131

August 1997

In this issue

MVS update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: stevep_xephon@compuserve.com

North American office

Xephon
1301 West Highway 407, Suite 201-450
Lewisville, TX 75067, USA
Telephone: 940 455 7050

Australian office

Xephon/RSM
PO Box 6258, Halifax Street
Adelaide, SA 5000
Australia
Telephone: 08 223 1391

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

Steve Piggott

Disclaimer

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

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £310.00 in the UK; $465.00 in the USA and Canada; £316.00 in Europe; £322.00 in Australasia and Japan; and £320.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £27.00 ($39.00) each including postage.

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.

Printed in England.

An exit for the expiration of a time limit

INTRODUCTION

IEFUTL receives control from the operating system upon the expiration of one of three time limits: a job's execution time limit as specified on its JOB statement, a step's execution time limit as specified on its EXEC statement (if not specified the job statement limit is taken), or a job's continuous wait time limit as specified in member SMFPRM00's JWT parameter in SYS1.PARMLIB. Complete documentation of the environmental and programming requirements for IEFUTL are contained in the IBM publication *Installation Exits* SC28-1459.

The source code for IEFUTL included with this article is primarily concerned with the expiration of the continuous wait time for TSO users. If a TSO user has been given approximately 30 minutes of continuous 'think' time, he or she is considered to be brain-dead and the TSO session is summarily terminated. In the interim, between the start and end of each user's allotted 30 minutes' worth of think time (approximately) at approximately three-minute intervals, an informational message containing the name of each waiting task and the approximate time it began waiting is sent to all operating system consoles in computer operations. Technical personnel at installations with a large number of TSO users may want to consider reducing the frequency with which IEFUTL displays messages for cogitating TSO users or eliminating the messages altogether.

Have you noticed that I use 'approximate' a lot? The reason I do is that entry to IEFUTL is not unerringly precise as it once was, and IBM 'support' personnel have informed me that there are no plans to make it so. I danced a merry jig with a few of IBM's support centre personnel over this one. Perhaps they get signed off of TSO a lot also. No one I communicated with at IBM could adequately explain to me why I, in a simple batch program, can receive control precisely every three minutes, or whatever time interval I happen to code, and cannot for the life of me receive control in IEFUTL every three minutes on the dot. Anyway, three minutes was selected as the value for JWT because that

is approximately the empirically-derived interval at which IEFUTL seems to receive control. Precision in this regard would enable someone to view SYSLOG, locate the point in time at which a job entered its wait state, and ascertain the reason for it. An approximation is the best I have been able to do, so far!

Jobs whose estimated JOB or EXEC statement time limits expire are unequivocally terminated – not much programming was required for that bit of processing. Batch jobs are allowed to wait interminably, or until an operator tires of seeing '...has been waiting...' messages spill across his or her screen and onto the floor and cancels them, or satisfies their pending requirements.

This IEFUTL exit depends entirely on the presence of an area obtained during a job's initial entry to my IEFUSI exit. A similar work area for IEFUTL must have been obtained in your IEFUSI exit as well. Three fields (KEEPTARY, KEEPCIAO, and KEEPWAIT) must be reset to binary zeros whenever a job changes steps. This is to be done in IEFUSI as well. In order to detect actual changes in a job's status, 30 seconds of time is added to JWT to determine whether a job has waited longer than JWT. I decided that it does not matter since there are no precise times available anyway.

SAMPLE OUTPUT

This is a sample of the output one can expect to be generated by IEFUTL. As can be readily seen, IEFUTL will provide only an approximation of the time a task enters a wait state, but it has proven to be helpful to us. During the time frame depicted below, can you spot the task that changed steps?

```
10:23:29.33 JOB00487 @15 XEPHON1  THIS IS A WAIT TEST
...
10:24:48.15 JOB00488 @16 XEPHON2  THIS IS ANOTHER WAIT TEST
...
10:27:46.38 JOB00487  XEPHON1   STEP   3 BOGUS00  WAITING SINCE 10:24:47.59
...
10:29:20.75 JOB00488  XEPHON2   STEP   3 BOGUS01  WAITING SINCE 10:26:21.96
...
10:30:55.12 JOB00487  XEPHON1   STEP   3 BOGUS00  WAITING SINCE 10:24:47.59
...
10:31:45.64 JOB00490 *IEF233A M 04F3,XEPHON,,XEPHON3,G,XEPHON3.WAIT.TEST
...
```

```
10:32:29.49 JOB00488  XEPHON2   STEP   3 BOGUS01  WAITING SINCE 10:26:21.96
...
10:33:34.79 JOB00487 @18 XEPHON1  THIS IS A WAIT TEST
...
10:35:38.24 JOB00488  XEPHON2   STEP   3 BOGUS01  WAITING SINCE 10:26:21.96
10:35:38.24 JOB00490  XEPHON3   STEP   3 BOGUS02  WAITING SINCE 10:32:39.45
...
10:37:12.61 JOB00487  XEPHON1   STEP   4 BOGUS00  WAITING SINCE 10:34:13.82
```

