGNAL ERR14Ø
    IF OPDIR > Ø
        THEN OPDIR = 'DSORG(PO) DIR('OPDIR')'
        ELSE OPDIR = 'DSORG(PS) DIR(Ø)'
    ZL = Y - X + 1              /* CALC LENGTH OF IP PARM TO DROP */
    ARGSTRING = DELSTR(ARGSTRING,X,ZL) /* DROP THE INPUT PARM FLD */
    END
OPDASD = ''; X = POS('*OPDASD(',ARGSTRING) /* FIND DASD DEF OS*/
IF X ¬= Ø THEN DO /* IS THE *OPDASD() OPTION USED?*/
    Y = POS(')',ARGSTRING,X)    /* FIND THE END OF THE INPUT PARM */
    ZS = X + 8; ZL = Y - X - 8  /* CALC START & LENGTH OF IP PARM */
    OPDASD = SUBSTR(ARGSTRING,ZS,ZL) /* SET THE CC PARM VALS  */
    OPDASD = TRANSLATE(OPDASD,' ','_,/.;')
    PARSE VAR OPDASD DSDUNIT DSDVOLS
    IF DSDUNIT ¬= '' THEN OPDASD = 'UNIT('DSDUNIT')'
    IF DSDVOLS ¬= '' THEN OPDASD = OPDASD 'VOLUME('DSDVOLS')'
    ZL = Y - X + 1              /* CALC LENGTH OF IP PARM TO DROP */
    ARGSTRING = DELSTR(ARGSTRING,X,ZL) /* DROP THE INPUT PARM FLD */
    END
OPSPACE = ''; X = POS('*OPSPACE(',ARGSTRING); V = 9
IF X = Ø THEN DO; X = POS('*SPACE(',ARGSTRING); V = 7; END
IF X ¬= Ø THEN DO /* IS THE *OPSPAC() OPTION USED?*/
    Y = POS(')',ARGSTRING,X)    /* FIND THE END OF THE INPUT PARM */
    ZS = X + V; ZL = Y - X - V  /* CALC START & LENGTH OF IP PARM */
    OPSPACE = SUBSTR(ARGSTRING,ZS,ZL) /* SET THE OPSPACE PARM VALS  */
    OPSPACE = TRANSLATE(OPSPACE,' ','_,/.;')
    IF DATATYPE(SPACE(SUBWORD(OPSPACE,2),Ø)) ¬= 'NUM' THEN SIGNAL ERR14Ø
    PARSE VAR OPSPACE SPCTYPE SPC1ST SPC2ND SPCBLKS Z
```

```
    IF Z ¬= '' THEN SIGNAL ERR15Ø
    IF LEFT(SPCTYPE,2) = 'TR'
       THEN SPCTYPE = 'TRACKS'
       ELSE IF LEFT(SPCTYPE,2) = 'CY'
       THEN SPCTYPE = 'CYLINDERS'
       ELSE IF LEFT(SPCTYPE,2) = 'BL'
       THEN DO
        IF SPCBLKS = '' THEN SIGNAL ERR15Ø
        IF DATATYPE(SPCBLKS) ¬= 'NUM' THEN SIGNAL ERR15Ø
        SPCTYPE = 'BLOCK('SPCBLKS')'
        END
       ELSE IF DATATYPE(SPCTYPE) = 'NUM'
       THEN SPCTYPE = 'BLOCK('SPCTYPE')'
       ELSE SIGNAL ERR15Ø
    OPSPACE = SPCTYPE 'SPACE('SPC1ST','SPC2ND')'
    ZL = Y - X + 1              /* CALC LENGTH OF IP PARM TO DROP */
    ARGSTRING = DELSTR(ARGSTRING,X,ZL) /* DROP THE INPUT PARM FLD */
    END
BEGIN:
IF CMS | DOS THEN SIGNAL ERRØ8Ø
/* DROP OPTIONS KEYWORD TEST FOLLOWS */
IF PEREN > Ø THEN PARSE VALUE ARGSTRING WITH ARGSTRING ' (' .
PARSE VALUE '' WITH VMIPNAME VMOPNAME . /* INIT VARS TO NULLS */
FMS = 'A AØ A1 A2 * B C D E F G H I J K L M N'
IF WORD(ARGSTRING,1) ¬= '$DSN' & POS("'",SUBWORD(ARGSTRING,1,3)) = Ø &,
   FIND(FMS,WORD(ARGSTRING,3)) ¬= Ø
   THEN DO
    VMIPNAME = SUBWORD(ARGSTRING,1,3)
    ARGSTRING = CMSTOTSO(VMIPNAME) SUBWORD(ARGSTRING,4)
    END
IF WORD(ARGSTRING,4) ¬= '$DSN' & POS("'",SUBWORD(ARGSTRING,4,3)) = Ø &,
   FIND(FMS '=',WORD(ARGSTRING,6)) ¬= Ø
   THEN DO 1
    Z = SUBWORD(ARGSTRING,4,3)
    IF FIND(Z,'=') ¬= Ø & VMIPNAME ¬= ''  /* IE FC IFN IFT A = OFT A */
       THEN DO X = 1 FOR 3
        IF WORD(Z,X) = '='
           THEN VMOPNAME = VMOPNAME WORD(VMIPNAME,X)
           ELSE VMOPNAME = VMOPNAME WORD(Z,X)
        END
       ELSE VMOPNAME = Z
    ARGSTRING = SUBWORD(ARGSTRING,1,3) CMSTOTSO(VMOPNAME) SUBWORD(ARGSTRING,7)
    END
IF WORD(ARGSTRING,1) = '$DSN'
   THEN ARGSTRING = SUBWORD(ARGSTRING,2,1) SUBWORD(ARGSTRING,4)
IF WORD(ARGSTRING,2) = '$DSN'
   THEN ARGSTRING = SUBWORD(ARGSTRING,1,1) SUBWORD(ARGSTRING,3,1),
           SUBWORD(ARGSTRING,5)
/********* CODE IS OBSOLETED BY 'PEREN = Ø' LOGIC IN THE BEGINNING OF PROGRAM
PARSE VALUE SUBWORD(ARGSTRING,3) WITH . ' (' X  /*DROP OPTIONS KEYWORD */
```

```
IF X ¬= '' THEN DO
    ARGSTR = X
    X = FIND(ARGSTR,'APPEND')
    IF X ¬= Ø THEN DO
        ARGSTR = DELWORD(ARGSTR,X,1)
        APPEND = 1
        END
    X = FIND(ARGSTR,'REPLACE')
    IF X ¬= Ø THEN DO
        ARGSTR = DELWORD(ARGSTR,X,1)
        REPLACE = 1
        END
    X = FIND(ARGSTR,'FROM')
    IF X ¬= Ø THEN DO
        FROM = SUBWORD(ARGSTR,X+1)
        ARGSTR = DELWORD(ARGSTR,X,2)
        END
    X = FIND(ARGSTR,'FOR')
    IF X ¬= Ø THEN DO
        FOR = SUBWORD(ARGSTR,X+1)
        ARGSTR = DELWORD(ARGSTR,X,2)
        END
    X = FIND(ARGSTR,')')
    IF X ¬= Ø THEN DO
        ARGSTR = DELWORD(ARGSTR,X,1)
        END
    X = ARGSTR
    IF X ¬= '' THEN SIGNAL ERR11Ø
    END
*****  END OF VM PARM LOGIC.  ***************************************/
IF FIND(FMS,WORD(ARGSTRING,3)) ¬= Ø
   THEN DO
    Z = "'"USERID()"."WORD(ARGSTRING,2)"."WORD(ARGSTRING,1)"'"
    RC = LISTDSI(Z)
    IF RC > 4 THEN Z = "'"USERID(),
        ||"."WORD(ARGSTRING,1)"("WORD(ARGSTRING,2)")'"
    RC = LISTDSI(Z)
    IF RC > 4 THEN Z = "'"USERID(),
        ||"."WORD(ARGSTRING,2)"("WORD(ARGSTRING,1)")'"
    RC = LISTDSI(Z)
    IF RC > 4 THEN Z = "'"USERID(),
        ||"."WORD(ARGSTRING,1)"."WORD(ARGSTRING,2)"'"
    ARGSTRING = Z SUBWORD(ARGSTRING,4)
    END
PARSE VAR ARGSTRING IDSN ODSN .
/* GO WITH CURRENT LINE LOC */
IF IPDSN = '' & IPDD = '' THEN SIGNAL ERRØØ8
IF OPDSN = '' & OPDD = '' THEN SIGNAL ERRØØ9
IF IPDSN = '' & IPDD ¬= '' THEN IPDSN = "DD("IPDD')'
IF OPDSN = '' & OPDD ¬= '' THEN OPDSN = "DD("OPDD')'
IDSN = STRIP(TRANSLATE(TRANSLATE(IDSN),' ',','))
```

```
ODSN = STRIP(TRANSLATE(TRANSLATE(ODSN),' ',','))
IF ODSN = '=' | ODSN = '(=)' THEN ODSN = IDSN
X = POS('(=)',ODSN)
IF X > Ø THEN DO
    IF POS('(',IDSN) > Ø
       THEN PARSE VAR IDSN '(' XMEM ')' .
       ELSE DO  /* IF NO MEM NAME USE LAST QUALIFIER */
        Y = LASTPOS('.',IDSN)
        IF Y = Ø
           THEN XMEM = STRIP(TRANSLATE(IDSN,' ","'"))
           ELSE XMEM = STRIP(TRANSLATE(SUBSTR(IDSN,Y+1),' ","'"))
        END
    ODSN = STRIP(LEFT(ODSN,X-1)||'('XMEM')'SUBSTR(ODSN,X+3))
    END
IF LEFT(ODSN,1) = '('
   THEN DO
    X = POS('(',IDSN)
    IF X = Ø THEN SIGNAL ERR13Ø
    IF LEFT(IDSN,1) = "'"
       THEN ODSN = SUBSTR(IDSN,1,X-1)ODSN"'"
       ELSE ODSN = SUBSTR(IDSN,1,X-1)ODSN
    END
IF LEFT(IDSN,1) ¬= "'" & IPDD = ''
   THEN IDSN = "'"USERID()"."IDSN"'"
IF LEFT(ODSN,1) ¬= "'" & OPDD = ''
   THEN ODSN = "'"USERID()"."ODSN"'"
ODX = Ø; NDX = Ø
/* TWO 'IF' ROUTINES CODED TO MAKE LOGIC EASIER TO READ AND UNDERSTAND. */
IF APPEND & OPDD = ''   /* IS THE APPEND NOT BEING DONE TO A DDNAME?    */
   THEN DO 1            /* ANS: YES        */
    "STATE $DSN" ODSN "DDNAM"
    /* CHECK TO SEE IF THERE IS A FILE TO APPEND */
    IF RC = Ø THEN LEAVE  /* YES THEN READ THE FILE BEFORE WRITING IT */
    APPEND = Ø            /* NO THEN TURN OFF THE APPEND LOGIC.       */
    END
IF APPEND & OPDD ¬= ''  /* IS THE APPEND TO BE DONE TO A DDNAME?      */
   THEN DO 1            /* ANS: YES        */
    RC = LISTDSI(OPDD 'FILE')
    /* CHECK TO SEE IF THERE IS A FILE TO APPEND */
    IF RC = Ø THEN LEAVE  /* YES THEN READ THE FILE BEFORE WRITING IT */
    APPEND = Ø            /* NO THEN TURN OFF THE APPEND LOGIC.       */
    END
IF TSO THEN DO 1
    IF LISTDSN(IDSN) > Ø THEN SIGNAL ERRØ12
    IF IPDD = ''
       THEN DO
        IF STATS THEN DO
            /* THIS LOGIC WILL STATUS A MEMBER IN AN MVS DATASET */
            /* ISPF MUST BE ACTIVE... */
            PARSE VALUE DSNPIECS(IDSN) WITH DSN MEM
            ADDRESS 'ISPEXEC'
```

```
         'CONTROL ERRORS RETURN'
         'LMINIT DATAID(SEE) DATASET('DSN') ENQ(SHRW)'
         IF RC ¬= Ø THEN SIGNAL ERR23Ø
         'LMOPEN DATAID('SEE') ORG(PO) OPTION(INPUT)'
         IF RC ¬= Ø THEN SIGNAL ERR24Ø
         'LMMFIND DATAID('SEE') MEMBER('MEM') STATS(YES)'
         IF RC ¬= Ø THEN SIGNAL ERR245
         'LMCLOSE DATAID('SEE')'
         IF RC ¬= Ø THEN SIGNAL ERR26Ø
         VERSION  = ZLVERS
         MODLEVEL = ZLMOD
         CREATED  =  ZLCDATE
         MODDATE  =  ZLMDATE
         MODTIME  =  ZLMTIME
         CURSIZE  =  ZLCNORC
         INITSIZE = ZLINORC
         MODRECS  =  ZLMNORC
         USER     = ZLUSER
         'LMFREE DATAID('SEE')'
         ADDRESS 'TSO'
         END
     "ALLOCATE DDNAME(CPYDD) DSNAME("IDSN") SHR REUSE"
     IF RC ¬= Ø THEN SIGNAL ERRØ1Ø
     END
IF IPDD = ''    /* SETUP THE APPROPRIATE DDNAME                    */
   THEN CPYDD = 'CPYDD'
   ELSE CPYDD = IPDD
"EXECIO * DISKR" CPYDD "( STEM IDSNLN. FINIS )"
IF RC ¬= Ø THEN SIGNAL ERRØ2Ø
"FREE DDNAME("CPYDD")"
IF RC ¬= Ø THEN SIGNAL ERRØ7Ø
IF IDSNLN.Ø = Ø & ¬EMPTY THEN SIGNAL ERRØ4Ø
/* IF RECMAX = Ø DROP FRONT RECS TO MAKE ROOM FOR NEW RECS */
IF RECMAX ¬= '' THEN IF RECMAX = Ø THEN RECMAX = IDSNLN.Ø
ODX = Ø   /* OUTPUT INDEX */
IF DEBUG ¬= '' THEN TRACE O
DO NDX = 1 FOR IDSNLN.Ø
    IF FROM ¬= ''
       THEN DO  /* CHECK AND SEE IF FROM COND MET */
        IF ¬FROM_IS_KEY & FROM > IDSNLN.Ø THEN SIGNAL ERRØ32
        IF FROM_IS_KEY
           THEN IF POS(FROM,IDSNLN.NDX) = Ø
               THEN ITERATE NDX
      MVS COPY UTILITY THAT ALSO DOES ROLLOVER FILES
```

Customizing edit/browse panels

INTRODUCTION

Have you ever been editing/browsing a dataset and wanted to know which volume it is on? Do you use many different TSOs and are not always sure which system or user-id your current session is using?

For many years IBM had user requests for ISPF to display that information, but they were given a very low priority. Finally, in ISPF 4.4 (which came first with OS/390 Version 1Release 3), it provided a new variable, ZDSVOL, which contains the VOLSER for edit/browse. IBM did not mention it in its list of changes and it did not display it on its own supplied panels. So if you want to see it, you must create modified panel definitions. And if you are making that change, you may as well display the system-id or user-id too. An example of how it could look is shown in Figure 1:

```
File  Edit  Confirm  Menu  Utilities  Test  Help            USER007 on MVSX
_____________________________________________________________________________
EDIT     USER007.DATASET(MEMBER)               [VOL069]   Columns 00001 00072
Command ===>                                                 Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001
000002   This is the text being editted.
000003
000004   At the top right of the panel you can see the TSO userid and
000005   the MVS system id. "USER007 on MVSX".
000006
000007   After the dataset name, the volume is shown  "[VOL069]"
000008
****** **************************** Bottom of Data ****************************
```

Figure 1: Example output

I/O FOR ISPF PANELS

ISPF does not use VLF to reduce I/Os, and two years ago the IBM ISPF development team advised that it has no plans to use it in future. However, ISPF does keep a buffer to store the most recently used panels' definitions. This buffer holds about 5-10 panels depending on their size (according to advice from the ISPF team about ten years

ago); but only panels that are valid for pre-processing are stored in the buffer – only panels that do *not* have a dynamically defined width or depth. Edit and browse need to dynamically determine the width and depth of the text displayed.

WHAT BASE PANELS TO USE

Before Version 4, ISPF used only panels ISREDDE and ISRBROBF for edit and browse. They are simple panels with no defined dynamic area, so they can be stored in the panel buffer – to minimize I/Os. When these panels are used for edit or browse, the program adds the necessary dynamic area for the data edited or browsed. This trick still works in all ISPF versions, so you could use ISREDDE or ISRBROBF as the basis for your own customized panels as long as you are making only simple changes, using the existing field attributes (for example changing the panel heading text).

However, the ISPF *Planning* and *Customizing* manuals recommend that you use panels ISREFR01, ISREFR02, ISREFR03, ISREFR04 or ISRBROB as your base panels for any customizing. These different panels are with/without action bars and with/without highlighting. Each is fully defined with a dynamic area. Therefore, if you are creating special edit or browse panels for a new application, it's generally safest to use these panels as the basis.

In a future article I will present an ISPF application with a customized edit panel, based on the panels described above. However, this article is about modifying the panels used during normal editing or browsing of datasets. Therefore, we must create modified versions of the normal panels.

ISPF Version 4 normally uses panels ISREDDE2, ISREDDE3, ISREDDE4, and ISRBROBA, ISRBROBN for edit and browse. They are fully defined with dynamic areas. The different panels (with/without action bars and with/without highlighting) are used according to the ISPF configuration options chosen in the ISRCONFG module. Usually ISPF is configured to have both action bars and highlighting so ISREDDE2 and ISRBROBA are used, and they are the ones I have customized here. I have also included the panels for listing PDS members (ie ISRBROM, ISREPO01, and ISRUDSM) with the same changes.

CUSTOMIZED PANELS

The customized panels are shown below:

ISREDDE2

```
)PANEL KEYLIST(ISRSPEC,ISR)
/*———————————————————————————————————————————————————————————————*/
/* &ZUSER, &ZSYSID added                                             */
/* &ZDSVOL added                                                     */
/* &ZTITLE has &ZLEVEL removed if it is too long                     */
/* default Scroll changed from PAGE -> CSR                           */
/*———————————————————————————————————————————————————————————————*/
)ATTR DEFAULT(   ) FORMAT(MIX)                                 /*  */
 15 TYPE(AB)
 2A TYPE(ABSL) GE(ON)
 2B TYPE(PT)
 2F TYPE(FP)
 14 TYPE(NT)
 13 TYPE(NEF) PADC(USER)
 16 TYPE(VOI) PADC(USER)
 26 AREA(DYNAMIC) EXTEND(ON) SCROLL(ON) USERMOD('2Ø')
 Ø1 TYPE(DATAOUT) INTENS(LOW)
 Ø2 TYPE(DATAOUT)
     ..
     ..
     ..
PDC DESC(‘Index') MNEM(5) ACTION RUN(TUTOR) PARM('ISR91ØØØ')
)ABCINIT
.ZVARS=EDMHELP
)BODY  EXPAND(//) WIDTH(&ZWIDTH)  CMD(ZCMD)
   File  Edit  Confirm  Menu  Utilities  Test  Help            &ZUSER on
&ZSYSID
 —/-/———————————————————————————-
 Z        Z/ /                                  [&ZDSVOL]  Columns Z     Z
 Command ===> Z/ /                                             Scroll ===> Z
 ZDATA,ZSHADOW/ /
 / /
)INIT
.ZVARS = ‘(ZVMODET ZTITLE ZCL ZCR ZCMD ZSCED)'
  IF (&ZVMODET = ‘VIEW') .HELP = ISR1ØØØØ  /* DEFAULT TUTORIAL NAME */
  ELSE                   .HELP = ISR2ØØØØ  /* DEFAULT TUTORIAL NAME */
  &zpm3 = Ø
  VGET (ZSCED) PROFILE        /* Fill Scroll Vars if       */
  IF (&ZSCED = ‘ ‘) &ZSCED = 'CSR '  /* Blank with CSR instead of PAGE  ******
  &MIXED = TRANS(&ZPDMIX N,EBCDIC *,MIX) /* set mixed format */
  &SCLVMODE = &ZVMODET        /* SAVE VIEW/EDIT MODE INFO  */
  VPUT (SCLVMODE) SHARED
/*VGET (ZDSVOL) SHARED   <=== commented out so &ZDSVOL is valid!!  Ron  ******
  &STITLE = TRUNC(&ZTITLE,' ')                                    /*  Ron  ******
  IF (VER(&STITLE,LEN,GT,29)) &ZTITLE = TRUNC(&ZTITLE,'-')     /*  Ron  ******
```

```
)REINIT
  REFRESH(*)
  IF (&ZVMODET = 'VIEW') .HELP = ISR10000  /* DEFAULT TUTORIAL NAME */
  ELSE                   .HELP = ISR20000  /* DEFAULT TUTORIAL NAME */
)PROC
  REFRESH(*)
  &ZCURSOR = .CURSOR
  &ZCSROFF = .CSRPOS
  VPUT (ZSCED) PROFILE
&ZLVLINE = LVLINE(ZDATA)
)END
```

ISRBROBA

```
)PANEL KEYLIST(ISRSPBC,ISR)
/*———————————————————————————————————————————————————————————————————*/
/* &ZUSER, &ZSYSID added                                                 */
/* &ZDSVOL added                                                         */
/* &ZTITLE has &ZLEVEL removed if it is too long                         */
/* default Scroll changed from PAGE -> CSR                               */
/*———————————————————————————————————————————————————————————————————*/
)ATTR DEFAULT(   ) FORMAT(MIX)                                  /*  */
 29 TYPE(AB)
 Ø4 TYPE(ABSL) GE(ON)
 Ø5 TYPE(PT)
 Ø9 TYPE(FP)
 ØA TYPE(NT)
 13 TYPE(NEF) PADC(USER)
 16 TYPE(VOI) PADC(USER)
 26 AREA(DYNAMIC) EXTEND(ON) SCROLL(ON)
 Ø1 TYPE(DATAOUT) INTENS(LOW)
 Ø2 TYPE(DATAOUT)
     ..
     ..
     ..
PDC DESC('Index') MNEM(1) ACTION RUN(TUTOR) PARM('ISR91ØØØ')
)ABCINIT
.ZVARS=BROHELP
)BODY  EXPAND(//) WIDTH(&ZWIDTH)  CMD(ZCMD)
   Menu  Utilities  Help                                  &ZUSER on
&ZSYSID
 —/-/———————————————————————————————-
 BROWSE   Z/ /                          [&ZDSVOL]  Line Z        Col Z
 Command ===> Z/ /                                    Scroll ===> Z
 ZDATA/ /
 / /
 / /
)INIT
.ZVARS = '(ZTITLE ZLINES ZCOLUMS ZCMD ZSCBR)'
.HELP = ISR1BØØØ
&ZCMD = ' '
VGET (ZSCBR) PROFILE     /* Fill Scroll Vars if       */
```

```
IF (&ZSCBR = ' ')  &ZSCBR  = 'CSR '  /* Blank with CSR      Ron ********
IF (&ZMEMB ¬= ' ') &ZTITLE = '&ZDSNT(&ZMEMB)&ZLEVEL ' /* OZ917Ø8 */
IF (&ZMEMB = ' ')  &ZTITLE = '&ZDSNT&ZLEVEL '
IF (VER(&ZTITLE,LEN,GT,25)) &ZTITLE = TRUNC(&ZTITLE,'-') /* Ron ********
&MIXED = MIX
IF (&ZPDMIX = N) &MIXED = EBCDIC
)REINIT
REFRESH(ZCMD,ZSCBR,ZDATA,ZLINES,ZCOLUMS,ZTITLE)
)PROC
&ZCURSOR = .CURSOR
&ZCSROFF = .CSRPOS
VPUT (ZSCBR) PROFILE     /*                          */
&ZLVLINE = LVLINE(ZDATA)
)END
```

ISRBROM

```
)PANEL KEYLIST(ISRSAB,ISR)
/*———————————————————————————————————————————————————————————————————*/
/* &ZUSER, &ZSYSID added                                               */
/* &ZDSVOL added                                                       */
/* default Scroll changed from PAGE -> CSR                             */
/*———————————————————————————————————————————————————————————————————*/
)ATTR DEFAULT(   ) FORMAT(MIX)                                  /*  */
 ØB TYPE(AB)
 ØD TYPE(PS)
 2D TYPE(ABSL) GE(ON)
 2E TYPE(PT)
 28 TYPE(FP)
 ØA TYPE(NT)
 13 TYPE(NEF) PADC(USER)
 16 TYPE(VOI) PADC(USER)
 26 AREA(DYNAMIC)
 Ø8 TYPE(DATAOUT) PAS(ON) CSRGRP(99)
 Ø9 TYPE(DATAOUT)
 29 AREA(DYNAMIC) EXTEND(ON) SCROLL(ON)
 Ø1 TYPE(DATAIN) CAPS(ON) PADC(&ZMLPAD) PAS(ON)
     ..
     ..
     ..
PDC DESC('Index') MNEM(3) ACTION RUN(TUTOR) PARM('ISR91ØØØ')
)ABCINIT
.ZVARS=MEMLHELP
)BODY  CMD(ZCMD)
   Menu  Functions  Utilities  Help                          &ZUSER on
&ZSYSID
 ———————————————————————————————————————
 BROWSE    Z                                    [&ZDSVOL]  Row Z     of Z
 Command ===> Z                                             Scroll ===> Z
 ZMLCOLD
 ZDATA
```

```
)INIT
.ZVARS = '(ZDSNT ZMLCR ZMLTR ZCMD ZSCML)'
     ..
     ..
IF (&MLC3 = ' ') &MLC3 = 'GREEN'
                                        /* Fill Scroll Vars if       */
IF (&ZSCML = ' ') &ZSCML = 'CSR '       /* Blank with CSR  (Ron)     */
)PROC
VPUT (ZSCML) PROFILE
IF (.CURSOR = ZDATA OR .CURSOR = ZMLCOLD) &ZMSCPOS = &ZCURPOS
ELSE &ZMSCPOS = '0000'
)PNTS
)END
```

ISREPO01

```
)PANEL KEYLIST(ISRSAB,ISR)
/*-------------------------------------------------------------------------*/
/* &ZUSER, &ZSYSID added                                                   */
/* &ZDSVOL added                                                           */
/* default Scroll changed from PAGE -> CSR                                 */
/*-------------------------------------------------------------------------*/
)ATTR DEFAULT(   ) FORMAT(MIX)                                    /*  */
 0B TYPE(AB)
 0D TYPE(PS)
 2D TYPE(ABSL) GE(ON)
 2E TYPE(PT)
 28 TYPE(FP)
 0A TYPE(NT)
 13 TYPE(NEF) PADC(USER)
 16 TYPE(VOI) PADC(USER)
 26 AREA(DYNAMIC)
 08 TYPE(DATAOUT) PAS(ON) CSRGRP(99)
 09 TYPE(DATAOUT)
 29 AREA(DYNAMIC) EXTEND(ON) SCROLL(ON)
 01 TYPE(DATAIN) CAPS(ON) PADC(&ZMLPAD) PAS(ON)
     ..
     ..
     ..
PDC DESC('Index') MNEM(3) ACTION RUN(TUTOR) PARM('ISR91000')
)ABCINIT
.ZVARS=MEMLHELP
)BODY  CMD(ZCMD)
   Menu  Functions  Utilities  Help                          &ZUSER on
&ZSYSID
 ----------------------------------------
 Z        Z                                      [&ZDSVOL]  Row Z     of Z
 Command ===> Z                                               Scroll ===> Z
 ZMLCOLD
 ZDATA
)INIT
.ZVARS = '(ZVMODET ZDSNT ZMLCR ZMLTR ZCMD ZSCML)'
```

```
     ..
     ..
IF (&MLC3 = ' ') &MLC3 = 'GREEN'
                                        /* Fill Scroll Vars if       */
IF (&ZSCML = ' ') &ZSCML = 'CSR '       /* Ron   98/11/17            */
)PROC
VPUT (ZSCML) PROFILE
IF (.CURSOR = ZDATA OR .CURSOR = ZMLCOLD) &ZMSCPOS = &ZCURPOS
ELSE &ZMSCPOS = '0000'
)PNTS
)END
```

ISRUDSM

```
)PANEL KEYLIST(ISRSAB,ISR)
/*-------------------------------------------------------------------------*/
/* &ZUSER, &ZSYSID added                                                   */
/* &ZDSVOL added                                                           */
/* default Scroll changed from PAGE -> CSR                                 */
/*-------------------------------------------------------------------------*/
)ATTR DEFAULT(   ) FORMAT(MIX)                                      /*  */
 0B TYPE(AB)
 0D TYPE(PS)
 2D TYPE(ABSL) GE(ON)
 2E TYPE(PT)
 28 TYPE(FP)
 0A TYPE(NT)
 13 TYPE(NEF) PADC(USER)
 16 TYPE(VOI) PADC(USER)
 26 AREA(DYNAMIC)
 08 TYPE(DATAOUT) PAS(ON)
 09 TYPE(DATAOUT)
 29 AREA(DYNAMIC) EXTEND(ON) SCROLL(ON)
 01 TYPE(DATAIN) CAPS(ON) JUST(LEFT) PADC(&ZMLPAD) PAS(ON)
 02 TYPE(DATAOUT) INTENS(&MLI2) SKIP(ON) COLOR(&MLC2) HILITE(&MLH2)
     ..
     ..
     ..
PDC DESC('Index') MNEM(5) ACTION RUN(TUTOR) PARM('ISR91000')
)ABCINIT
.ZVARS=MEMLHELP
)BODY  CMD(ZCMD)
   Menu  Functions  Confirm  Utilities  Help                  &ZUSER on
&ZSYSID
 ----------------------------------------
 Z                Z                             [&ZDSVOL]  Row Z     of Z
 Command ===> Z                                               Scroll ===> Z
 ZMLCOLD
 ZDATA

)INIT
.ZVARS = '(ZMLHDRV ZDSN ZMLCR ZMLTR ZCMD ZSCML)'
```

```
     ..
     ..
VGET (ZSCML) PROFILE     /* Fill Scroll Vars if       */
IF (&ZSCML = ' ') &ZSCML = 'CSR '            /* Ron 98-10-06 */
)PROC
VPUT (ZSCML) PROFILE
IF (.CURSOR = ZDATA OR .CURSOR = ZMLCOLD) &ZMSCPOS = &ZCURPOS
ELSE &ZMSCPOS = '0000'
)PNTS
)END
```

IMPLEMENTATION

These sample panels are from ISPF 4.5 (comes with OS/390 Version 2 Release 5) but the changes work with any ISPF Version 4. However, versions before ISPF 4.4 (ie before OS/390 Version 1 Release 3) will show blanks instead of the VOLSER.

Panels have attribute bytes that are defined near the start of the panel in the)ATTR section. For example in panel ISREDDE2 X‘14’ is the attribute for TYPE(NT) ‘Normal Text’, and x‘2B’ is for TYPE(PT) ‘Panel Title’ text.

Do not try to use the panels directly from this article – they are not complete and they need the correct attribute bytes to function. Copy your standard panels and make the same changes – preserving all the attribute bytes. Note that you will need to insert an attribute byte before the fields &ZDSVOL and &ZUSER in each panel. Try using whatever has been defined as the attribute for TYPE(NT) before ‘&ZUSER on &ZSYSID’ and the attribute for TYPE(PT) before ‘[&ZDSVOL]’.

I have also changed the default Scroll amount from PAGE to CSR.

If these panels are only for your own use you can put them in your own ISPPLIB concatenation (ahead of the standard panels). If they are to be used by all users, they should be SMP/E usermods; otherwise you might forget to update them for a new version of ISPF.

CONCLUSION

The above changes are very simple to make, but they can be very useful.

Ron Brown
Systems Programmer (Germany)

Batch job ‘elapsed time monitor’

INTRODUCTION

At our site we have many production batch jobs running during the day. It is important that they are turned around as quickly as possible, and any problems spotted (the most obvious sign of this is extended elapsed times – anything over one minute would be regarded as slow). As elapsed times did not appear to be available using SDSF, and to make sure that any delays are quickly noticed, I wrote this simple elapsed time ‘monitor’. It uses an ISPF panel to display the information (jobname, stepname, jobclass, I/Os, and elapsed time) with a program that sets TSO variables. The relevant jobs to be monitored are easily identified by Jobname and Jobclass, and thresholds (set in the program) used to set the display to green (normal), yellow (dubious), or red (most likely a problem). Jobs that are swapped out (probably because of SRM problems or an enqueue) are displayed in pink. These thresholds can be easily modified if unsuitable. A program waits for five seconds after each display). We use a ‘dedicated’ TSO session to run the monitor. It is started using the ‘BATCHMON’ REXX. A sample display is shown in Figure 1.

```
———————— BATCHMON : CURRENT BATCH ELAPSED TIMES ————————-
  17:47:11                                                    22/01/99


===========PRODUCTION ( 1)===========    ============STANDBY ( 0)=============
-Jobname-Stepname C —I/O— -Elapsed-    -Jobname-Stepname C —I/O— -Elapsed-
FPPBMMLS STEP0025 P     295  00:00:05




SAMPLE:      7 / 30       >>>>>                        STARTED AT: 17:46:40
```

Figure 1: Sample BATCHMON display

BATCHTIM REXX

```
/* ============================================================== */
/*  BATCHTIM : Display batch jobs *elapsed* times.                */
/*  This information is not available through SDSF.               */
/* ============================================================== */
   Trace n
/* ------------------------------ */
/*  Display the main panel...                                     */
/* ------------------------------ */
usr = Userid()
fpprmsg = ''

Address "TSO"
x = Outtrap("out.",1Ø,"NOCONCAT")            /* Trap output          */
"PROFILE"                                    /* Get profile settings */
x = Outtrap("OFF")                           /* Do not trap output   */
intercom = 'INTERCOM'                        /* Default setting      */
If out.Ø ¬= Ø Then Do
  Parse Upper Var out.1 . . . . intercom .   /* Save intercom setting*/
  Address "TSO"
  "PROFILE NOINTERCOM"                       /* Please do not disturb*/
End
speclusr = 'MONTR98'
Do Forever                                   /* All this is mine...  */
If usr ¬= speclusr Then                      /* Special user?        */
  sampl = 3Ø
Else Do
  Address "ISPEXEC"
  "ISPEXEC DISPLAY PANEL(BTCHPRMP)"          /* How many samples?    */
  If rc = 8 Then                             /* If PF3 then get out  */
    Call Exit_rtn
  sampl = fpz
End
fpstrt = Time()
sampno = Ø
Do a = 1 to sampl
  Address "TSO"
  fpsw = 'N'
  fpptm. = ''                                /* Reset variables      */
  fpstm. = ''                                /* Reset variables      */
  fplnmsg1 = ''                              /* Reset variables      */
  fplnmsg2 = ''                              /* Reset variables      */
  "BATCHMON"                                 /* Update variables     */
  rett = rc
  If rett = Ø Then Do
    ptm = fpptm.Ø
    xx = Left(ptm,1)
    If xx = 'Ø' Then
      ptm = ' '||Right(ptm,1)                /* Leading blank...     */
    stm = fpstm.Ø
    xx = Left(stm,1)
```