PROGRAM SOURCE CODE IEFUTL

```
        TITLE  'IEFUTL - PROCESS TASKS WITH EXCESSIVE WAIT/CPU TIMES'
*********************************************************************
*       IEFUTL - SMF JOB EXIT...ENTERED FOR WAIT/CPU EXCEEDERS      *
*              ALLOW JOBS TO CONTINUE WITH 3 MIN EXTENSIONS,        *
*                 FOREVER, WHEN SYS1.PARMLIB(SMFPRM00) JWT (JOB     *
*                 WAIT TIME) IS EXCEEDED...WITH CONSOLE WTO MSG.    *
*              THE ASSUMPTION IS THAT JWT IS SPECIFIED SMALL...EG   *
*                 JWT(0003), IMPLIES 3 MIN CONTIGUOUS WAIT BEFORE   *
*                 IEFUTL RECEIVES THE INITIAL CALL FOR THE JOB.     *
*              SMF CHECKS FOR EXCEEDERS EVERY 90 SEC, CUMULATIVE WAIT*
*                 TIME REPORTED WILL BE KINDA CORRECT IF WE GO WITH *
*                 A MULTIPLE OF 3MIN...COUNT IS RESET AT STEP CHANGE*
*              TSO SESSIONS RECEIVE NINE CONTIGUOUS EXTENSIONS;     *
*                 AFTER THAT CONTIGUOUS TIME IS EXCEEDED, THE       *
*                 SESSION IS CANCELLED.                             *
*              CANCEL JOBS IF JOB (STEP) TIME LIMIT IS EXCEEDED.    *
*           REGISTERS:                                              *
*                       R0 - R7 WORK.                               *
*                       R8 = UNUSED                                 *
*                       R9  = A(USER ACCOUNTING AREA - KEEPSECT)    *
*                       R10 = A(GETMAINED AREA)                     *
*                       R11 = A(COMMON EXIT PARM AREA - IEFJMR)     *
*                       R12 = A(IEFUTL) MY BASE                     *
*                       R13 = A(MY SAVEAREA). R14 = RETURN.         *
*                       R15 = RETURN CODE.                          *
*           PATCH AREA INIT. TO 'ZAP*'                              *
*                ATTRIBUTES = SCHEDULER KEY 0, REENTRANT,           *
*                   SUPERVISOR STATE, ENABLED                       *
*             ENTRY FROM INITIATOR VIA MODULE IEATLEXT              *
*             INPUT:  REGISTER 1 POINTS TO A LIST OF FULL WORDS     *
*                THE FIRST OF WHICH POINTS TO THE SMF COMMON EXIT   *
*                PARAMETER AREA THAT IS MAPPED BY THE IEFJMR MACRO. *
*             OUTPUT:  R15 = RETURN CODE, R1= TIME EXTENSION IN SEC.*
*********************************************************************
IEFUTL   CSECT
IEFUTL   AMODE 31                ADDRESSING MODE
IEFUTL   RMODE ANY               RESIDENCY MODE
         SAVE  (14,12),,IEFUTL.IPO.&SYSTIME._&SYSDATE SAVE REGS
         LR    R12,R15            LOAD REGISTER 12 FROM 15
```

```
         USING IEFUTL,R12          ESTABLISH IEFUTL ADDRESSABILITY
         USING PSA,RØ              ESTABLISH PSA ADDRESSABILITY
*************************************************************************
*        ESTABLISH ADDRESSABILITY TO SMF PARAMETER AREA DSECT           *
*        EVALUATE REGISTER Ø = Ø - JOB CPU TIME EXCEEDED                *
*                           = 4 - STEP CPU TIME EXCEEDED                *
*                           = 8 - JOB WAIT TIME EXCEEDED                *
*************************************************************************
         L     R11,DØ(R1)          LOAD POINTER TO PARAMETER AREA
         USING JMR,R11             ADDRESSABILITY SMF PARM DSECT
         ICM   R9,15,JMRUCOM       LOAD POINTER TO KEEPSECT
         BE    PPGERROR            BRANCH IF NOT AVAILABLE
         USING KEEPSECT,R9         ADDRESS TO TENN ACCOUNTING DSECT
         LA    R1,KEEPLEN(R9)      POINT TO WORKAREA