```
    If xx = 'Ø' Then
      stm = ' '||Right(stm,1)
    fplnmsg1 = '===========PRODUCTION ('ptm')==========   '
    fplnmsg1 = fplnmsg1' ============STANDBY ('stm')============='
    fplnmsg2 = '-Jobname-Stepname C —I/O— -Elapsed-'
    fplnmsg2 = fplnmsg2'    -Jobname-Stepname C —I/O— -Elapsed-'
  End
  If rett = 4 Then Do
    fplnmsga = '//////////////////'
    fplnmsgb = '\\\\\\\\\\\\\\\\\\'
    fplnmsg1 = '  No Batch Jobs are currently running  '
    fplnmsg1 = fplnmsga||fplnmsg1||fplnmsgb
    fpsw = 'Y'                          /* Red/revvideo */
  End
  If rett = 12 Then
    fplnmsg1 = '*** Error in "BATCHMON" module - contact Systems ***'
  If rett = 16 Then Do
    fplnmsg1 = '*** "BATCHMON" internal error - sample missed ***'
    fpsw = 'Y'                          /* Red/revvideo */
  End
  sampno = sampno + 1
  /* Update 'status bar' at bottom of panel */
  xx = sampl / 25                       /* Calc samples per 'notch' */
  yy = sampno / xx                      /* Calc 'notch'es so far    */
  fpprogrs = ''
  Do loop = 1 to yy
    fpprogrs = fpprogrs'>'
  End
  fppovrfl = ''
  fpsovrfl = ''
  If fpptm.13 ¬= '' Then                /* See if there's too many  */
    fppovrfl = 'MORE...'                /* entries to display       */
  If fpstm.13 ¬= '' Then                /* Ditto...                 */
    fpsovrfl = 'MORE...'
  Address "ISPEXEC"
  fptime = Time()
  fpdat = Date('E')
  "ISPEXEC CONTROL DISPLAY LOCK"
  "ISPEXEC DISPLAY PANEL(BTCHTIM)"          /* Display main panel   */
  Address "TSO"
  "CALL 'GENERAL.LINKLIB(WAIT5)'"
End                                     /* Loop                     */
fpprmsg = '********* 'sampl 'samples completed *********'
If usr ¬= speclusr Then                 /* Special user?            */
  Call Exit_rtn                         /* No...get out of here */
End                                     /* Do Forever               */
Exit_rtn:
Address "TSO"                           /* Restore profile          */
"PROFILE "intercom
Exit                                    /* Get out of here          */
```

BATCHMON PROGRAM

```
         PRINT NOGEN
*********************************************************************
* MODULE:   BATCHMON                                      GWC 12/11/98  *
*                                                                       *
* FUNCTION: FIND OUT (UNDER TSO) HOW LONG BATCH JOBS HAVE BEEN          *
*           RUNNING, AND RETURN THE VALUE IN TSO VARIABLES 'FPPTM.'     *
*           (PRODUCTION BATCH JOBS) OR 'FPSTM.' (STANDBY BATCH JOBS)    *
*                                                                       *
* METHOD:   SCAN THE ASCBS TO SEE IF ANY FP* JOBS ARE RUNNING. IF     *
*           NOT GO BACK WITH RC=4. IF THERE ARE, EXTRACT THE TIMING     *
*           DATA FROM THE ASCB AND CALCULATE THE ELAPSED TIME, THEN     *
*           CONVERT THIS INTO 'HH:MM:SS' AND PLACE THIS INTO THE        *
*           TSO VARIABLE 'ELAPSTIM'. IF THE JOB IS SWAPPED OUT,THEN     *
*           APPEND THE TIME WITH AN '*'.                                *
*                                                                       *
* OUTPUT:   RETURN CODE  Ø  - TIME VALUE LOCATED, AND RETURNED          *
*                        4  - NO FPP* OR FPX* JOBS RUNNING              *
*                       12  - UNABLE TO UPDATE TSO VARIABLE             *
*                       16  - SET IF ESPIE CONDITION ENCOUNTERED        *
*                                                                       *
* ATTR:     NOT REQUIRED TO BE AUTHORIZED                               *
*           AMODE 31                                                    *
*           RMODE 24                                                    *
*           NON-REENTRANT                                               *
*********************************************************************
BATCHMON CSECT
BATCHMON AMODE 31
BATCHMON RMODE 24
*********************************************************************
* HOUSEKEEPING...                                                   *
*********************************************************************
         BAKR  R14,Ø                   SAVE CALLER DATA ON STACK
         LR    R12,R15                 GET ENTRY POINT
         USING BATCHMON,R12            ESTABLISH ADDRESSABILITY
*********************************************************************
* INITIAL PROCESSING:                                               *
* 1  TRAP ANY NASTY ERRORS SO WE DON'T ABEND AND LEAVE OUR ISPF     *
*    PANEL IN AN UNUSEABLE STATE.                                   *
* 2  GET SSCVT ADDRESS (FOR LATER USE IN OBTAINING JES JOBCLASS).   *
* 3  SCAN ASCB'S TO SEE IF WE HAVE ANY FPP* OR FPX* JOBS RUNNING.   *
*********************************************************************
         ESPIE SET,SETRC16,(9)         <== TRAP THESE INTERRUPTS (ØC9)
         L     R2,16                   GET CVT ADDRESS
         USING CVTMAP,R2               ADDRESSABILITY TO CVT
         L     R1,CVTJESCT             GET A(JES CONTROL TABLE)
         USING JESCT,R1                ADDRESSABILITY
         L     R1,JESSSCT              GET POINTER TO FIRST SSCVT
         USING SSCT,R1                 ADDRESSABILITY
SSCTLOOP DS    ØH
         LTR   R1,R1                   ANY SUBSYSTEM FOUND?
         BZ    GETASCBS                NO...(CAN'T DETERMINE JOBCLASS)
```

```
         CLC   SSCTSNAM(4),=C'JES2'    IS IT OURS? (CALLED 'JES2')
         BE    SAVESSCV                YES..SAVE ADDRESS
         L     R1,SSCTSCTA             NO...GET NEXT ONE
         B     SSCTLOOP                KEEP LOOKING
SAVESSCV DS    ØH
         ST    R1,SSCVTADR             SAVE SSCVT ADDRESS FOR LATER
GETASCBS DS    ØH
         L     R2,CVTASVT              GET ASVT ADDRESS
         DROP  R2
         USING ASVT,R2                 ADDRESSABILITY TO ASVT
         L     R5,ASVTMAXU             GET MAX NUMBER OF ADDR SPACES
         ZAP   FPPSTM,PØØ              INITIALIZE 'STEM' TO ZERO
         ZAP   FPSSTM,PØØ              INITIALIZE 'STEM' TO ZERO
*********************************************************************
* LOOP THROUGH ALL ASCBS...                                         *
*********************************************************************
NEXTASCB DS    ØH
         L     R3,ASVTENTY             R3 -> ASCB
         USING ASCB,R3                 ADDRESSABILITY TO ASCB
         CLC   ASCBASCB,=C'ASCB'       CONSTANT PRESENT?
         BNE   NEXTONE                 NO...SKIP THIS ASCB
         L     R4,ASCBJBNI *JOBNAME*   R4 -> JOBNAME FIELD
         LTR   R4,R4                   POINTER TO JOBNAME PRESENT?
         BZ    NEXTONE                 NO...DO NOT WANT IT
GOTIT    DS    ØH
         CLC   Ø(3,R4),=C'FPP'         PROD BATCH JOB?
         BE    ITSFPP                  YES..SET RELEVANT DETAILS
         CLC   Ø(3,R4),=C'FPX'         STBY BATCH JOB?
         BE    ITSFPS                  YES..SET RELEVANT DETAILS
NEXTONE  DS    ØH
         LA    R2,4(R2)                POINT TO NEXT ASCB
         BCT   R5,NEXTASCB             KEEP LOOKING
         CP    STEM,PØØ                END OF ASCBS - ANY JOBS FOUND?
         BE    NOJOBBYS                NO...GO AND SET RC=4
*********************************************************************
* ALL ASCBS CHECKED AND SOME FPP/FPX JOBS FOUND. NOW UPDATE THE     *
* VARIABLE 'FP?TM.Ø' WITH THE NUMBER OF VARIABLES SET...            *
*********************************************************************
SETSTEM  DS    ØH
         MVC   NAME,=C'FPPTM.'         SET CORRECT VARIABLE NAME
         MVI   NAMESTEM,C'Ø'           SET UP 'FPPTM.Ø' CONSTANT
         MVC   NAMELEN,=F'7'           LENGTH OF 'FPPTM.Ø' CONSTANT
         UNPK  VALJOBN(2),FPPSTM       UNPK COUNT OF VARIABLES
         OI    VALJOBN+1,X'FØ'         SET CORRECT SIGN
         MVC   VALUELEN,=F'2'          LENGTH OF COUNT
         BAL   R9,PUTVAR               PUT THE STEM
         MVC   NAME,=C'FPSTM.'         SET CORRECT VARIABLE NAME
         MVI   NAMESTEM,C'Ø'           SET UP 'FPSTM.Ø' CONSTANT
         MVC   NAMELEN,=F'7'           LENGTH OF 'FPSTM.Ø' CONSTANT
         UNPK  VALJOBN(2),FPSSTM       UNPK COUNT OF VARIABLES
         OI    VALJOBN+1,X'FØ'         SET CORRECT SIGN
         MVC   VALUELEN,=F'2'          LENGTH OF COUNT
         BAL   R9,PUTVAR               PUT THE STEM
```

```
         B     SETRCØ                   GO SET RC=Ø AND RETURN
*************************************************************************
* SET RELEVANT VARIABLE NAME AND UPDATE RELEVANT STEM...                *
*************************************************************************
ITSFPP   DS    ØH
         AP    FPPSTM,PØ1               UPDATE COUNT
         ZAP   STEM,FPPSTM              MOVE COUNT TO WORK FIELD
         MVC   NAME,=C'FPPTM.'          SET CORRECT VARIABLE NAME
         B     CHEKSWAP
ITSFPS   DS    ØH
         AP    FPSSTM,PØ1               UPDATE COUNT
         ZAP   STEM,FPSSTM              MOVE COUNT TO WORK FIELD
         MVC   NAME,=C'FPSTM.'          SET CORRECT VARIABLE NAME
*************************************************************************
* IF ITS SWAPPED OUT WE'LL APPEND AN '*' TO THE TIME...                 *
*************************************************************************
CHEKSWAP DS    ØH
         USING CHNAME,R4
         L     R4,CHCSCBP               GET CSCB ADDRESS
         USING CHAIN,R4
         MVC   VALSTEP(8),CHSTEP        SET JOBNAME IN PARMS
         MVC   VALJOBN(8),CHKEY         SET JOBNAME IN PARMS
         MVI   VALSWAP,C' '             RESET DEFAULT VALUE
         TM    ASCBRCTF,ASCBOUT         SWAPPED OUT?
         BNO   GETIOS                   NO...DON'T WORRY
         MVI   VALSWAP,C'*'             YES..ADD THE '*'
*************************************************************************
* GET THE I/O COUNT FROM THE ASCB. THEY MAY BE ZERO (INITIALLY) OR      *
* UNAVAILABLE, BUT TREAT BOTH THESE POSSIBILITIES THE SAME (IE 'NA')    *
*************************************************************************
GETIOS   DS    ØH
         MVC   VALIOS(7),=XL7'60609561816060' '-N/A-' (LOWER CASE)
         L     R1,ASCBIOSC              GET ASCBIOSC
         LTR   R1,R1                    ANYTHING?
         BZ    GETCLASS                 NO...
         CVD   R1,DWORD                 MAKE IT DECIMAL
         MVC   UNPKFLD2(1Ø),=X'4020204020206B202120'   MASK FOR EDIT
         ED    UNPKFLD2(1Ø),DWORD+4     EDIT IN SIO COUNT
         MVC   VALIOS(7),UNPKFLD2+3     MOVE EDIT SIO COUNT
*************************************************************************
* GET JES2 JOBCLASS. NOTE THAT THIS LATER GIVES US THE OPTION TO SET    *
* DIFFERENT COLOURS FOR DIFFERENT ELAPSED TIMES (THRESHOLDS) DEPEND-    *
* ING ON CLASS...                                                       *
*************************************************************************
GETCLASS DS    ØH
         MVI   VALCLASS,C'*'            SET DEFAULT VALUE
         ICM   R1,15,SSCVTADR           GET SSCVT ADDRESS
         BZ    GETTIME                  IF NONE DON'T BOTHER
         USING SSCT,R1                  ADDRESSABILITY
         L     R1,SSCTSUS2              GET ADDRESS OF HCCT
         USING HCCT,R1                  ADDRESSABILITY
         LH    R6,ASCBASID              GET ASID
         STH   R6,ASID                  SAVE FOR LATER
```

```
         SLL   R6,2                    ASID * 4
         L     R1,CCTHAVT              GET HAVT ADDRESS
         L     R1,Ø(R6,R1)             GET 1ST HASB ADDRESS FOR ASID
         LTR   R1,R1                   ANY?
         BZ    GETTIME                 IF NO HASBS FORGET IT
         USING HASB,R1                 ADDRESSABILITY TO GET A(SJB)
         ICM   R1,15,HSBSJB            GET FIRST SJB ADDRESS
         BZ    GETTIME                 IF NO SJBS FORGET IT
         USING SJB,R1                  ADDRESSABILITY
*********************************************************************
* NOW, A LITTLE LOOP TO FIND THE LAST SSJB FOR THIS ASID...         *
*********************************************************************
SSJBLOOP DS    ØH
         ICM   R6,15,SJBSJB            THE LAST SJB?
         BZ    SJBFOUND                YES..
         LR    R1,R6                   NO...POINT TO NEXT ONE...
         B     SSJBLOOP                ...AND KEEP LOOKING
SJBFOUND DS    ØH
         MVC   VALCLASS(1),SJBJCLAS    MOVE JOBCLASS TO PARMS
*********************************************************************
* CALCULATE TIME. NOTE THAT TIMES ARE 64-BITS, WITH ONLY THE HIGH   *
* ORDER 52-BITS RELEVANT...                                         *
*********************************************************************
GETTIME  DS    ØH
         TIME  STCK,DWORD              GET TOD CLOCK (64 BITS)
         LM    R6,R7,DWORD             LOAD TIME
         SL    R7,ASCBINTS+4           LESS JOB SELECTION TIME
         BC    3,SUBTRTIM              BRANCH IF ANY CARRY
SUBTRTIM DS    ØH
         SL    R6,ASCBINTS             LESS JOB SELECTION TIME
CALCTIME DS    ØH
         SRDL  R6,12                   MOVE OUT IRRELEVANT BITS
         D     R6,=F'1000000'          DIVIDE BY 1000000: R7=SECONDS
*                                      CALCULATE HOURS
         LA    R1,3600                 3600 SECONDS IN AN HOUR
         XR    R6,R6                   CLEAR R6
         DR    R6,R1                   R7 = HOURS, R6=SECS LEFT OVER
         CVD   R7,DWORD                CONVERT HOURS TO DECIMAL
         UNPK  UNPKFLD(5),DWORD+3(5)
         OI    UNPKFLD+4,X'FØ'         SET CORRECT SIGN
         MVC   TIMEHH(2),UNPKFLD+3     MOVE TO HH AREA
*                                      CALCULATE MINUTES/SECONDS
         LA    R1,6Ø                   GET SECONDS IN A MINUTE
         SRDA  R6,32                   SECS LEFT OVER TO R7, CLEAR R6
         DR    R6,R1                   R7=MINUTES, R6=SECS LEFT OVER
         CVD   R7,DWORD                R7 = MINUTES
         UNPK  UNPKFLD(5),DWORD+3(5)
         OI    UNPKFLD+4,X'FØ'         SET CORRECT SIGN
         MVC   TIMEMM(2),UNPKFLD+3     MOVE TO MM AREA
         CVD   R6,DWORD                R6 = SECONDS
         UNPK  UNPKFLD(5),DWORD+3(5)
         OI    UNPKFLD+4,X'FØ'         SET CORRECT SIGN
         MVC   TIMESS(2),UNPKFLD+3     MOVE TO SS AREA
```

```
         MVC   VALTIME(9),TIME         SET TIME IN PARMS
         CP    STEM,PØ9                GREATER THAN 9?
         BH    TWODIGIT                YES..REQUIRES A 2-DIGIT STEM
         MVC   NAMELEN,=F'7'           LENGTH OF 'FP?TM.X'
         UNPK  NAMESTEM(1),STEM        UNPK STEM NUMBER (LESS THAN 1Ø)
         OI    NAMESTEM,X'FØ'          SET CORRECT SIGN
         B     PUTIT
TWODIGIT DS    ØH
         MVC   NAMELEN,=F'8'           LENGTH OF 'FP?TM.XX'
         UNPK  NAMESTEM(2),STEM        UNPK STEM NUMBER (1Ø OR MORE)
         OI    NAMESTEM+1,X'FØ'        SET CORRECT SIGN
PUTIT    DS    ØH
         MVC   VALUELEN,=A(VALUELN)    SET LENGTH OF VALUE
         BAL   R9,PUTVAR               GO AND SET THE VARIABLE
*********************************************************************
* NOW UPDATE THE 'COLOUR' VARIABLE (FOR THE PANEL DISPLAY)...       *
* CURRENT THRESHOLDS/COLOURS ARE:                                   *
*  SWAPPED OUT                          PINK                        *
*  ELAPSED TIME UNDER 1 MINUTE          GREEN                       *
*  ELAPSED TIME OVER 1 MINUTE           YELLOW                      *
*  ELAPSED TIME OVER 1 MINUTE 3Ø SECONDS    RED                     *
*********************************************************************
         MVC   NAME+3(3),=C'CLR'       CHANGE THE VARIABLE NAME
         CLI   VALSWAP,C'*'            NEED SPECIAL COLOUR?
         BNE   CHKCOLR                 NO...
         MVC   VALJOBN(6),=C'PINK  '   'SWAPPED OUT' COLOUR
         B     PUTIT2                  SET COLOUR
CHKCOLR  DS    ØH
         MVC   VALJOBN(6),=C'GREEN '   DEFAULT COLOUR
         CLC   TIMEMM,=C'ØØ'           ARE MINUTES ZERO?
         BE    PUTIT2                  YES..SET DEFAULT
         CLC   TIMEMM,=C'Ø1'           MORE THAN 1 MINUTE?
         BH    ITSRED                  YES..SET COLOUR TO RED
         CLC   TIMESS,=C'3Ø'           NO...MORE THAN 1 MINUTE 3Ø SEC?
         BH    ITSRED                  YES..THAT IS RED AS WELL
         MVC   VALJOBN(6),=C'YELLOW'   MEDIUM WARNING
         B     PUTIT2                  YES..SET DEFAULT
ITSRED   DS    ØH
         MVC   VALJOBN(6),=C'RED   '   HIGH WARNING
PUTIT2   DS    ØH
         MVC   VALUELEN,=F'6'          LENGTH OF COLOUR
         BAL   R9,PUTVAR               GO AND SET THE VARIABLE
         B     NEXTONE                 GET NEXT ASCB (CHECK THEM ALL)
*********************************************************************
* CALL 'IKJCT441' TO UPDATE TSO VARIABLE...                         *
*********************************************************************
PUTVAR   DS    ØH
         LINK  EP=IKJCT441,            PUT VALUE INTO VARIABLE         X
               PARAM=(ECODE,                                           X
               NAMEPTR,                                                X
               NAMELEN,                                                X
               VALUEPTR,                                               X
```

```
              VALUELEN,                                               X
              TOKEN),                                                 X
              VL=1
         LTR  R15,R15                 OK?
         BZR  R9                      YES..RETURN FROM ROUTINE
         B    SETRC12                 NO...SET RC=12
*********************************************************************
* SET RELEVANT RETURN CODE BEFORE EXIT...                           *
*********************************************************************
SETRCØ   DS   ØH
         XR   R15,R15                 SUCCESS!
         B    GO_BACK
NOJOBBYS DS   ØH
         LA   R15,4                   EXPECTED JOB(S) NOT RUNNING
         B    GO_BACK
SETRC12  DS   ØH
         LA   R15,12                  UNABLE TO UPDATE TSO VARIABLE
         B    GO_BACK
SETRC16  DS   ØH
         LA   R15,16                  SET IF ESPIE TRIGGERED
         B    GO_BACK
*********************************************************************
* RETURN WITH RELEVANT RC...                                        *
*********************************************************************
GO_BACK  DS   ØH
         PR   ,
*********************************************************************
* WORK AREAS, EQUATES, ETC...                                       *
*********************************************************************
         LTORG
DWORD    DS   D
SSCVTADR DC   F'Ø'
ASID     DS   H
UNPKFLD  DS   CL5
UNPKFLD2 DS   CL1Ø
TIME     DC   C'HH:MM:SS '
TIMEHH   EQU  TIME+Ø,2
TIMEMM   EQU  TIME+3,2
TIMESS   EQU  TIME+6,2
STEM     DC   PL2'Ø'
FPPSTM   DC   PL2'Ø'
FPSSTM   DC   PL2'Ø'
PØØ      DC   PL1'Ø'
PØ1      DC   PL1'1'
PØ9      DC   PL1'9'
*-------------------------------------------------------------------*
* IKJCT441 PARMLIST (TSO VARIABLE ACCESS ROUTINE)...                *
*-------------------------------------------------------------------*
NAME     DC   C'FPXTM.'               TSO VARIABLE NAME
NAMESTEM DS   XL2                     VARIABLE STEM
NAMELN   EQU  *-NAME                  VARIABLE (INCL STEM) LENGTH
NAMELEN  DC   A(NAMELN)                  "        "
VALJOBN  DC   CL8' '                  JOBNAME
```

```
VALSWAP  DC    CL1' '                  SWAPPED OUT INDICATOR
VALSTEP  DC    CL8' '                  JOBSTEP
         DC    CL1' '
VALCLASS DC    CL1' '                  JOBCLASS
         DC    CL1' '                  BLANK!
VALIOS   DC    CL7' '                  SIOS (TO ' 99,999')
         DC    CL2' '
VALTIME  DC    CL8' '                  ELAPSED TIME
VALUELN  EQU   *-VALJOBN               VALUE LENGTH
VALUELEN DC    A(VALUELN)                "      "
NAMEPTR  DC    A(NAME)                 POINTER TO VARIABLE NAME
VALUEPTR DC    A(VALJOBN)              POINTER TO VARIABLE VALUE
TOKEN    DC    F'Ø'                    TOKEN (UNUSED HERE)
ECODE    DC    A(TSVEUPDT)             ENTRY CODE FOR 'SET'
*        DSECTS HERE...
         CVT   DSECT=YES
         IHAASVT
         IHAASCB
         IEECHAIN
         IRAOUCB
         IKJTSVT
         IEFJESCT
         IEFJSCVT
        #HCCT
        #SJB
        #HASB
        #HASPEQU
        #HFAME
        #XECB
        #SCAT
        #TQE
         END
```

WAIT5 PROGRAM

```
*************************************************************************
* PROGRAM TO WAIT FOR 5 SECONDS (USED BY 'BATCHTIM' EXEC)               *
*************************************************************************
WAIT5    CSECT
         BAKR  14,Ø                    SAVE CALLER DATA ON STACK
         USING WAIT5,12                ADDRESSABILITY
         LR    12,15                   LOAD BASE
         STIMER WAIT,BINTVL=SEC5       WAIT 5 SECONDS
         XR    15,15                   R15=Ø
         PR    '                       RESTORE CALLER DATA AND RETURN
SEC5     DC    F'5ØØ'                  5 SECONDS
         LTORG
         END
```

BTCHPRMP PANEL

```
)ATTR
      \ TYPE(OUTPUT) INTENS(LOW) COLOR(YELLOW) HILITE(BLINK)
)BODY WIDTH(8Ø) EXPAND(@@)
+@-@+BATCMON : CURRENT BATCH ELAPSED TIME(S) @-@
%
%
%
%
%
%          o+Enter number of samples to monitor for (1 - 999):
%                            ===>_fpz+
%
+           Warning: this monitor will lock up this TSO session.
+           1ØØ samples will take approximately 8 minutes...
+
+
+                \fpprmsg                                    +
+
+
%                          PF3+-%Leave This Option
)PROC
  VER (&FPZ,NB)
  VER (&FPZ,NUM)
)END
```

BTCHTIM PANEL

```
)ATTR
   /* ********************************************************** */
   /* All fields and constants within the 'BODY' section of this  */
   /* panel MUST be in UPPER CASE (this is because all of the     */
   /* lower-case letters have been used for the field attributes  */
   /* ********************************************************** */
      $ TYPE(OUTPUT) INTENS(HIGH) JUST(LEFT)
      ¬ TYPE(OUTPUT) INTENS(HIGH) JUST(RIGHT)
      # TYPE(OUTPUT) INTENS(HIGH) COLOR(BLUE) CAPS(OFF)
      ! TYPE(TEXT)   INTENS(LOW)   SKIP(ON)
      Y TYPE(OUTPUT) INTENS(HIGH) COLOR(BLUE) CAPS(OFF)
      _ TYPE(OUTPUT) INTENS(HIGH) COLOR(GREEN) CAPS(OFF)
      \ TYPE(TEXT)   INTENS(LOW) COLOR(GREEN) HILITE(REVERSE)
      a TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPPCLR1)
      b TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPPCLR2)
      c TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPPCLR3)
      d TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPPCLR4)
      e TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPPCLR5)
      f TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPPCLR6)
      g TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPPCLR7)
      h TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPPCLR8)
      i TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPPCLR9)
      j TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPPCLR1Ø)
```

```
      k TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPPCLR11)
      l TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPPCLR12)
      m TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPPCLR13)
      n TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPSCLR1)
      o TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPSCLR2)
      p TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPSCLR3)
      q TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPSCLR4)
      r TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPSCLR5)
      s TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPSCLR6)
      t TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPSCLR7)
      u TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPSCLR8)
      v TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPSCLR9)
      w TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPSCLR1Ø)
      x TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPSCLR11)
      y TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPSCLR12)
      z TYPE(OUTPUT) INTENS(LOW) CAPS(OFF) COLOR(&FPSCLR13)
)BODY WIDTH(8Ø) EXPAND(@@)
+@-@+BATCHMON : CURRENT BATCH ELAPSED TIME(S) @-@
+ $FPTIME  +
$FPDAT   +
+
+
+
+
YFPLNMSG1                                                              +
#FPLNMSG2                                                              +
aFPPTM.1                      + nFPSTM.1                               +
bFPPTM.2                      + oFPSTM.2                               +
cFPPTM.3                      + pFPSTM.3                               +
dFPPTM.4                      + qFPSTM.4                               +
eFPPTM.5                      + rFPSTM.5                               +
fFPPTM.6                      + sFPSTM.6                               +
gFPPTM.7                      + tFPSTM.7                               +
hFPPTM.8                      + uFPSTM.8                               +
iFPPTM.9                      + vFPSTM.9                               +
jFPPTM.1Ø                     + wFPSTM.1Ø                              +
kFPPTM.11                     + xFPSTM.11                              +
lFPPTM.12                     + yFPSTM.12                              +
mFPPTM.13                     + zFPSTM.13                              +
$FPPOVRFL       +                       $FPSOVRFL
+
+
+
%SAMPLE:¬SAMPNO%/$SAMPL+ \ _FPPROGRS                \ +  %STARTED
AT:$FPSTRT  +
)INIT
  IF (&FPSW = 'Y')
    .ATTRCHAR(Y) = 'COLOR(RED) HILITE(REVERSE)'
)PROC
)END
```

Determining the LMOD date of load modules

THE PROBLEM

In the MVS environment, although we can easily ascertain when a non-load member (such as JCL) was last updated by using the ISPF 3.4 dataset list utility panel, we are unable to see the creation date of a load module. The creation date is the same as the link-edited date.

THE SOLUTION

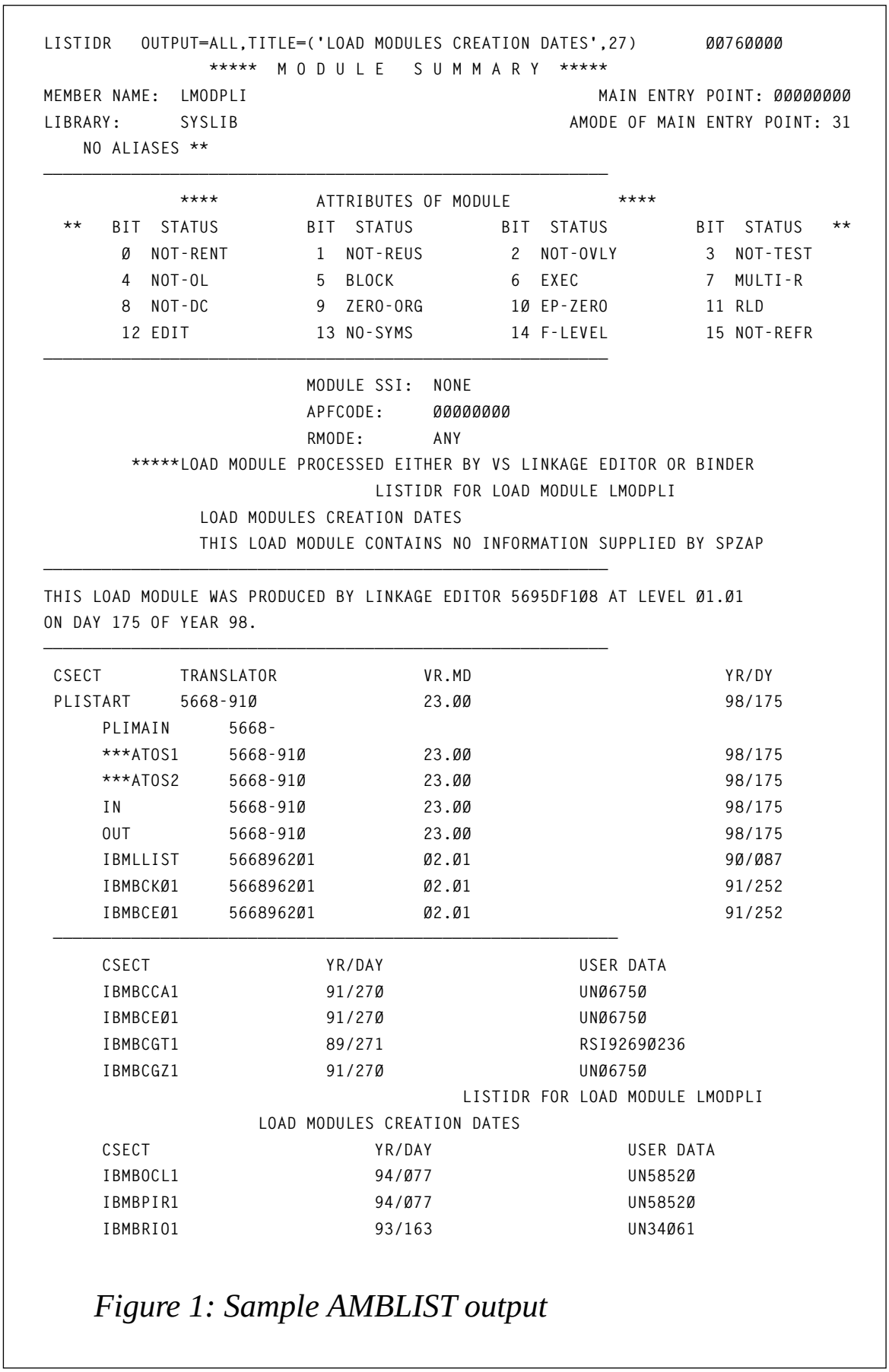
To alleviate this problem, I wrote a CLIST and its accompanying PL/I program used in a JES2 procedure that determines these dates. From now on we will refer to these creation dates as LMOD dates.

The AMBLIST service aid program in MVS is the essential backbone of the work. The service is used via a JCL with LISTIDR control statement. It lists all CSECT (Control Section) identification records for all modules. So, it lists all modules with their LMOD dates in Julian date format as well. IDR stands for IDentification Records. Sample output from this utility is shown in Figure 1.

HOW IT WORKS

Once the CLIST is run, the user is asked to enter a load library name (partitioned organization dataset name). Then the LMOD dates of the load library members will appear on the screen.

This way instead of having no information concerning how old a load module is, we can easily find out the LMOD dates for each member. You will be able to see, not only the LMOD dates of your own load libraries, but also those of system load libraries. For example you can even run this utility for the system dataset SYS1.LPALIB. For example, whilst browsing the output for the dataset SYS1.LPALIB, I came across some modules dating back to 1978 (such as some modules beginning with ‘IGG’).



```
LISTIDR   OUTPUT=ALL,TITLE=('LOAD MODULES CREATION DATES',27)       00760000
               *****  M O D U L E   S U M M A R Y  *****
MEMBER NAME:  LMODPLI                                MAIN ENTRY POINT: 00000000
LIBRARY:      SYSLIB                               AMODE OF MAIN ENTRY POINT: 31
    NO ALIASES **
_____________________________________________________________

             ****          ATTRIBUTES OF MODULE          ****
  **   BIT  STATUS         BIT  STATUS         BIT  STATUS         BIT  STATUS   **
       0  NOT-RENT         1  NOT-REUS         2  NOT-OVLY         3  NOT-TEST
       4  NOT-OL           5  BLOCK            6  EXEC             7  MULTI-R
       8  NOT-DC           9  ZERO-ORG         10 EP-ZERO          11 RLD
       12 EDIT             13 NO-SYMS          14 F-LEVEL          15 NOT-REFR
_____________________________________________________________

                         MODULE SSI:  NONE
                         APFCODE:     00000000
                         RMODE:       ANY
        *****LOAD MODULE PROCESSED EITHER BY VS LINKAGE EDITOR OR BINDER
                                LISTIDR FOR LOAD MODULE LMODPLI
              LOAD MODULES CREATION DATES
              THIS LOAD MODULE CONTAINS NO INFORMATION SUPPLIED BY SPZAP
_____________________________________________________________

THIS LOAD MODULE WAS PRODUCED BY LINKAGE EDITOR 5695DF108 AT LEVEL 01.01
ON DAY 175 OF YEAR 98.
_____________________________________________________________

 CSECT        TRANSLATOR              VR.MD                          YR/DY
 PLISTART     5668-910                23.00                          98/175
      PLIMAIN      5668-
      ***ATOS1     5668-910            23.00                          98/175
      ***ATOS2     5668-910            23.00                          98/175
      IN           5668-910            23.00                          98/175
      OUT          5668-910            23.00                          98/175
      IBMLLIST     566896201           02.01                          90/087
      IBMBCK01     566896201           02.01                          91/252
      IBMBCE01     566896201           02.01                          91/252
 _____________________________________________________________

      CSECT                 YR/DAY                  USER DATA
      IBMBCCA1              91/270                  UN06750
      IBMBCE01              91/270                  UN06750
      IBMBCGT1              89/271                  RSI92690236
      IBMBCGZ1              91/270                  UN06750
                                          LISTIDR FOR LOAD MODULE LMODPLI
                     LOAD MODULES CREATION DATES
      CSECT                       YR/DAY                  USER DATA
      IBMBOCL1                    94/077                  UN58520
      IBMBPIR1                    94/077                  UN58520
      IBMBRIO1                    93/163                  UN34061
```

Figure 1: Sample AMBLIST output

For very large load libraries, it is much better to use only the JCL part of the work without using the CLIST. This is because it can be quite time-consuming to use the CLIST.

With minor changes, you can adapt this work for your own requirements – for example for change management purposes, you can keep track of module changes and update frequencies in a specific production load library.

After extracting the required portions of the output of the AMBLIST utility by using the ISPF search-for utility and one PL/I program, the final result is directed to a sequential dataset. As a last step, it is browsed and presented to the user.

OTHER AMBLIST FUNCTIONALITY?

The AMBLIST service aid can also be used in determining the AMODE and RMODE of a load module. The module summary produced by the LISTLOAD control statement contains the AMODE of the main entry point and the AMODE of each alias, as well as the RMODE specified for the load module. Load module symbols can be printed using the LISTLOAD function, specifying the OUTPUT=XREF option. Object module symbols can be printed using the LISTOBJ function of the service aid program.

You can refer to MVS/ESA SP Version 5 *Diagnosis: Tools and Service Aids* manual for more information concerning AMBLIST.

THE ENVIRONMENT

We use MVS 5.2.2 running on an ES/9000 580 mainframe. The versions of other products are as follows: ISPF Version 4 R.2, DFSORT R13, PL/I Version 2 R.3, JES2 Version 5.2.

EXAMPLE

Upon giving as input SDIAGAS. USER. LOAD, an output something like that shown in Figure 2 would be displayed on the screen.