         USING WORKAREA,R1         ESTABLISH WORKAREA ADDRESSABILITY
*************************************************************************
*        ENSURE THAT THIS AREA BELONGS TO ME                            *
*************************************************************************
         CLC   CLAMLOVE,=CL4'LOVE' TEST IF JMRUCOM HAS BEEN CORRUPTED
         BNE   PPGERROR            BRANCH IF IT HAS
         DROP  R1                  FORGET WORKAREA
         L     R5,PSAAOLD          CURRENT ASCB
         USING ASCB,R5             ESTABLISH ASCB ADDRESSABILITY
         L     R7,ASCBSWTL         FETCH STEP WAIT TIME
         LR    R3,RØ               PRESERVE REASON FOR ENTRY TO IEFUTL
         TIME  STCK,KEEPCONV       OBTAIN TIME OF ENTRY TO IEFUTL
*************************************************************************
*        FIELDS ARE INITIALIZED TO BINARY ZEROES IN IEFUSI AT THE       *
*        BEGINNING OF EACH STEP OF A JOB                                *
*************************************************************************
         CLC   KEEPTARY,FULLØ      TEST IF 'FIRST' ENTRY
         BNE   PPGSKIP             BYPASS SETTING START OF WAIT
         L     R6,KEEPCONV         'BEGINNING' OF WAIT
         SR    R6,R7               COMPUTE APPROXIMATE 'BEGINNING'
         ST    R6,KEEPTARY         STOW IT
         XC    KEEPWAIT,KEEPWAIT   RESET FOR TSO USERS
PPGSKIP  C     R3,FULL8            SEE IF JOB/STEP WAIT TIME EXCEEDED
         L     R6,KEEPCONV         FETCH START OF WAIT INTERVAL - HA!
         BNE   CANCEL              NO.....GO CANCEL JOB
*************************************************************************
*        WAIT TIME EXCEEDED                                             *
*        - ALLOW TSO SESSION TO BE EXTENDED NINE CONTIGUOUS TIMES       *
*        - PROVIDE JOBS WITH 3 MINUTE EXTENSIONS AND NOTIFY OPERATOR    *
*        FORMAT A MESSAGE IN VIRTUAL STORAGE OBTAINED HERE              *
*************************************************************************
         STORAGE OBTAIN,LENGTH=WRKLJ,SP=241 GETMAIN AREA FOR WTO DATA
         LR    R1Ø,R1              SAVE ADDRESS OF GETMAINED AREA
         USING WRKAREA,R1Ø
         MVC   WAITMSG(MSGL),JOBMSG  FORMAT WORKAREA
         MVC   JN(8),JMRJOB        MOVE JOB NAME TO MSG
         MVC   USER(8),JMRUSEID    MOVE USER TO MSG
```

```
         SR    R3,R3               ZERO WORK REGISTER
         IC    R3,JMRSTEP          PICK UP STEP NUMBER
         CVD   R3,DBLWORD          CONVERT
         MVC   STEPN,=XL4'40202120' PLACE EDIT PATTERN INTO WTO AREA
         ED    STEPN,DBLWORD+6     EDIT STEP NUMBER INTO WRITE-TO-OPER
         DROP  R5                  FORGET ASCB
*********************************************************************
*        COMPUTE THE APPROXIMATE BEGINNING OF A WAIT STATE          *
*********************************************************************
         LR    R5,R6               REMEMBER TIME OF ENTRY TO IEFUTL
         ICM   R4,15,KEEPCIAO      FETCH TIME OF LAST ENTRY TO IEFUTL
         ST    R5,KEEPCIAO         SET TO TIME OF CURRENT ENTRY
         BE    PPGOLD              BRANCH IF FIRST ENTRY
         SR    R6,R4               COMPUTE LENGTH OF WAIT
         LA    R15,32(R7)          ADD ABOUT THIRTY SECONDS
         CR    R6,R15              TEST IF FRESH ENTRY
         BNH   PPGOLD              BRANCH IF LESS THAN JWT
         SR    R5,R7               COMPUTE APPROXIMATE 'BEGINNING'
         ST    R5,KEEPTARY         SET NEW BEGINNING WAIT TIME
         XC    KEEPWAIT,KEEPWAIT   RESET TSO ENTRY
*********************************************************************
*        CONVERT TIME TO AN UNDERSTANDABLE FORMAT; STOW IN WTO      *
*        DISPLAY 'WAITING' MESSAGE                                  *
*        RELEASE STORAGE OBTAINED IN IEFUTL                         *
*********************************************************************
PPGOLD   STCKCONV STCKVAL=KEEPTARY,CONVVAL=KEEPCONV,TIMETYPE=DEC,      M
               DATETYPE=MMDDYYYY,MF=(E,STCKLIST)
         MVC   NMINS,=XL12'4021207A20207A20204B2020' TIME'S EDIT PAT
         ED    NMINS,KEEPCONV      TIME TASK ENTERED A WAIT STATE
         LA    R1,WAITMSG          MESSAGE ADR
         SVC   WTOSVC              ISSUE WTO SVC
         STORAGE RELEASE,LENGTH=WRKLJ,ADDR=(R10),SP=241 FREE WTO WKAREA
         TM    JMROPT,JMRFIND      SEE IF TSO SESSION
         BNO   JCANCEL             NO..GIVE JOB 3 MIN EXTENTIONS
*********************************************************************
*        ALLOW TSO SESSIONS 9 3 MINUTE EXTENSIONS, THEN CANCEL THEM *
*        (UNLESS THEY ARE SACROSANCT ONES)                          *
*********************************************************************
         CLC   JMRJOB(7),=C'AG03RMF' OPERATIONS MOSTLY USING RMF MON
         BE    JCANCEL                   YES, DON'T LOGOFF
         CLC   JMRJOB(7),=C'DCP0000' EXTRA-SPECIAL TENNCARE USER-ID
         BE    JCANCEL                   YES, DON'T LOGOFF
         CLC   JMRJOB(7),=C'DCP0001' EXTRA-SPECIAL TENNCARE USER-ID
         BE    JCANCEL                   YES, DON'T LOGOFF
         LH    R14,KEEPWAIT        FETCH WAIT COUNT
         LA    R14,1(R14)          INCREMENT IT BY ONE
         STH   R14,KEEPWAIT        REVISE WAIT COUNT
         C     R14,FULL8           TEST IF WAIT TIME HAS BEEN EXCEEDED
         BH    CANCEL              YES...GO CANCEL TSO SESSION
*********************************************************************
*        PROVIDE A GRATUITOUS RESPITE FROM TERMINATION FOR THIS TASK *
*********************************************************************
```

```
JCANCEL  L     R2,RET8             INDICATE RETURN CODE OF 8 -
         L     R1,WAITJOB          LOAD TIME EXTENSION IN SECONDS
         B     RETURN              DEPART
*********************************************************************
*        TERMINATE THIS TASK                                        *
*********************************************************************
CANCEL   DS    0H
         SR    R2,R2               INDICATE RETURN CODE OF ZERO -
*                                  CONTINUE CANCEL OF JOB....
*********************************************************************
*        NORMAL END PROCESSING                                      *
*********************************************************************
RETURN   DS    0H
         L     R14,D12(,R13)       LOAD ADDRESS FOR RETURN
         LR    R15,R2              LOAD RETURN CODE FROM REGISTER 2
         LM    R2,R12,D28(R13)     RESTORE REGISTERS 2 TO 12
         BR    R14                 RETURN TO CALLER
PPGERROR WTO   'OIR714I  IEFUTL - ADDRESS OF KEEPSECT IS INVALID'
         B     JCANCEL             ALLOW TASK TO CONTINUE
         TITLE 'IEFUTL - CONSTANTS AND DSECTS'
*********************************************************************
*        CONSTANTS, DSECTS, AND OTHER SUCH JUNK                     *
*********************************************************************
FULL0    DC    F'0'
FULL8    DC    F'8'
RET8     DC    F'8'
WAITJOB  DC    F'180'          EXTEND JOB WAIT 3 MIN.(180 SEC)
PPGTRANS DC    C'0123456789ABCDEF'
****************************** MESSAGES ******************************
JOBMSG   DC    AL2(WTMSGLEN)