```
 ------------------------------------------------
 BROWSE    SDIAGAS.LMOD.CHG                        Line 00000000 Col 001 079
 Command ===>                                            Scroll ===> CSR
******************************** Top of Data ********************************
 LMODPLI  98175  24.JUNE     .1998 SDIAGAS.USER.LOAD
 MERGE    98170  19.JUNE     .1998 SDIAGAS.USER.LOAD
 DENE2    98168  17.JUNE     .1998 SDIAGAS.USER.LOAD
 MARTA    98152  01.JUNE     .1998 SDIAGAS.USER.LOAD
 MARTAPV  98149  29.MAY      .1998 SDIAGAS.USER.LOAD
 ATALAY   98148  28.MAY      .1998 SDIAGAS.USER.LOAD
 HISTORY2 98148  28.MAY      .1998 SDIAGAS.USER.LOAD
 VILLAR   98146  26.MAY      .1998 SDIAGAS.USER.LOAD
 DIRECT   98141  21.MAY      .1998 SDIAGAS.USER.LOAD
 PDS      98141  21.MAY      .1998 SDIAGAS.USER.LOAD
 GUL      98138  18.MAY      .1998 SDIAGAS.USER.LOAD
 HISTORY1 92273  29.SEPTEMBER.1992 SDIAGAS.USER.LOAD
 TAPECOPY 91100  10.APRIL    .1991 SDIAGAS.USER.LOAD
 GRAPH    90108  18.APRIL    .1990 SDIAGAS.USER.LOAD
 REBLOCK  88004  04.JANUARY  .1988 SDIAGAS.USER.LOAD
 TAPEMAP  86291  18.OCTOBER  .1986 SDIAGAS.USER.LOAD
 TAPEMAP2 84202  20.JULY     .1984 SDIAGAS.USER.LOAD
 TAPESCAN 82329  25.NOVEMBER .1982 SDIAGAS.USER.LOAD
******************************** Bottom of Data ********************************
```

Figure 2: Sample output screen

CLIST : SDICLMOD

```
PROC Ø
/*********************************************************************/
/*                                                                   */
/* Clist    : Sdiclmod                                               */
/* Function : This CLIST is the main part of the work. It  gets  the */
/*            name of load library from the user, prepares JCL and   */
/*            submits it, then present the final result file to  the */
/*            user by means of the ISPF BROWSE DATASET command.      */
/*                                                                   */
/*********************************************************************/
CONTROL MAIN NOCONLIST NOMSG NOFLUSH
SET &SYSMSG = OFF
/*********************************************************************/
/* Get input from the user.                                          */
/*********************************************************************/
WRITE Please enter the load library of which LMODDATES you want to see
READ &KITAPLIK
/*********************************************************************/
/* Verify whether it is a cataloged dataset.                         */
/*********************************************************************/

IF &SYSDSN('&KITAPLIK') ¬= OK THEN +
      DO
        WRITE There is no such library in the system.
        EXIT
      END
/*********************************************************************/
/* If it is catalogued, then check out that it is a load library.    */
/* (If record format is 'U' then it's considered as a load library.) */
/*********************************************************************/

LISTDSI '&KITAPLIK'
IF &SYSRECFM ¬= U THEN +
                       DO
                         WRITE Sorry, the dataset you entered is not +
a load library.
                         EXIT
                       END
/*********************************************************************/
/* Allocate the sequential file in which LMODDATES of library members */
/* will be put.                                                       */
/*********************************************************************/

SET &SONUC  = &SYSUID..LMOD.CHG
IF &SYSDSN('&SONUC') ¬= OK THEN +

   DO
   FREE   DA('&SONUC')
   ALLOC  FI(SON) DA('&SONUC') NEW SPACE(1,1) TRACKS VOL(SYSDA1)  +
```

```
   BLKSIZE(27966) LRECL(79) DSORG(PS) RECFM(F,B)
   END

FREE DA('&SONUC')
ALLOC  DA('&SONUC') SHR REUSE

WRITE Please wait just a few seconds. Job is being submitted. It may +
 take more time for big load libraries.

/*********************************************************************/
/* The procedure LMODPROC is invoked. At the end of run of this      */
/* procedure, the result file is ready to be browsed by the user.    */
/* &SONUC is the sequential result file.                             */
/*********************************************************************/

SUBMIT * END(SS)

 //&SYSUID.U JOB (&SYSUID),CLASS=A,
 //             MSGCLASS=X,MSGLEVEL=(1,1)
 //PERNAS   EXEC LMODPROC,SONUC='&SONUC',
 // LOAD='&KITAPLIK'
 SS

/*********************************************************************/
/* Wait here until the above job is finished. Then browse the result */
/* file.                                                             */
/*********************************************************************/

MARTA:ISPEXEC CONTROL ERRORS RETURN

IF &ZFBROWS = &Z THEN -
         ISPEXEC CONTROL NONDISPL END
ISPEXEC BROWSE DATASET('&SONUC')

SET &RC = &LASTCC
ISPEXEC CONTROL ERRORS CANCEL
   IF &RC = 12 THEN  GOTO MARTA

/*********************************************************************/
/* Sequential result file is presented to the user. User can keep    */
/* browsing it until PFØ3 key id is pressed.                         */
/*********************************************************************/

ISPEXEC BROWSE DATASET('&SONUC')
FREE DA('&SONUC')
DELETE  '&SONUC'
END
```

LMODPROC

```
//CHANGE      PROC SONUC=,LOAD=
//*******************************************************************
//* Procedure  : Lmodproc
//* Function   : Extracts the LMODDATES of the any load library
//*              of which name is given by the user.
//* Parameters :
//*
//*     1 - LOAD :Library given by the user.               (As INPUT )
//*     2 - SONUC:Result file to be presented to the user  (As OUTPUT)
//*         Its format is userid.LMOD.CHG
//*******************************************************************
//*
//*******************************************************************
//*  Execute the Amblist service aid and get all the load module
//*  information.
//*******************************************************************
//STEP1        EXEC PGM=AMBLIST
//SYSPRINT     DD   DSN=&&IDR1,DISP=(,PASS),UNIT=SYSDA,
// SPACE=(CYL,(1,1)),DCB=(LRECL=121,BLKSIZE=27951,RECFM=FBA)
//SYSLIB       DD   DISP=SHR,DSN=&LOAD
//LOADLIB      DD   DISP=SHR,DSN=&LOAD
//SYSIN        DD   DISP=SHR,DSN=SDID.MVS.LIB.DATA(CTRL1),FREE=CLOSE
//*
//*******************************************************************
//*  Extract the necessary portions of the Amblist output by using
//*  ISPF Batch Search-for utility.
//*******************************************************************
//*
//STEP2        EXEC PGM=ISRSUPC,PARM='DELTAL,SRCHCMP,ANYC'
//SYSPRINT     DD   SYSOUT=*
//NEWDD        DD   DSN=&&IDR1,DISP=(OLD,DELETE)
//OUTDD        DD   DSN=&&IDR2,DISP=(,PASS),UNIT=SYSDA,
// SPACE=(CYL,(1,1)),DCB=(LRECL=133,BLKSIZE=3325,RECFM=FBA)
//SYSIN        DD   DISP=SHR,DSN=SDID.MVS.LIB.DATA(CTRL2),FREE=CLOSE
//*
//*******************************************************************
//*  Build the result file by using the PL/I program (Lmodpli). Here,
//*  the dates in the format of both Julian and Normal is written to
//*  the result file. Input given by the user is passed to the PL/I
//*  program.
//*******************************************************************
//*
//STEP3        EXEC PGM=LMODPLI,PARM='&LOAD'
//STEPLIB      DD   DISP=SHR,DSN=SDIAGAS.USER.LOAD
//SYSPRINT     DD   SYSOUT=*
//GO.IN        DD   DSN=&&IDR2,DISP=(OLD,DELETE)
//GO.OUT       DD   DISP=SHR,DSN=&SONUC
//*
//*******************************************************************
//*  Sort the result file by date.
//*******************************************************************
```

```
//*

//STEP4        EXEC PGM=SORT
//SYSOUT       DD   SYSOUT=*
//SORTIN       DD   DISP=SHR,DSN=&SONUC
//SORTOUT      DD   DISP=SHR,DSN=&SONUC
//SORTWKØ1     DD   UNIT=SYSDA,SPACE=(CYL,2Ø)
//SORTWKØ2     DD   UNIT=SYSDA,SPACE=(CYL,2Ø)
//SYSIN        DD   DISP=SHR,DSN=SDID.MVS.LIB.DATA(CTRL3),FREE=CLOSE
//*

AMBLIST SERVICE AID Control Statement (CTRL1)

LISTIDR   OUTPUT=ALL,TITLE=('LOAD MODULES CREATION DATES',27)

ISPF SEARCH UTILITY Control Statement (CTRL2)

SRCHFOR 'MEMBER NAME'
SRCHFOR 'OF YEAR'

DFSORT Control Statement (CTRL3)

SORT FIELDS=(11,5,D),FORMAT=BI
RECORD TYPE=F,LENGTH=17
END
```

LMODPLI

```
//SDIAGAS1 JOB  (SDIAGAS),MSGCLASS=X,MSGLEVEL=(1,1),CLASS=P
//GUVEN    EXEC PLIXCL
//SYSPRINT  DD   SYSOUT=*
//PLI.SYSIN DD   *
 ATOS:PROC(PARM)  OPTIONS(MAIN NOEXECOPS)                              ;
 /*****************************************************************/
 /* Program  : Lmodpli                                            */
 /* Function : This PL/I program formats the output of the Amblist   */
 /*            utility and builds the result file.                */
 /*                                                               */
 /*****************************************************************/
 DCL IN  FILE RECORD SEQL INPUT                                      ,
     OUT FILE RECORD SEQL OUTPUT                                     ,
     (SUBSTR,INDEX,ONCODE) BUILTIN                                   ,
     A BIT(1) INIT('1'B)                                             ,
     SAHA CHAR(133)                                                  ,
     (MEMB_NAME,MEMB_DATE_GUN,MEMB_DATE_YIL) CHAR(2Ø) VAR            ,
     PARM CHAR(44) VAR                                               ,
     1 ALAN                                                          ,
       2   FILLER1       CHAR(1) INIT('')                            ,
       2   MEMB_NAME     CHAR(8)                                     ,
       2   FILLER2       CHAR(1) INIT('')                            ,
       2   MEMB_DATE_YIL PIC'(2)X'                                   ,
```

```
      2   MEMB_DATE_GUN PIC'(3)X'                                    ,
      2   FILLER3       CHAR(2) INIT('')                             ,
      2   TARIH         CHAR(18) INIT('')                            ,
      2   DSN           CHAR(44)                                     ;

ON ENDFILE(IN) A='Ø'B                                                ;

/*******************************************************************/
/* Open the files.                                                 */
/*******************************************************************/

OPEN FILE(IN)                                                        ;
OPEN FILE(OUT)                                                       ;
READ FILE(IN) INTO(SAHA)                                             ;

DO WHILE(A='1'B)                                                     ;
    IF (INDEX(SAHA,'MEMBER NAME') ¬=Ø) | (INDEX(SAHA,'OF YEAR') ¬=Ø)
       THEN
        DO                                                           ;
           ALAN.MEMB_NAME=SUBSTR(SAHA,3Ø,8)                          ;
           READ FILE(IN) INTO(SAHA)                                  ;
           ALAN.MEMB_DATE_GUN=SUBSTR(SAHA,1Ø2,3)                     ;
           ALAN.MEMB_DATE_YIL=SUBSTR(SAHA,114,2)                     ;
           ALAN.DSN=PARM                                             ;
           CALL ALT(ALAN.MEMB_DATE_YIL,ALAN.MEMB_DATE_GUN,ALAN.TARIH);
           WRITE FILE(OUT) FROM(ALAN)                                ;
        END                                                          ;
    READ FILE(IN) INTO(SAHA)                                         ;
END                                                                  ;

/*******************************************************************/
/* Close the files.                                                */
/*******************************************************************/

CLOSE FILE(IN)                                                       ;
CLOSE FILE(OUT)                                                      ;

ALT:PROC(M_DATE_YIL,M_DATE_GUN,YAZI)                                 ;
/*******************************************************************/
/*  This sub_program converts the Julian date to the normal date.  */
/*  eg Julian date 98Ø67 will be converted to 'Ø8.MARCH.1998'      */
/*                                                                 */
/*  NOTE: This conversion does not apply the rules for the leap    */
/*        years such as  1ØØ year and 4ØØ year rule. It just       */
/*        applies 'divisable by 4' rule. The oldest MVS load       */
/*        module is NO MORE THAN 2Ø YEARS. SO THERE IS NO          */
/*        PROBLEM AT ALL in terms of the leap year.                */
/*******************************************************************/
DCL JDATE PIC'99999'                                                 ,
  M_DATE_YIL PIC'XX'                                                 ,
```

```
   M_DATE_GUN PIC'XXX'                                                  ,
   CONV CHAR(5)                                                         ,
   YAZI CHAR(18)                                                        ,
   YEAR PIC'9999'                                                       ,
   DAY PIC'99'                                                          ,
   MONTH CHAR(9)                                                        ,
   (SUM1,I,SUM2) FIXED BIN(31)                                          ,
   YIL(12) FIXED BIN(31) INIT(31,28,31,30,31,30,31,31,30,31,30,31)      ,
   AY(12) CHAR(9) VAR INIT('JANUARY','FEBRUARY','MARCH','APRIL','MAY'   ,
   'JUNE','JULY','AUGUST','SEPTEMBER','OCTOBER','NOVEMBER','DECEMBER'),
   MOD BUILTIN                                                          ;

 CONV = M_DATE_YIL || M_DATE_GUN                                        ;
 SUM1,SUM2 = 0                                                          ;
 JDATE = CONV                                                           ;
 YEAR = JDATE/1000                                                      ;
 JDATE = JDATE - 1000 * YEAR /*Number of days portion of Julian date*/;
 YEAR = 1900 + YEAR /* Add 1900 to the year portion of Julian date */ ;
 /*****************************************************************/
 /* Every year divisible by 4 is a leap year. (MOD builtin function)*/
 /* If it is a leap year then change the array to reflect leap year.*/
 /*****************************************************************/
 IF MOD(YEAR,4) =0 THEN YIL(2) = 29                                     ;

 DO I=1 TO 12                                                           ;
      SUM1 = SUM1 + YIL(I)                                              ;
      IF SUM1     >= JDATE THEN LEAVE                                   ;
      SUM2         = SUM1                                               ;
 END                                                                    ;

 MONTH = AY(I)                                                          ;

 IF SUM1 = JDATE THEN DAY = YIL(I)                                      ;
                 ELSE DAY = JDATE - SUM2                                ;
 /*****************************************************************/
 /* At last, the normal format date is retured to the main program. */
 /*****************************************************************/
 SUBSTR(YAZI,1,2)  = DAY                                                ;
 SUBSTR(YAZI,3,1)  = '.'                                                ;
 SUBSTR(YAZI,4,9)  = MONTH                                              ;
 SUBSTR(YAZI,13,1) = '.'                                                ;
 SUBSTR(YAZI,14,4) = YEAR                                               ;
 END ALT                                                                ;
 END ATOS                                                               ;
//LKED.SYSLMOD  DD DISP=SHR,DSN=SDIAGAS.USER.LOAD(LMODPLI)
```

Atalay Gul
Systems Programmer
Central Bank of Turkey

A column manipulation utility

INTRODUCTION

The following edit macro can perform two different functions:

- It can move or copy data from one range of columns to another (default is move). Excluded lines are omitted.
- It can shift data left or right a specified number of columns. This is the same as "((" OR "))" line commands. Excluded lines are omitted.

The command syntax is shown below. The move format is as follows:

```
COLUTIL begcol endcol tgtcol (MOVE) (.label1) (.label2)
```

The copy format:

```
COLUTIL begcol endcol tgtcol COPY  (.label1) (.label2)
```

The shift format:

```
COLUTIL <LEFT | RIGHT> amount (.label1) (.label2)
```

Remember:

- If you are using the line range labels for a move operation – MOVE must be specified as the fourth positional parameter.
- Also, COPY, MOVE, LEFT and RIGHT can be abbreviated using one or more of their characters.
- Excluded lines are always omitted.

Move examples:

```
COLUTIL 10 20 30
COLUTIL 45 55 10 M
COLUTIL 45 55 10 MOVE
COLUTIL 45 55 10 MOVE .A .B
```

Copy examples:

```
COLUTIL 10 15 20 C
COLUTIL 10 15 20 COPY
COLUTIL 45 50 15 COPY .A .B
```

Shift examples:

```
COLUTIL R 4
COLUTIL L 6
COLUTIL RIGHT 1Ø
COLUTIL RIGHT 1Ø .A .B
COLUTIL LEFT 12
COLUTIL LEFT 25  .A .B
```

ISREDIT

```
/* REXX                                                          */
/* TRACE ?R                                                      */
Address ISREDIT
"MACRO (begcol endcol tgtcol type label1 label2)"
/* Address ISPEXEC "CONTROL ERRORS RETURN"                       */
/*****************************************************************/
/*   VERIFY INPUT PARAMETERS                                     */
/*****************************************************************/
begcol = Translate(begcol)    /* change to upper case if alpha */
"(width) = DATA_WIDTH "        /* length of line               */
width = Format(width)          /* remove leading zeros         */
shift = 'NO'                   /* shift flag                   */
If begcol = '' then do
  zedsmsg = 'MISSING PARAMETER'
  zedlmsg = 'A SHIFT TYPE OR BEGINNING COLUMN NUMBER',
            'MUST BE SPECIFIED.'
  Address ISPEXEC "SETMSG MSG(ISRZØØ1)"  /* msg - with alarm   */
  Exit 12
End
Select
  When Datatype(begcol,Number) = 1 & endcol = '' then do
    zedsmsg = 'NO ENDING COLUMN'
    zedlmsg = 'AN ENDING COLUMN FOR THE',
              'OPERATION MUST BE SPECIFIED.'
    Address ISPEXEC "SETMSG MSG(ISRZØØ1)"  /* msg - with alarm */
    Exit 12
  End /* when */
  When Datatype(begcol,Number) =1 & Datatype(endcol,Number) <>1 then do
    zedsmsg = 'END COLUMN NOT NUMERIC'
    zedlmsg = 'THE ENDING COLUMN FOR THE',
              'OPERATION MUST BE NUMERIC.'
    Address ISPEXEC "SETMSG MSG(ISRZØØ1)"  /* msg - with alarm */
    Exit 12
  End /* when */
  When Datatype(begcol,Number) =1 & Datatype(endcol,Number) =1 then do
    If endcol < begcol then do
      zedsmsg = 'END COL < START COL'
      zedlmsg = 'THE ENDING COLUMN MUST BE GREATER THAN OR',
                'EQUAL TO THE STARTING COLUMN.'
      Address ISPEXEC "SETMSG MSG(ISRZØØ1)" /* msg - with alarm */
      Exit 12
```