         DC    XL2'8000'
JN0      DC    CL8'JOBNAME '
         DC    CL2'  '
STEP0    DC    CL4'STEP'
STEPN0   DC    CL4' '
         DC    CL1' '
USER0    DC    CL8'AG03Z   '
         DC    CL1' '
         DC    C'WAITING SINCE'
NMINS0   DC    CL12' '
         DC    XL2'0400'
         DC    XL2'4000'
****************************** PATCH AREA ****************************
         DS    0F
PATCH    DC    8CL4'ZAP*'
********************** DSECTS (MAPPING MACROS) **********************
WRKAREA  DSECT
DBLWORD  DC    D'0'
DBLWORD1 DC    D'0'     WORK
SAV13    DC    F'0'
```

```
WAITMSG  DC    AL2(WTMSGLEN)
         DC    XL2'8000'
JN       DC    CL8'JOBNAME '
         DC    CL2'  '
STEP     DC    CL4'STEP'
STEPN    DC    CL4' '
         DC    CL1' '
USER     DC    CL8'AG03Z   '
         DC    CL1' '
         DC    C'WAITING SINCE'
NMINS    DC    CL12' '
WTMSGLEN EQU   *-WAITMSG
         DC    XL2'0400'
         DC    XL2'4000'
MSGL     EQU   *-WAITMSG
         DS    0F
STCKLIST STCKCONV MF=L
WRKLJ    EQU   *-WRKAREA
*********************************************************************
*        THE FOLLOWING DSECT DESCRIBES STORAGE WHICH IS ACQUIRED     *
*        DURING THE FIRST STEP OF THE JOB AND IS RELEASED WHEN THE   *
*        JOB ENDS.  THE ADDRESS OF THIS AREA IS KEPT IN THE COMMON   *
*        EXIT USER DATA FIELD OF THE COMMON EXIT TABLE.              *
*        STORAGE IS ACQUIRED IN IEFUSI AND RELEASED IN IEFACTRT.     *
*********************************************************************
KEEPSECT DSECT
KEEPJCT  DS    A                    ADDRESS OF JOB'S JCT IF HASP IS UP
KEEPSPAR DS    F                    SPARE
KEEPEXCP DS    F                    SUM OF ALL EXCPS(DA,TP,UR)
KEEPCPU  DS    F                    SUM OF CPU FOR ALL STEPS
KEEPBMP  DS    F                    BMP
KEEPINRT DS    H                    SAVE HASP INPUT ORIGIN
KEEPPRRT DS    H                    SAVE HASP PRINT ROUTE
KEEPPURT DS    H                    SAVE HASP PUNCH ROUTE
KEEPUSI  DS    X                    SAVE USI FLAGS
KEEPSMBF DS    X                    SMB PRINT FLAG
KEEPXXXX DS    X
KEEPYYYY DS    X
KEEPZZZZ DS    X
KEEPTMS  DS    F                    TMS ET AL WORK AREA
KEEPUTL  DS    F                    IEFUTL ET AL WORK AREA
         ORG   KEEPUTL
*                                   BEEN HERE BEFORE - SWITCHES
KEEPWAIT DS    H                    CONTIGUOUS WAIT TIME
KEEPXTRA DS    H
         ORG
KEEPTPR  DS    H                    TALLY AREA FOR SPECIFIC TAPE MOUNTS
KEEPPTM  DS    H                    TALLY AREA - NON-SPECIFIC TAPE MNTS
KEEPUSCT DS    H                    TALLY AREA FOR TAPE DRIVES USED
KEEPRSVD DS    H                    AVAILABLE
```

```
KEEPTARY DS    F                    HOLD AREA FOR ASCBEWST
KEEPCIAO DS    F                    HOLD AREA FOR PREVIOUS ASCBEWST
KEEPCONV DS    CL16                 WORK AREA FOR CONVERT OF WAIT-BEGIN
KEEPLEN  EQU   ((*-KEEPSECT+7)/8)*8 COMPUTE LENGTH FOR GET- & FREEMAIN
         TITLE 'WORK STORAGE DSECTS'
*********************************************************************
*        VIRTUAL STORAGE IS OBTAINED IN IEFUSI                      *
*********************************************************************
WORKAREA DSECT
         DS    40F                  REG SAVE AREA FOR CALLED RTNS
PATRACF  DS    20F                  AREA FOR LIST FORM OF RACF MACROS
PATWORD  DS    0F                   GENERAL PURPOSE WORD
PATBYTE  DS    X                    GENERAL PURPOSE FLAG AREA
PATSPARE DS    XL3                  GENERAL PURPOSE SPARE
CLAMLOVE DS    F                    LOCATOR
CLAMSTEP DS    F                    STEP SMF TYPE 30 RECORD
CLAMTYPE DS    C                    LAST SMF TYPE 30 RECORD
CLAMJOB  DS    AL3                  JOB SMF TYPE 30 RECORD
SAVE1    DS    46F                  SAVE AND WORK AREAS
SAVELAST DS    F                    ADDRESS OF SAVE AREA ABOVE US
TERMTIME DS    F
TERMDATE DS    F
CLAMWORK DS    2D
CLAMHOLD DS    CL8
MSGLEN   DC    AL2(L'MSG)
MSGADDR  DC    A(MSG)
TEMPD1   DS    D
         ORG   TEMPD1               SET UP FIELDS FOR DEVICE PROCESSING
TMPDEVC  DS    B
TMPDEVT  DS    B
TMPDEVAD DS    H
TMPCOUNT DS    F
TEMPD2   DS    D
DOUBLE   DS    D
WORKTIME DS    F
WORKDATE DS    PL4
RUNTIME  DS    F
ADDRLCT  DS    A                    HOLDS ADDRESS OF LCT
ADDREXD  DS    A                    HOLDS ADDRESS OF EXD
MSG      DS    CL80                 BUFFER FOR PRINTING MESSAGES
*        DEFINE ARGUMENT LISTS FOR ISDACTRT
         ORG
STEPARGS DS    0D                   START OF ARGUMENT LIST FOR STEP CALL
CPUTIME  DS    F                    CPU TIME FOR THE STEP
VIOEXCPS DS    F                    SUMMATION OF JES AND VIO EXCPS
DISKEXCP DS    F                    TOTAL OF EXCPS TO DISK DEVICES
DISKUSCT DS    H                    TOTAL OF MOUNTABLE DISK UNITS USED
DISKMONT DS    H                    TOTAL OF DISKS ACTUALLY MOUNTED-
TAPEEXCP DS    F                    TOTAL OF EXCPS TO TAPE DEVICES
TAPEUSCT DS    F                    TOTAL OF TAPE UNITS USED
```

```
URECEXCP DS    F                   TOTAL OF EXCPS TO UNIT REC DEVICES
         ORG   STEPARGS            GO BACK TO BEGINNING OF ARGS
JOBARGS  DS    ØD                  START OF ARGUMENT LIST FOR JOB CALL
CRDSREAD DS    F                   NUMBER OF CARDS READ BY HASP
PUNCHCRD DS    F                   NUMBER OF CARDS GENERATED BY HASP
PRNTLNES DS    F                   NUMBER OF LINES GENERATED BY HASP
PRNTCOPY DS    X                   NUMBER OF PRINT COPIES REQUESTED
         ORG   ,                   GET BACK TO NEXT AVAILABLE SLOT
*        DEFINE LIST OF ARGUMENTS RETURNED FROM ISDACTRT
RETRNARG DS    ØF                  BEGINNING OF LIST RETURNED
RETCOST  DS    F                   CRU   COST
RETOCOST DS    F                   CPU COST
RETXCOST DS    F                   EXCP COST
RETBCOST DS    F                   BMP COST
RETICOST DS    F                   COST OF CARDS READ
RETLCOST DS    F                   COST OF PRINTED LINES
RETCCOST DS    F                   COST OF PUNCHED CARDS
RETSCOST DS    F                   COST OF A SPECIFIC TAPE MOUNT
RETNCOST DS    F                   COST OF NON-SPECIFIC TAPE MOUNT
*        DEFINE WORK AREA FOR ISDACTRT (MUST REMAIN IN GIVEN ORDER)
CALIOTIM DS    F                   I/O EXCPS  *  (CRU/EXCP)
CALBPTIM DS    F                   BMP CALLS  *  (CRU/BMP CALLS)
CALFACPU DS    F                   CPU TIME   *  (CRU/CPU)
CALFACRU DS    F                   TOTAL CRU TIME 1/100 SEC
         ORG   ,                   GET TO LAST AVAILABLE SLOT
*        DEFINE LENGTH OF DYNAMIC STORAGE AREA
         DS    ØD                  FORCE DOUBLEWORD BDRY FOR LENGTH
WORKLEN  EQU   *-WORKAREA          COMPUTE LENGTH FOR GET-, FREEMAIN
CLEARLEN EQU   *-TEMPD1            AREA TO BE ZEROED AFTER GETMAIN
*************************** LOCAL EQUATES *****************************
DØ       EQU   Ø
D12      EQU   12
D28      EQU   28
ONE      EQU   1
WTOSVC   EQU   35
ZERO     EQU   Ø
***************************** REGISTER EQUATES ************************
         YREGS
***************************** OS CONTROL BLOCKS ***********************
         IHAASCB
         IHAPSA
         IEFJMR
         END   IEFUTL
```