```
  End
  If tgtcol <> '' then do
    If Datatype(tgtcol,Number) <> 1 then do
      zedsmsg = 'TARGET COL NOT NUMERIC'
      zedlmsg = 'THE TARGET COLUMN FOR THE',
                'OPERATION MUST BE NUMERIC.'
      Address ISPEXEC "SETMSG MSG(ISRZØØ1)"  /* msg - with alarm */
      Exit 12
    End
  End
  If tgtcol = '' then do
    zedsmsg = 'NO TARGET COLUMN'
    zedlmsg = 'YOU MUST SPECIFY A TARGET COLUMN',
              'FOR THE OPERATION.'
    Address ISPEXEC "SETMSG MSG(ISRZØØ1)"    /* msg - with alarm */
    Exit 12
  End
  If type = '' then type = 'MOVE'
  else do
    type = Translate(type)              /* change to upper case */
    If Abbrev('MOVE',type,1) = Ø & ,
       Abbrev('COPY',type,1) = Ø then do
      zedsmsg = 'INVALID OPERATION'
      zedlmsg = 'OPERATION MUST BE "MOVE" OR "COPY".'
      Address ISPEXEC "SETMSG MSG(ISRZØØ1)"  /* msg - with alarm */
      Exit 12
    End
  End /* else do */
  If begcol < 1 | endcol < 1 | tgtcol < 1 then do
    zedsmsg = 'INVALID COLUMN NUMBER'
    zedlmsg = 'ALL COLUMN SPECIFICATIONS MUST BE' ,
              'BETWEEN 1 AND' width
    Address ISPEXEC "SETMSG MSG(ISRZØØ1)"    /* msg - with alarm */
    Exit 12
  End
  If begcol > width | endcol > width | tgtcol > width then do
    zedsmsg = 'INVALID COLUMN NUMBER'
    zedlmsg = 'ALL COLUMN SPECIFICATIONS MUST BE' ,
              'BETWEEN 1 AND' width
    Address ISPEXEC "SETMSG MSG(ISRZØØ1)"    /* msg - with alarm */
    Exit 12
  End
  If begcol =  tgtcol then do
    zedsmsg = 'NO ACTION TAKEN'
    zedlmsg = 'THE STARTING COLUMN AND TARGET COLUMN',
              'CAN NOT BE THE SAME.'
    Address ISPEXEC "SETMSG MSG(ISRZØØ1)"    /* msg - with alarm */
    Exit 12
  End
End                                          /* when            */
Otherwise
    shift = 'YES'
  If Abbrev('LEFT',begcol,1) = Ø & ,
```

```
       Abbrev('RIGHT',begcol,1) = Ø then do
      zedsmsg = 'INVALID SHIFT DIRECTION'
      zedlmsg = 'A SHIFT DIRECTION OF "LEFT" OR "RIGHT"',
                'MUST BE SPECIFIED'
      Address ISPEXEC "SETMSG MSG(ISRZØØ1)"   /* msg - with alarm */
      Exit 12
    End
    If endcol = '' then do
      zedsmsg = 'NO SHIFT AMOUNT'
      zedlmsg = 'A SHIFT AMOUNT FOR THE',
                'OPERATION MUST BE SPECIFIED.'
      Address ISPEXEC "SETMSG MSG(ISRZØØ1)"   /* msg - with alarm */
      Exit 12
    End
    Else do
      If Datatype(endcol,Number) <> 1 then do
        zedsmsg = 'SHIFT AMOUNT NOT NUMERIC'
        zedlmsg = 'THE SHIFT AMOUNT SPECIFIED FOR THE',
                  'OPERATION MUST BE NUMERIC.'
        Address ISPEXEC "SETMSG MSG(ISRZØØ1)"  /* msg - with alarm */
        Exit 12
      End
      If endcol > width - 1  | endcol < 1 then do
        zedsmsg = 'INVALID SHIFT AMOUNT'
        zedlmsg = 'THE SHIFT AMOUNT SPECIFIED MUST',
                  'BE BETWEEN 1 AND' width - 1 || '.'
        Address ISPEXEC "SETMSG MSG(ISRZØØ1)"  /* msg - with alarm */
        Exit 12
      End
    End  /* else do */
End /* select */
/*********************************************************************/
/*   SHIFT PROCESSING SET-UP                                         */
/*********************************************************************/
If shift = 'YES'  then do
  label1   = tgtcol
  label2   = type
  shiftamt = endcol
  type     = 'MOVE'        /* shift is really a MOVE operation    */
  If Abbrev('LEFT',begcol,1) <> Ø then do           /* left shift */
    shifttyp = 'LEFT'
    begcol   = shiftamt + 1
    endcol   = width
    tgtcol   = 1
  End
  Else do                                          /* right shift */
    shifttyp = 'RIGHT'
    begcol   = 1
    endcol   = width
    tgtcol   = shiftamt + 1
  End
End                                          /* if shift = 'YES'  */
/*********************************************************************/
/*   FIND OUT IF LABELS ARE BEING USED                               */
/*********************************************************************/
```

```
Call FIND_LABELS
/*****************************************************************/
/* INITIALIZE VARIABLES NEEDED IN PROCESSING LOOP                */
/*****************************************************************/
count    = Ø                          /* count of changed lines  */
tgtlen   =  endcol-begcol+1           /* length of operation     */
/*****************************************************************/
/*   BEGIN COLUMN MANIPULATION LOOP                              */
/*****************************************************************/
Do until lastln = firstln-1
  /* copy the data in the current line to variable 'data1'       */
  "(data1) = LINE "firstln
  "ISREDIT (chkexcl) = XSTATUS" firstln
  If chkexcl = "NX" then do
    count = count + 1
    tgtdata = Substr(data1,begcol,tgtlen)
    If shift = 'YES' & shifttyp = 'LEFT' then , /* clr data for left */
       data1 = Overlay(' ',data1,width-shiftamt,shiftamt+1) /* shift */
    else
    If Abbrev('MOVE',type,1) <> Ø then ,         /* clear data for  */
       data1 = Overlay(' ',data1,begcol,tgtlen)  /* column MOVE     */
    data1 = Overlay(tgtdata,data1,tgtcol,tgtlen) /* COPY - no clear */
  End
  /* copy the modified line back into the current line    */
  "LINE" firstln "= (data1)"
  firstln = firstln + 1
End   /* do until   */
/*****************************************************************/
/*   END COLUMN MANIPULATION LOOP                                */
/*****************************************************************/
If shift <> 'YES' then do
   If Abbrev('MOVE',type,1) <> Ø then msgtype = 'MOVED'
    else msgtype = 'COPIED'
   If tgtlen+tgtcol-1 ¬> width then do       /* no truncation    */
     zedsmsg = count 'LINES CHANGED'
     zedlmsg = 'COLUMNS' begcol 'THROUGH' endcol 'ON' count ,
               'LINES WERE' msgtype 'TO COLUMN' tgtcol || '.'
     Address ISPEXEC "SETMSG MSG(ISRZØØØ)"     /* msg - no alarm */
     Exit Ø
   End
   Else do
     zedsmsg = count 'LINES TRUNCATED'
     zedlmsg = 'COLUMNS' begcol 'THROUGH' endcol 'ON' count ,
               'LINES WERE' msgtype 'TO COLUMN' tgtcol 'AND TRUNCATED.'
     Address ISPEXEC "SETMSG MSG(ISRZØØ1)"   /* msg - with alarm */
     Exit 4
   End
End
Else do                              /* total messages for shift */
  zedsmsg = count 'LINES SHIFTED'
  zedlmsg = count 'LINES WERE SHIFTED' shiftamt 'COLUMNS' ,
            'TO THE' shifttyp || '.'
  Address ISPEXEC "SETMSG MSG(ISRZØØØ)"       /* msg - no alarm */
```

```
  Exit Ø
End
/*********************************************************************/
/*  SUB-ROUTINE TO FIND LABELS                                       */
/*********************************************************************/
FIND_LABELS:
If label1 = '' then do
  firstln  = 1
  "(lastln) = LINENUM .ZLAST"
End
Else do
  If label2 = '' then label2 = label1
  firstsv = 'NOTFOUND'
  lastsv  = 'NOTFOUND'
  label1 = Translate(label1)
  label2 = Translate(label2)
  "(saveln) = DISPLAY_LINES"
  "UP MAX"
  Do forever
    "LOCATE LAB NEXT"
    if rc <> Ø then leave
    "(labline,junk) = DISPLAY_LINES"
    "(lab,junk) = LABEL" labline
    if lab = label1 then firstsv = labline
    if lab = label2 then lastsv  = labline
  End
  /* return display lines to originial position  */
  "UP MAX"
  If saveln <> 1 then "DOWN " saveln  /* do not scroll if at top */
  If firstsv = 'NOTFOUND'  then do
    zedsmsg = 'RANGE LABEL ERROR'
    zedlmsg = 'THE SPECIFIED RANGE LABEL "' || label1 '" WAS',
              'NOT FOUND'
    Address ISPEXEC "SETMSG MSG(ISRZØØ1)"   /* msg - with alarm */
    Exit 12
  End
  If lastsv  = 'NOTFOUND'  then do
    zedsmsg = 'RANGE LABEL ERROR'
    zedlmsg = 'THE SPECIFIED RANGE LABEL "' || label2 '" WAS',
             'NOT FOUND'
    Address ISPEXEC "SETMSG MSG(ISRZØØ1)"   /* msg - with alarm */
    Exit 12
  End
  If firstsv > lastsv then do
    firstln = lastsv
    lastln  = firstsv
  End
  Else do
    firstln = firstsv
    lastln  = lastsv
  End
End
Return
```

Assembler instruction trace – part 3

This month we continue our look at the code for the Assembler instruction trace.

```
            WHEN  CLC,=F'4',EQ,OLDREGS
               MVC   PRTLINE+69(6),=CL6' SET'
            WHEN  CLC,=F'8',EQ,OLDREGS
               MVC   PRTLINE+69(6),=CL6' RESET'
            WHEN  CLC,=F'12',EQ,OLDREGS
               MVC   PRTLINE+69(6),=CL6' TEST'
            ENDSEL
         WHEN  C,R15,EQ,=A(X'1D')
            MVC   PRTLINE+63(15),=CL15'-VSAM CB UPDATE'
         WHEN  C,R15,EQ,=A(X'1E')
            MVC   PRTLINE+63(Ø8),=CLØ8'-MSGDISP'
         WHEN  C,R15,EQ,=A(X'1F')
            MVC   PRTLINE+63(Ø8),=CLØ8'-SYNCDEV'
         WHEN  C,R15,EQ,=A(X'2Ø')
            MVC   PRTLINE+63(11),=CL11'-NOTE/POINT'
         WHEN  C,R15,EQ,=A(X'21')
            MVC   PRTLINE+63(14),=CL14'-OUTDEL/OUTADD'
         WHEN  C,R15,EQ,=A(X'22')
            MVC   PRTLINE+63(14),=CL14'-MVC BDT'
         WHEN  C,R15,EQ,=A(X'26')
            MVC   PRTLINE+63(14),=CL14'-DFSORT'
         WHEN  C,R15,EQ,=A(X'27'),OR,C,R15,EQ,=A(X'2C')
            MVC   PRTLINE+63(14),=CL14'-DFSMS(AOM)'
         ENDSEL
         MODEXIT
         EJECT
INFO_SVC116    MODENTRY                                          +
               INITBASE=R1Ø,BAKR=YES
         L     R15,OLDREGS+(15*4)
         SELECT
         WHEN  LTR,R15,R15,Z
            MVC   PRTLINE+63(Ø9),=CLØ9'-IECTRDTI'
         WHEN  C,R15,EQ,=F'1'
            MVC   PRTLINE+63(Ø9),=CLØ9'-IECTATNI'
         WHEN  C,R15,EQ,=F'2'
            MVC   PRTLINE+63(Ø9),=CLØ9'-CHNGNTRY'
         WHEN  C,R15,EQ,=F'3'
            MVC   PRTLINE+63(Ø9),=CLØ9'-IECTCHGA'
         WHEN  C,R15,EQ,=F'4'
            MVC   PRTLINE+63(Ø9),=CLØ9'-RESETPL'
         WHEN  C,R15,EQ,=F'8'
            MVC   PRTLINE+63(Ø9),=CLØ9'-CALLDISP'
         ENDSEL
         MODEXIT
```

```
         EJECT
INFO_SVC120    MODENTRY                                                +
               INITBASE=R1Ø,BAKR=YES
         IF    TM,OLDREGS+15*4+3,1,O
            MVC   PRTLINE+53(16),=CL16'FREEMAIN'
         ELSE
            MVC   PRTLINE+53(16),=CL16'GETMAIN'
         ENDIF
         MVC   PRTLINE+65(3),=C'SP='
         XR    R1,R1
         IC    R1,OLDREGS+4*15+2
         CVD   R1,DUB
         OI    DUB+7,X'ØF'
         UNPK  PRTLINE+68(3),DUB+6(2)
         IF    CLC,=F'Ø',NE,OLDREGS
            MVC   PRTLINE+75(14),=C'LV=X'''''
            UNPK  PRTLINE+8Ø(9),OLDREGS(5)
            MVI   PRTLINE+89,C''''
            TR    PRTLINE+8Ø(8),HEXCHAR-C'Ø'
            MVC   PRTLINE+95(2),=C'A='
            UNPK  PRTLINE+97(9),OLDREGS+4(5)
            MVI   PRTLINE+1Ø6,X'4Ø'
            TR    PRTLINE+97(8),HEXCHAR-C'Ø'
         ENDIF
         MVC   PRTLINE+11Ø(4),=C'LOC='
         SELECT
         WHEN  TM,OLDREGS+15*4+3,X'3Ø',O    .BOTH BITS - ANY
            MVC   PRTLINE+114(3),=C'ANY'
         WHEN  CC=8                         .NO BITS - RES
            MVC   PRTLINE+114(3),=C'RES'
         WHEN  NONE
            MVC   PRTLINE+114(5),=C'BELOW'
         ENDSEL
         IF    TM,OLDREGS+4*15+3,4,O
            MVC   PRTLINE+12Ø(4),=C'PAGE'
         ENDIF
         IF    TM,OLDREGS+4*15+3,2,Z
            MVC   PRTLINE+125(4),=C'COND'
         ENDIF
         MODEXIT
         EJECT
INFO_SVC122    MODENTRY                                                +
               INITBASE=R1Ø,BAKR=YES
         L     R15,OLDREGS+(15*4)
         SELECT
         WHEN  C,R15,EQ,=F'5'
            MVC   PRTLINE+63(Ø9),=CLØ9'-EVENTS'
         WHEN  C,R15,EQ,=F'7'
            MVC   PRTLINE+63(14),=CL14'-EXTENDED LINK'
         WHEN  C,R15,EQ,=F'8'
            MVC   PRTLINE+63(14),=CL14'-EXTENDED XCTL'
         WHEN  C,R15,EQ,=F'9'
```

```
            MVC   PRTLINE+63(14),=CL14'-EXTENDED LOAD'
         ENDSEL
         MODEXIT
         EJECT
INFO_SVC137    MODENTRY                                                +
               INITBASE=R1Ø,BAKR=YES
         L     R15,OLDREGS+(15*4)
         SELECT
         WHEN  LTR,R15,R15,Z
            MVC   PRTLINE+63(Ø9),=CLØ9'-CALLDISP'
         ENDSEL
         MODEXIT
         EJECT
         LTORG
         EJECT
EXEC_PR  MODENTRY NEWBASE=R1Ø
         MVC   XCELL(2),Ø(R9)
         MVC   FLAGS,=AL2(RRBIT+LMSTMBIT)
         LA    R15,PR_STACK
         IF    C,R15,LE,CUR_PR
            MVC   OLDREGS(16*4),REGTBL
            STM   RØ,R14,TEMPREGS
            LR    R15,R7
         ELSE
            PERF  WRITE
            MVC   PRTLINE(56),=C'***** TRACE TERMINATED BECAUSE OF ''PR+
               '' INSTRUCTION *****'
            PERF  WRITE
            PERF  WRITE
            MVC   PRTLINE(26),=C'REGS BEFORE PR INSTRUCTION'
            PERF  WRITE
            PERF  DUMPREGS
            MVC   PRTLINE(33),=C'ACCESS REGS BEFORE PR INSTRUCTION'
            PERF  WRITE
            PERF  DUMP_ARS
            MVC   PRTLINE(13),=C'RETURN PSW = '
            LA    R15,1
            ESTA  R2,R15
            STM   R2,R3,DUB
            UNPK  PRTLINE+13(9),DUB(5)
            MVI   PRTLINE+13+8,X'4Ø'
            TR    PRTLINE+13(8),HEXCHAR-C'Ø'
            UNPK  PRTLINE+22(9),DUB+4(5)
            MVI   PRTLINE+22+8,X'4Ø'
            TR    PRTLINE+22(8),HEXCHAR-C'Ø'
            PERF  WRITE
            ST    R9,REGTBL+14*4
            B     BREAK_LOOP
         ENDIF
         PR
* IF THE PR RETURNS TO HERE, THEN:
* R15 WILL CONTAIN A(MYSAVE).
```