Checking VRSs under DFSMSrmm

INTRODUCTION

Sometimes, in a DFSMSrmm environment, we need to know which Vital Record Specifications (VRSs) the installation has. We can use option 3.3.5 (VRS search) of the RMM panels to do it, but this information is provided in two panels and we need to press the left and right keys to be able to view all the information we require.

In order to avoid this and to get more into the VRS display, I have developed a simple REXX program called SRCHVRS that gives us detailed information such as: the VRS name, its location, owner, type, whether it's catalogued or not, count, retention type etc.

This program can be executed in a batch environment using IKJEFT01 as in the example or in an ISPF environment by keying TSO %SRCHVRS on the command line. The program uses an input parameter that is the name of the VRS to be searched or '*' to search all VRSs. If you choose to execute the program in the batch environment, the results can go directly to the SYSOUT queue. Otherwise, the program creates a sequential file that is browsed at the end, an example of which is shown in Figure 1.

JCL TO EXECUTE IN BATCH AND PRINT THE DATASET

```
//JOBREXX  JOB CLASS=B,MSGCLASS=X,NOTIFY=&SYSUID
//*
//BATCHREX EXEC PGM=IKJEFTØ1,REGION=4M
//SYSPROC  DD   DISP=SHR,DSN=YOUR.REXX.DATASET
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
   %SRCHVRS
/*
//TEST  IF (BATCHREX.RC = Ø) THEN
//*
//PRINT    EXEC PGM=ICEGENER
//SYSUT1   DD   DISP=SHR,DSN=&SYSUID..VRS.LIST
//SYSUT2   DD   SYSOUT=B
//SYSOUT   DD   SYSOUT=*
//SYSIN    DD   DUMMY
//IFNOT    ELSE
//END ENDIF
//*
```

```
BROWSE --                    ---------------------- LINE 00000000 COL 001 080
COMMAND ===>                                          SCROLL ===> CSR
******************************** TOP OF DATA *********************************
VRS Name                        Location  Owner    Type  WCtlg Count  Ret.Type
VRS Description                    NextVRS    Del.Date     Delay     LocDays
================================================================================
AA.TEST.VRS.**                  REMOTE    USERID1  GDG   YES      12  CYCLES
THIS IS ONLY A TEST VRS            NXTVRS     31/07/1997       5           3
--------------------------------------------------------------------------------
YY.EMPLOYEE.G501.**             HOME      USERID2  GDG   YES       5  CYCLES
                                             31/12/1999       0       99999
--------------------------------------------------------------------------------
PAY.MENT.FILE.**                ROBOT     USERID6  GDG   YES       5  CYCLES
PAYMENT DATASET                    OUTVRS     31/12/1999       0       99999
--------------------------------------------------------------------------------
ABRHSM.*.**                     ROBOT     USERID4  GDG   YES       1  DAYS
PROTECT  ABARS DATASETS            OUTVRS     31/12/1999       3       99999
--------------------------------------------------------------------------------
BACKUP.DAILY.**                 REMOTE    USERID3  GDG   YES       5  CYCLES
DAILY BACKUP                                 31/12/1999       0       99999
--------------------------------------------------------------------------------
MAIN.APPL.INST.**               HOME      USERID1  GDG   YES       5  CYCLES
                                             31/12/1999       0       99999
```