```
* TEMPREGS WILL CONTAIN TRACE'S OTHER REGS
PR_RET   DS    ØH
         STM   R2,14,REGTBL+2*4-MYSAVE(R15)    .PR DOES NOT RESTORE...
         STAM  R2,R14,AR_SAVE+2*4-MYSAVE(15)   .... R15-R1 OR AR15-AR1
         LM    RØ,R14,TEMPREGS-MYSAVE(R15)     .RESTORE TRACE'S REGS
         LAM   RØ,R15,=16F'Ø'
         PERF  SHOWINST
         MVC   OPCODE,=CL5'PR'
         UNPK  GR_1(9),REGTBL+15*4(5)
         ED    GR_1(8),HEXCHAR-C'Ø'
         MVI   GR_1+8,X'4Ø'
         MVC   GR_1-4(4),=C'R15='
         L     R15,CUR_PR
         MVC   NEW_IPTR,Ø(R15)
         L     R9,Ø(,R15)
         S     R15,=A(L'PR_STACK)
         ST    R15,CUR_PR
         MODEXIT
         EJECT
         LTORG
         EJECT
EXEC_RR  MODENTRY  NEWBASE=R1Ø
         MVI   CODEFLD+1,X'24'          .I AM GOING TO USE R2 & R4
         IC    R1,XCELL+1               .RR
         N     R1,=XL4'FØ'              .RØ
         SRL   R1,2                     .RØ*4
         LA    R15,AR_SAVE(R1)
         LA    R1,REGTBL(R1)
         LM    R2,R3,Ø(R1)              .PRIME R2, AND R3 FOR DBL REGS
         IF    IAC,R14,NZ
            XR    R14,R14
            IC    R14,XCELL
            A     R14,=A(AR_ØØ)
            IF    TM,Ø(R14),AR_R1,O
               IF    TM,XCELL+1,X'FØ',Z
                  LAM   R2,R2,=F'Ø'
               ELSE
                  LAM   R2,R2,Ø(R15)
               ENDIF
            ENDIF
         ENDIF
         IC    R1,XCELL+1               .RR
         N     R1,=F'15'                .ØR
         SLL   R1,2                     .*4
         LA    R15,AR_SAVE(R1)
         LA    R1,REGTBL(R1)
         LM    R4,R5,Ø(R1)              .PRIME R4, AND R5 FOR DBL REGS
         IF    IAC,R14,NZ
            XR    R14,R14
            IC    R14,XCELL
            A     R14,=A(AR_ØØ)
            IF    TM,Ø(R14),AR_R2,O
```

```
      IF    TM,XCELL+1,X'ØF',Z
         LAM   R4,R4,=F'Ø'
      ELSE
         LAM   R4,R4,Ø(R15)
      ENDIF
   ENDIF
ENDIF
RUN_INST
STM   R4,R5,Ø(R1)             .SAVE REGS, IN CASE THEY CHANGED
LA    R9,2(,R9)               .INCREMENT INSTRUCTION PTR.
IC    R1,XCELL+1              .RR
N     R1,=XL4'FØ'             .RØ
SRL   R1,2                    .R*4
LA    R1,REGTBL(R1)
STM   R2,R3,Ø(R1)             .SAVE R1 PAIR
PERF  SHOWINST
IC    R3,XCELL+1
PERF  REG_OPS
PERF  SHOW_GRS
IF    IAC,R14,NZ
   XR    R15,R15
   IC    R15,XCELL
   A     R15,=A(AR_ØØ)
   IF    TM,Ø(R15),B'ØØ111111',NZ
      MVC   AR_LINE,PRTLINE
      MVC   PRTLINE,=CL133' '
      SHOW_AR FROM=(XCELL+1),TO=(GR_1-5)
      IC    R1,XCELL+1
      SLL   R1,4
      STC   R1,DUB
      SHOW_AR FROM=DUB,TO=(GR_2-5)
      MVC   I_PTR(35),=C'RELATED ACCESS REGS FOR ABOVE INSTR'
   ENDIF
ENDIF
IF    CLI,XCELL,EQ,X'ØE',OR,CLI,XCELL,EQ,X'ØF'
   IF    CLC,AR_LINE,NE,=CL133' '
      XC    AR_LINE,PRTLINE
      XC    PRTLINE,AR_LINE
      XC    AR_LINE,PRTLINE
      PERF  WRITE                .WRITE TRACE LINE
      MVC   PRTLINE,AR_LINE
      MVC   AR_LINE,=CL133' '
   ENDIF
   PERF  WRITE
   XR    R1,R1
   IC    R1,XCELL+1
   SRL   R1,4
   PERF  DISPLAY_LONG
   PERF  WRITE
   IC    R1,XCELL+1
   N     R1,=F'15'
   PERF  DISPLAY_LONG
```

```
            MVC   I_PTR(3),=C'OP2'
         ENDIF
         MODEXIT
         EJECT
DISPLAY_LONG   MODENTRY INITBASE=R1Ø,BAKR=YES
         LR    R14,R1
         SLL   R14,2
         IF    LTR,R1,R1,NZ,AND,IAC,R2,2
            LA    R2,AR_SAVE(R14)
            LAM   R1,R1,Ø(R2)
         ENDIF
         LA    R15,OLDREGS(R14)
         MVC   I_PTR(39),=C'OP1 ADDR=XXXXXXXX LEN=X''XXXXXXXX'' DATA='
         UNPK  I_PTR+9(9),Ø(5,R15)
         MVI   I_PTR+9+8,X'4Ø'
         TR    I_PTR+9(8),HEXCHAR-C'Ø'
         UNPK  I_PTR+24(9),4(5,R15)
         MVI   I_PTR+24+8,C''''
         TR    I_PTR+24(8),HEXCHAR-C'Ø'
         L     R1,OLDREGS(R14)
         IF    ICM,R5,15,4(R15),NZ
            IF    C,R5,GT,=F'88'
               LA    R5,88
            ENDIF
            LA    R2,I_PTR+39
            LR    R3,R5
            DO    WHILE=(C,R3,GT,=F'7')
               UNPK  Ø(15,R2),Ø(8,R1)
               LA    R2,14(,R2)
               LA    R1,7(,R1)
               S     R3,=F'7'
            ENDDO
            LR    R14,R3
            BCTR  R14,Ø
            EX    R14,MOVE_OP
            LR    R14,R3
            SLL   R14,1+4           .*2, AND SHIFT TO NEXT NIBBLE
            LA    R15,Ø(R3,R14)
            EX    R15,UNPK_OP
            SLL   R3,1
            LA    R14,Ø(R3,R2)
            MVI   Ø(R14),X'4Ø'
            LA    R6,I_PTR+39-11
            SLL   R5,1                    .* 2
            BCTR  R5,Ø                    .MAKE EXEC LEN
            EX    R5,TRANS
            CPYA   R1,R12
         ENDIF
         MODEXIT
         EJECT
EXEC_RX  MODENTRY  NEWBASE=R1Ø,LIST=YES
         XR    R1,R1
```

```
ICM   R1,B'ØØ11',CODEFLD+2
LR    R15,R1
IF    N,R15,=A(X'FØØØ'),NZ
   N     R1,=F'4Ø95'
   O     R1,=A(X'6ØØØ')
   STCM  R1,B'ØØ11',CODEFLD+2
   SRL   R15,12-2
   L     R6,AR_SAVE(R15)
   SAR   R6,R6
   L     R6,REGTBL(R15)
ENDIF
IF    TM,XCELL+2,X'FØ',Z,OR,IAC,R14,Z
   LAM   R6,R6,=F'Ø'
ELSE
   XR    R14,R14
   IC    R14,XCELL
   A     R14,=A(AR_ØØ)
   IF    TM,Ø(R14),AR_B2,Z
      LAM   R6,R6,=F'Ø'
   ENDIF
ENDIF
IC    R14,XCELL+1
N     R14,=A(X'FØ')
SRL   R14,2             .MOVE REG # TO LOW-ORDER & MULT BY 4
L     R2,REGTBL(R14)
L     R3,REGTBL+4(R14)
LA    R1,X'2Ø'
IF    TM,XCELL+1,15,NZ
   O     R1,=F'4'
   IC    R15,XCELL+1
   N     R15,=F'15'
   SLL   R15,2
   L     R4,REGTBL(R15)
   L     R5,REGTBL+4(R15)
ENDIF
STC   R1,CODEFLD+1
XR    RØ,RØ
RUN_INST
IC    R1,XCELL+1
N     R1,=A(X'FØ')
SRL   R1,2
ST    R2,REGTBL(R1)
ST    R3,REGTBL+4(R1)
IF    TM,XCELL+1,X'FØ',NZ,AND,IAC,R14,NZ
   XR    R14,R14
   IC    R14,XCELL
   A     R14,=A(AR_ØØ)
   IF    TM,Ø(R14),AR_R1+AR_UR1,NZ
      LA    R1,AR_SAVE(R1)
      STAM  R2,R2,Ø(R1)
   ENDIF
ENDIF
```

```
         CPYA  R6,R12
         PERF  SHOWINST
         IC    R3,XCELL+1
         PERF  REG_OPS
         LA    R3,XCELL+2
         PERF  SHOW_BD
         IC    R3,XCELL+1
         PERF  SHOW_GRS
         SELECT
         WHEN  CLI,XCELL,EQ,X'4E',OR,CLI,XCELL,EQ,X'4F' CVD/CVB
            LA    R5,8
         WHEN  CLI,XCELL,EQ,X'5C',OR,CLI,XCELL,EQ,X'5D' M/D
            LA    R5,4
         WHEN  CLI,XCELL,EQ,X'41'      .LA?
            XR    R5,R5               .SET DISPLAY LENGTH = Ø
         WHEN  TM,FLAGS,FULLBIT,O      .FULLWORD INSTRUCTION?
            LA    R5,4                .DISPLAY LENGTH = 4
         WHEN  TM,FLAGS,HALFBIT,O      .HALFWORD ?
            LA    R5,2                .LENGTH = 2
         WHEN  TM,FLAGS,DBLBIT,O       .DOUBLEWORD ?
            LA    R5,8                .LENGTH = 8
         WHEN  NONE                    .ANYTHING ELSE, I DON'T KNOW,
            XR    R5,R5               .SO SET LENGTH = Ø
         ENDSEL
         SHOW_EFA TO=(EFA1-1),FROM=(XCELL+2)
         XR    R15,R15
         IC    R15,XCELL
         A     R15,=A(AR_ØØ)
         ST    R15,AR_FLAG
         IF    TM,Ø(R15),AR_UR1,O,ORIF,                                +
               TM,Ø(R15),B'ØØ111111',NZ,AND,IAC,R14,NZ
            MVC   AR_LINE,PRTLINE
            MVC   PRTLINE,=CL133' '
            MVC   I_PTR(35),=C'RELATED ACCESS REGS FOR ABOVE INSTR'
            L     R15,AR_FLAG
            IF    TM,Ø(R15),AR_B2,O
               SHOW_AR FROM=(XCELL+2),TO=(EFA1-5)
            ENDIF
            L     R15,AR_FLAG
            IF    TM,Ø(R15),AR_R1+AR_UR1,NZ
               SHOW_AR NEW,FROM=(XCELL+1),TO=(GR_1-5)
            ENDIF
         ENDIF
         LA    R9,4(,R9)
         MODEXIT
         EJECT
EXEC_BX  MODENTRY  NEWBASE=R1Ø         .BRXH, BRXLE, BXH OR BXLE
         IC    R2,XCELL+1             .RR
         N     R2,=XL4'FØ'            .RØ
         SRL   R2,2                   .ØR*4
         L     R3,REGTBL(R2)          .GET VALUE
         IC    R1,XCELL+1             .RR
```

```
N     R1,=F'15'               .ØR
SLL   R1,2                    .* 4
L     R4,REGTBL(R1)           .GET VALUE
AR    R3,R4                   .ADD THEM
ST    R3,REGTBL(R2)           .AND SAVE UPDATED VALUE
IF    TM,XCELL+1,1,Z          .2ND REG EVEN?
   C     R3,REGTBL+4(R1)      .THEN COMPARE PAIRED ODD REG
ELSE
   CR    R3,R4                .COMPARE AGAINST GIVEN ODD REG
ENDIF
LA    R8,XCELL+2
IF    CC=2                    .CC R3 > COMPARE VALUE
   SELECT
   WHEN  TM,XCELL,1,O         .BRXLE OR BXLE? ('85' OR '87')
      LA    R9,4(,R9)          .BXLE, GO TO NXT SEQ INSTR
   WHEN  CLI,XCELL,EQ,X'86'   .BXH?
      PERF  EVALBD            .R1 WILL CONTAIN DEST ADDR
      LR    R9,R1             .YES, NDX HI, SO BRANCH
   WHEN  NONE                 .X'84', BRXH
      LH    R1,XCELL+2
      SLA   R1,1
      AR    R9,R1
   ENDSEL
ELSE                          .R3 <= COMPARE VALUE
   SELECT
   WHEN  TM,XCELL,1,Z         .BRXH OR BXH? ('84' OR '86')
      LA    R9,4(,R9)          .BXLE, GO TO NXT SEQ INSTR
   WHEN  CLI,XCELL,EQ,X'87'   .BXLE?
      PERF  EVALBD            .R1 WILL CONTAIN DEST ADDR
      LR    R9,R1             .YES, NDX LE, BRANCH
   WHEN  NONE                 .BRXLE
      LH    R1,XCELL+2
      SLA   R1,1
      AR    R9,R1
   ENDSEL
ENDIF
CPYA  R1,RØ
PERF  SHOWINST
IC    R3,XCELL+1
PERF  REG_OPS
MVI   Ø(R6),C','
   LA    R6,1(,R6)
IF    CLI,XCELL,LT,X'86'      .RELATIVE INST
   MVC   Ø(7,R6),=C'X''1234'''
   UNPK  2(5,R6),XCELL+2
   MVI   7(R6),C''''
   TR    XCELL+2(4),HEXCHAR-C'Ø'
   L     R1,NEW_IPTR
   LH    R15,XCELL+2
   SLA   R15,1
   AR    R1,R15
   ST    R1,DUB
```

```
            UNPK  EFA2(9),DUB(5)
            MVI   EFA2-1,C'('
            MVI   EFA2+8,C')'
            TR    EFA2(8),HEXCHAR-C'Ø'
         ELSE
            PERF  SHOW_GRS
            LA    R6,FIELDS+8
            LA    R3,XCELL+2
            PERF  SHOW_BD
            SHOW_EFA TO=(EFA2-1),FOR=Ø,FROM=(XCELL+2)
         ENDIF
         MODEXIT
         EJECT
EXEC_SI  MODENTRY  NEWBASE=R1Ø
         ICM   R1,3,CODEFLD+2
         LR    RØ,R1
         IF    N,R1,=A(X'ØØØØFØØØ'),NZ
            N     RØ,=A(X'ØØØØØFFF')
            O     RØ,=A(X'ØØØØ6ØØØ')
            STCM  RØ,3,CODEFLD+2
            SRL   R1,12                  .MOVE REG NO TO LOW-NIBBLE
            SLL   R1,2                   .MULT. BY 4
            L     R6,AR_SAVE(R1)
            SAR   R6,R6
            L     R6,REGTBL(R1)
         ENDIF
         RUN_INST
         CPYA  R6,R12
         PERF  SHOWINST
         LA    R6,FIELDS
         LA    R3,XCELL+2
         PERF  SHOW_BD
         IF    CLI,XCELL,NE,X'93'      .TS DOESN'T HAVE IMMEDIATE FLD
            MVC   Ø(6,R6),=C',X''  '''
            XR    R1,R1
            IC    R1,XCELL+1
            SRL   R1,4
            IC    R1,HEXCHAR(R1)
            STC   R1,3(R6)
            IC    R1,XCELL+1
            N     R1,=F'15'
            IC    R1,HEXCHAR(R1)
            STC   R1,4(R6)
         ENDIF
         IF    CLI,XCELL,EQ,X'AF' .X'AF' (MC) DOES NOT ADDRESS STG
            XR    R5,R5
         ELSE
            LA    R5,1
         ENDIF
         SHOW_EFA TO=(SS_EFA1-1),FROM=(XCELL+2)
         XR    R15,R15
         IC    R15,XCELL
```

```
         A     R15,=A(AR_ØØ)
         IF    TM,Ø(R15),AR_B1,O,AND,IAC,R14,NZ
            MVC   AR_LINE,PRTLINE
            MVC   PRTLINE,=CL133' '
            MVC   I_PTR(35),=C'RELATED ACCESS REGS FOR ABOVE INSTR'
            SHOW_AR FROM=(XCELL+2),TO=(GR_1-5)
         ENDIF
         LA    R9,4(,R9)
         MODEXIT
         EJECT
EXEC_BC  MODENTRY  NEWBASE=R1Ø
         XR    R1,R1
         IC    R1,REALCC               .GET CC + PGM MASKS
         SRL   R1,4                    .DROP PGM MASKS
         IC    R1,HEXCC(R1)            .GET TESTABLE CC
         XR    RØ,RØ                   .GET TESTED CC
         IC    RØ,XCELL+1
         SRL   RØ,4
         SELECT
         WHEN  NR,R1,RØ,Z              .NO CORRESPONDING BITS
            LA    R9,2(,R9)            .THEN NO BRANCH TO BE GENNED
            IF    TM,FLAGS+1,RXBIT,O   .GO TO NEXT SEQ INSTR.
               LA    R9,2(,R9)
            ENDIF
         WHEN  TM,FLAGS+1,RXBIT,O      .BC INST
            LA    R8,XCELL+2           .GO TO DEST ADDR
            PERF  EVALBD
            LR    R9,R1
         WHEN  TM,XCELL+1,15,NZ        .BCR, NON-ZERO DEST REG
            IC    R1,XCELL+1
            N     R1,=F'15'
            SLL   R1,2
            L     R9,REGTBL(R1)
         WHEN  NONE                    .BCR, DEST REG = RØ, NO BRANCH
            LA    R9,2(,R9)
         ENDSEL
         PERF  SHOWINST
         LA    R6,FIELDS
         IF    TM,XCELL+1,B'',M   .NOT COND Ø OR COND F
            IC    R1,XCELL+1           .THEN SHOW
            SRL   R1,4                 .WHICH COND WAS
            N     R1,=F'15'            .REQUESTED
            IC    R1,HEXCHAR(R1)
            STC   R1,FIELDS
            MVI   FIELDS+1,C','
            LA    R6,2(,R6)
         ENDIF
         IF    TM,FLAGS+1,RRBIT,O
            IC    R1,XCELL+1
            N     R1,=F'15'
            CVD   R1,DUB
            OI    DUB+7,X'ØF'
```

```
            UNPK  Ø(3,R6),DUB+6(2)
            MVI   Ø(R6),C'R'
            SLL   R1,2
            LA    R1,REGTBL(R1)
            UNPK  GR_1(9),Ø(5,R1)
            MVI   GR_1+8,X'4Ø'
            TR    GR_1(8),HEXCHAR-C'Ø'
         ELSE
            LA    R3,XCELL+2
            PERF  SHOW_BD
            SHOW_EFA TO=(GR_1-1),FROM=(XCELL+2),FOR=Ø
         ENDIF
         MODEXIT
         EJECT
EXEC_LM  MODENTRY  NEWBASE=R1Ø
         LA    R8,XCELL+2
         PERF  EVALBD                  .GET DEST ADDR
         IC    R15,XCELL+1             .GET FIRST REG
         N     R15,=XL4'FØ'
         SRL   R15,4                   .MOVE TO LO NIBBLE, THEN *2
         SLL   R15,2                   .MOVE TO LO NIBBLE, THEN *2
         IC    R14,XCELL+2
         N     R14,=A(X'ØØØØØØFØ')
         SRL   14,4   .MOVE TO LOW-NIBLLE, MULT BY 4
         SLL   14,2   .MOVE TO LOW-NIBLLE, MULT BY 4
         L     R14,AR_SAVE(R14)
         IF    CLI,XCELL,EQ,X'98',OR,  .LM?                              +
               CLI,XCELL,EQ,X'9A'      .LAM?
            LR    R2,R1                .SRC PTR = 2ND OPERAND
            SAR   R2,R14
            IF    CLI,XCELL,EQ,X'98'
               LA    R3,REGTBL(R15)       .DEST PTR = REGTBL
            ELSE
               LA    R3,AR_SAVE(R15)      .SRC PTR = REGTBL
            ENDIF
         ELSE
            SAR   R3,R14
            LR    R3,R1                .DEST PTR = 2ND OPERAND
            IF    CLI,XCELL,EQ,X'9Ø'
               LA    R2,REGTBL(R15)       .SRC PTR = REGTBL
            ELSE
               LA    R2,AR_SAVE(R15)      .SRC PTR = REGTBL
            ENDIF
         ENDIF
         IC    R15,XCELL+1
         LR    R5,R15
         N     R15,=F'15'              .GET SECOND REG NUMBER
         N     R5,=XL4'FØ'             .GET FIRST REG NUMBER
         SRL   R5,4                    .MOVE FIRST REG NO TO LO NIBBLE
         IF    CR,R15,LT,R5            .SECOND REG LT FIRST REG?
            S     R5,=F'16'            .THEN CATER FOR WRAPAROUND
            LPR   R5,R5                .FROM CURRENT REG TO R15
```

```
            SLL   R5,2                .*4
            BCTR  R5,Ø                .EXECUTABLE LEN
            EX    R5,MOV_LM           .MOVE REG VALUES
            IF    CLI,XCELL,EQ,X'98'  .LM?
               LA    R3,REGTBL        .POINT AT RØ ENTRY
               LA    R2,1(R2,R5)      .POINT PAST REGS ALREADY MOVED
            ELSE
               LA    R2,REGTBL
               LA    R3,1(R3,R5)
            ENDIF
            XR    R5,R5
         ENDIF
         SR    R15,R5                .DIFFERENCE BETWEEN 1ST REG&2ND
         LA    R15,1(,R15)           .BUT COUNT OFFSET
         SLL   R15,2                 .* 4
         BCTR  R15,Ø                 .EXECUTABL LEN
         EX    R15,MOV_LM
         PERF  SHOWINST
         IC    R3,XCELL+1
         PERF  REG_OPS
         MVI   FIELDS+7,C','
         LA    R6,FIELDS+8
         LA    R3,XCELL+2
         PERF  SHOW_BD
         SHOW_EFA TO=(GR_1-1),FROM=(XCELL+2),FOR=Ø
         SHOW_AR  TO=(DR1+2),FROM=(XCELL+2)
         LA    R9,4(,R9)
         MODEXIT
MOV_LM   MVC   Ø(Ø,R3),Ø(R2)
         EJECT
EXEC_B2  MODENTRY  NEWBASE=R1Ø       .'B2' EXTENDED OPCODES
         CLI   XCELL+1,X'79'         .DEFINED UP TO B279 ONLY (SACF)
         BH    ILGLOP
         XR    R1,R1
         IC    R1,XCELL+1            .GET OPCODE'S 2ND BYTE
         SLL   R1,1                  .* 2
         L     R15,=A(B2FLAGS)
         LH    RØ,Ø(R1,R15)          .GET B2-FLAGS
         STH   RØ,FLAGS
         TM    FLAGS,ILGLBIT
         BO    ILGLOP
         SELECT
         WHEN  CLI,XCELL+1,EQ,X'4Ø'  .BAKR?
            PERF  DO_BAKR            .YES
         WHEN  CLI,XCELL+1,EQ,X'49'  .EREG?
            PERF  DO_EREG            .YES
         WHEN  CLI,XCELL+1,EQ,X'18'  .PC?
            PERF  EXEC_PC
         WHEN  CLI,XCELL+1,EQ,X'1A'  .CFC?
            PERF  EXEC_CFC
         WHEN  CLI,XCELL+1,EQ,X'22'  .IF IPM, WE ALREADY HAVE TRUE
            IC    R1,XCELL+3        .CC AND PROGAM MASKS
```

```
            N     R1,=XL4'FØ'
            SRL   R1,2
            LA    R2,REGTBL(R1)        .GET VALUE OF OPERAND REG
            MVC   Ø(1,R2),REALCC
         WHEN  CLI,XCELL+1,EQ,X'4D',OR, COPY ACCESS REGS    (CPYA)     +
               CLI,XCELL+1,EQ,X'4E',OR, SET ACCESS REG.     (SAR)      +
               CLI,XCELL+1,EQ,X'4F'    .EXTRACT ACCESS REG. (EAR)
            PERF  EXEC_CPYA_SAR_EAR
         WHEN  NONE                    .ANY OTHER B2XX
            PERF  PRIME_B2             .PRIME THE REGS FOR INSTRUCTION
            IF    TM,FLAGS+1,B2RØBIT,O    .RØ IMPLICITLY USED?
               L     RØ,REGTBL
            ENDIF
            IF    TM,FLAGS+1,B2R1BIT,O    .R1 IMPLICITLY USED?
               L     R1,REGTBL+4
            ENDIF
            RUN_INST
            IF    TM,FLAGS+1,B2RØBIT,O    .RØ IMPLICITLY USED?
               ST    RØ,REGTBL
            ENDIF
            IF    TM,FLAGS+1,B2R1BIT,O    .R1 IMPLICITLY USED?
               ST    R1,REGTBL+4
            ENDIF
            PERF  AFTER_B2             .SAVE ANY USED REGS
         ENDSEL
         PERF  PRT_B2                  .GO PRINT THE RESULTS
         IF    TM,FLAGS,BRBIT,Z
            LA    R9,4(,R9)               .INCREMENT INSTR PTR
         ENDIF
         MODEXIT
         EJECT
EXEC_PC  MODENTRY
         PERF  WRITE
         MVC   PRTLINE(34),=C' ****** REGISTERS BEFORE PC ******'
         PERF  DUMPREGS
         LA    R14,PC_UNSTACK
         BSM   14,Ø                        .GET CURRENT AMODE
         BAKR  14,Ø
         LM    RØ,R6,REGTBL          .RESTORE MOST REGS, SINCE
         LM    R8,R15,REGTBL+8*4     .WE WILL NOW LOSE CONTROL
         LAM   RØ,R6,AR_SAVE
         LAM   R8,R15,AR_SAVE+8*4
         EX    Ø,XCELL               .WE USE R7, CANNOT CHANGE
         EREG  R7,R7
         STM   RØ,R6,REGTBL          .SAVE ALL REGS AGAIN
         STM   R13,R15,REGTBL+13*4
         STAM  RØ,R6,AR_SAVE
         STAM  R8,R15,AR_SAVE+8*4
         PR
PC_UNSTACK     DS  ØH
         MODEXIT
         EJECT
```

```
EXEC_CFC MODENTRY
         LH    R1,CODEFLD+2
         LR    R15,R1
         IF    N,R1,=A(X'F000'),NZ
         N     R15,=A(X'FFF')
         O     R15,=A(X'6000')
         STCM  R15,3,CODEFLD+2
         SRL   R1,12-2
         L     R6,REGTBL(R1)
         ENDIF
         LM    R1,R3,REGTBL+4
         IF    IAC,R15,NZ
            LAM   R1,R1,AR_SAVE+4
         ENDIF
         RUN_INST
         STM   R1,R3,REGTBL+4
         MODEXIT
         EJECT
EXEC_CPYA_SAR_EAR  MODENTRY
         XR    R1,R1
         IC    R1,XCELL+3
         LR    R15,R1
         N     R15,=A(X'F0')
         SRL   R1,4-2
         N     R15,=A(X'0F')
         SLL   R15,2
         SELECT
         WHEN  CLI,XCELL+1,EQ,X'4D'          .CPYA - BOTH REGS ARE AR'S
            LA    R1,AR_SAVE(R1)
            LA    R15,AR_SAVE(R15)
         WHEN  CLI,XCELL+1,EQ,X'4E'          .SAR - 1ST REG AR, 2ND GPR
            LA    R1,AR_SAVE(R1)
            LA    R15,REGTBL(R15)
         WHEN  NONE                          .EAR - 1ST REG GPR, 2ND AR
            LA    R1,REGTBL(R1)
            LA    R15,AR_SAVE(R15)
         ENDSEL
         MVC   0(4,R1),0(R15)
         MODEXIT
         EJECT
PRIME_B2 MODENTRY
         XR    R14,R14
         IC    R14,XCELL+1
         A     R14,=A(AR_B2_00)
         SELECT
         WHEN  TM,FLAGS+1,B2STGBIT+B2ADRBIT,NZ   .DOES IT USE STG/ADDR?
            ICM   R1,B'0011',XCELL+2
            LR    R15,R1
            SELECT EVERY
            WHEN  N,R1,=XL4'F000',NZ         .BASE REG NOT R0
               N     R15,=A(X'FFF')          .LEAVE OFFSET ONLY
               O     R15,=A(X'6000')         .THEN USE R6, WHATEVER
```

```
            STH   R15,CODEFLD+2          .WAS ORIGININALLY SPEC'D
            SRL   R1,12-2
            L     R6,REGTBL(R1)     .AND PRIME R6 CORRECTLY
            IF    TM,Ø(R14),AR_B2,O,AND,IAC,R4,NZ
               L     R15,AR_SAVE(R1)
               SAR   R6,R15
            ENDIF
         WHEN  TM,FLAGS+1,B2R1BIT,O   .R1 IMPLICITLY USED?
            L     R1,REGTBL+4
         ENDSEL
      WHEN  TM,FLAGS+1,B2RBIT,O     .DOES INTRUCTION USE REGS?.
         LA    RØ,X'2Ø'             .YES, ALWAYS USE R2 AS FIRST REG
         IC    R1,XCELL+3
         N     R1,=A(X'FØ')
         SRL   R1,2
         L     R2,REGTBL(R1)     .PRIME R2
         L     R3,REGTBL+4(R1)   .AND R3, IN CASE OF DBL REG INST
         IF    TM,Ø(R14),AR_R1+AR_UR1,NZ,AND,IAC,R15,NZ
            LA    R15,AR_SAVE(R1)
            LAM   R2,R3,Ø(R15)
         ENDIF
         IF    TM,FLAGS+1,B2R2BIT,O .2ND REG USED?
            A     RØ,=F'4'           .SO WE WILL USE R4
            IC    R1,XCELL+3
            N     R1,=F'15'
            SLL   R1,2
            L     R4,REGTBL(R1)     .PRIME R4
            L     R5,REGTBL+4(R1)   .R5 TOO, IN CASE OF DBL REG
            IF    TM,Ø(R14),AR_R2+AR_UR2,NZ,AND,IAC,R15,NZ
               LA    R15,AR_SAVE(R1)
               LAM   R4,R5,Ø(R15)
            ENDIF
         ENDIF
         STC   RØ,CODEFLD+3         .MODIFY THE INST - USE R2 (+R4)
      ENDSEL
      MODEXIT
      EJECT
AFTER_B2 MODENTRY
      IF    TM,FLAGS+1,B2RØBIT,O    .RØ IMPLICITLY USED?
         ST    RØ,REGTBL
      ENDIF
      XR    R14,R14
      IC    R14,XCELL+1
      A     R14,=A(AR_B2_ØØ)
      SELECT    EVERY
      WHEN  TM,FLAGS+1,B2R1BIT,O
         ST    R1,REGTBL+4          .R1 MAY HAVE BEEN MODIFIED
      WHEN  TM,FLAGS+1,B2RBIT,O
         IC    R1,XCELL+3
         N     R1,=XL4'FØ'
         SRL   R1,2
         ST    R2,REGTBL(R1)        .SAVE (MODIFIED) REG S
```

```
            ST    R3,REGTBL+4(R1)      .AND ITS DBL PARTNER
            IF    TM,Ø(R14),AR_R1+AR_UR1,NZ,AND,IAC,R15,NZ
               LA    R15,AR_SAVE(R1)
               STAM  R2,R3,Ø(R15)
            ENDIF
            IF    TM,FLAGS+1,B2R2BIT,O .DOES IT USE 2 REGS?
               IC    R1,XCELL+3        .YES, SO SAVE THEM TOO,
               N     R1,=F'15'         .IN CASE THEY
               SLL   R1,2              .WERE MODIFIED
               ST    R4,REGTBL(R1)
               ST    R5,REGTBL+4(R1)
               IF    TM,Ø(R14),AR_R2+AR_UR2,NZ,AND,IAC,R15,NZ
                  LA    R15,AR_SAVE(R1)
                  STAM  R2,R3,Ø(R15)
               ENDIF
            ENDIF
         WHEN  CLI,XCELL+1,EQ,X'4A'    .ESTA?
            LA    RØ,PR_STACK
            L     R14,CUR_PR
            IF    CR,R14,GE,RØ
               IC    R1,XCELL+3
               LR    R15,R1
               N     R1,=F'15'
               SLL   R1,2
               L     R1,OLDREGS(R1)
               N     R15,=A(X'FØ')
               SRL   R15,2
               LA    R15,REGTBL(R15)
               SELECT
               WHEN  C,R1,EQ,=F'1'     .EXTRACT PSW
                  MVC   4(4,R15),Ø(R14)
               WHEN  C,R1,EQ,=F'2'     .EXTRACT BRANCH ADDR
                  MVC   4(4,R15),L'PR_STACK-4(R14)
               ENDSEL
            ENDIF
         ENDSEL
         MODEXIT
         EJECT
DO_BAKR  MODENTRY
         MVC   OLDREGS(16*4),REGTBL
         PERF  BAKRSAVE
         STM   RØ,R14,TEMPREGS
         LR    R1,R7
         L     R15,=A(PR_RET)
         BSM   R15,Ø
         LAM   RØ,R15,AR_SAVE
         LM    R2,R14,REGTBL+2*4
         BAKR  R15,Ø
         IAC   R15
         SAC   Ø
         STM   R2,R14,REGTBL+2*4-MYSAVE(R1)
         LM    RØ,R14,TEMPREGS-MYSAVE(R1)
```

```
         LAM   RØ,R15,=16F'Ø'
         SAC   Ø(R15)
         L     R1,CUR_PR
         L     R9,L'PR_STACK-4(,R1)
         LR    R1,R9
         N     R1,=A(X'8ØØØØØØØ')
         LA    R15,BAKRMODE
         OR    R15,R1
         BSM   RØ,R15
BAKRMODE DS    ØH
         MODEXIT
         EJECT
BAKRSAVE MODENTRY
         IF    ICM,R3,15,CUR_PR,Z
            LA    R3,PR_STACK
         ELSE
            LA    R1,L'PR_STACK
            MH    R1,=H'4Ø'
            LA    R1,PR_STACK(R1)
            LA    R3,L'PR_STACK(,R3)
            IF    CR,R3,GE,R1
               WTO   'BAKR STACK OVERFLOW'
               ABEND 666,DUMP
            ENDIF
         ENDIF
         ST    R3,CUR_PR
         IF    TM,XCELL+3,X'FØ',Z
            LA    R14,4(,R9)
            BSM   R14,Ø
         ELSE
            XR    R1,R1
            IC    R1,XCELL+3
            N     R1,=A(X'FØ')
            SRL   R1,2
            L     R14,REGTBL(R1)
         ENDIF
         IF    TM,XCELL+3,X'ØF',Z
            LA    R2,4(,R9)
            BSM   R2,Ø
         ELSE
```

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

Pieter Wiid
Advisory Systems Engineer
Persetel (South Africa)

MVS news

IBM has released Version 2.2 of its Tivoli Manager for MQSeries for OS/390, with more functionality and support for S/390 systems, and management control of company-wide MQSeries from one desktop machine.

Tivoli Enterprise Console (TEC) Adapter for OS/390 support has been improved to enable NetView event notification and automation, in addition to, or instead of, using the TEC server in a distributed architecture. Basically, the capabilities available previously for distributed Unix and Windows platforms, like using the same interface as that used in the distributed environment, providing the means to manage everything from the single PC are now available for MQSeries on OS/390. The Tivoli Manager for MQSeries can correlate data from infrastructure components such as network hardware with other middleware and application events and present the information in the context of a specific business system. It allows MQSeries requests for information to pass from a managed node to an OS/390 system for execution.

For further information contact:
Tivoli Systems Inc, 9442 Capital of Texas Highway, North Austin, TX 78759, USA.
Tel: (512) 436 8000
Fax: (512)794 0623 or
Tivoli Systems (UK) Ltd, Sefton Park, Bells Hill, Stoke Poges, Buckinghamshire, SL2 4HD, UK.
Tel: (01753) 896 943
Fax: (01753) 896 882
http://www.tivoli.com

* * *

Iona has announced its OrbixPL/I for OS/390, adding CORBA middleware functions to the PL/I development environments. It is said to allow developers to integrate existing mainframe applications with other server platforms and the Internet. The pre-release programme follows similar efforts with Orbix for OS/390 and the Orbix IMS and CICS Adapters.

OrbixPL/I for OS/390, we're told, supports interface-based application development by using the CORBA standard Interface Definition Language and Internet InterORB Protocol for interface definition and execution between the mainframe and other platforms.

Iona's OrbixPL/I for OS/390 features both an IDL to PL/I compiler and a PL/I object adapter that exposes key Orbix interfaces in PL/I to help develop PL/I Orbix applications. The product currently provides support for batch PL/I clients and servers and PL/I IMS and CICS servers.

For further information contact:
Iona Technologies, Inc, 60 Aberdeen Avenue, Cambridge, MA 02138, USA.
Tel: (617) 949 9000
Fax: (617) 949 9001 or
Iona Technologies (UK) Ltd, The Atrium Court, Apex Plaza, Reading, RG1 1AX, UK.
Tel: (0118) 925 4241
Fax: (0118) 958 7724
http://www.iona.com

* * *