Figure 1: Sample output

SRCHVRS REXX EXEC

```
/*NOCOMMENT Rexx  */
/* This REXX EXEC lists all VRSs from the RMM control dataset, putting
  the information in a sequential file that is browsed before the end
  of the program. Can be run in a batch environment and the result is
  printable.                                                    */
parse upper arg vrs                          /* VRS to list      */
if vrs = '' then
   vrs = '*'                                 /* Search all VRSs  */
isispf =1;wskip = ' '
if sysvar("sysenv") = 'FORE' then            /* Test  the envir. */
   wcount = 21                               /* 21 lines - screen*/
else
  do
    wcount = 60                              /* 60 lines - batch */
    isispf = 0                               /* Isn't ISPF       */
    wskip  = 1                               /* Print control    */
  end
if sysdsn(userid().VRS.LIST) = 'OK' then
```

```
   do
     x=outtrap(lixo.)
     "DELETE ('"userid()".VRS.LIST'"            /* Delete the file, */
     x=outtrap(off)                             /* if it exists.    */
   end
f=0;count=0
sysauth.edgdate = 'EUROPEAN'
"RMM SS DSNAME("vrs") LIMIT(*) OWNER(*)"        /* Search all VRSs  */
call CHECKRC
do a = 1 to edg@vrs.0
   if count = 0 then
      do
       f=f+1                                    /* Format lines     */
       line.f = wskip||left('VRS Name',30)||'  Location  Owner',
                  '   Type  WCtlg Count  Ret.Type'
       f=f+1
       line.f = ' 'left('VRS Description',30)||'     NextVRS   ',
                   'Del.Date     Delay     LocDays'
       f=f+1
       line.f = ' '||copies('=',80)
       count=count+3
      end
   f=f+1
   "RMM LS DSNAME('"edg@vrs.a"')"               /* List specific VRS*/
                                                /* to get info.     */
   call CHECKRC
   line.f = ' '||left(edg@vrs.a,32)||left(edg@loc.a,10)||,
                 left(edg@own.a,9)||left(edg@typ.a,6)||,
                 left(edg@rwc.a,7)||right(edg@vrc,4)||'  '||edg@ret
   f = f + 1
   line.f = ' '||left(edg@desc,35)||left(edg@nvrs,11)||edg@ddt,
               '  'right(edg@vdd,5)'      ' right(edg@sc1,5)
   f=f+1
   line.f = ' '||copies('-',80)
   count=count+3
   if count = wcount then                       /* Skip  control    */
      count=0
end
                                                /* Create file      */
"ALLOC DA('"userid()".VRS.LIST') F(FILE1) NEW SPACE(1,1) RECFM(F,B)",
"LRECL(80) DSORG(PS) UNIT(WORK)"
"EXECIO * DISKW FILE1 (FINIS STEM line.)"
"FREE F(FILE1)"
if isispf then                                  /* Envir. is ISPF ? */
   do
     address ISPEXEC
     "BROWSE DATASET('"userid()".VRS.LIST')"    /* Browse the file  */
   end
exit
/**/
```

```
CHECKRC:                                        /* Verify ret. codes*/
select
  when rc = 4 then
    zedlmsg = "Subcommand completed but some operands may have been ignored or",
              "modified. Check the reason code. Rcode -> "edg@rc
  when rc = 8 then
    zedlmsg = "You're not authorized to issue the command."
  when rc = 12 then
    zedlmsg = "There's an error in subcommand. Check the reason code.",
              " Rcode -> "edg@rc
  when rc = 16 then
    zedlmsg = "Error. The DFSMSrmm subsystem is not active."
  when rc = 2Ø then
    zedlmsg = "Error. Incomplete or invalid data and the TSO user",
              "has set NOPROMPT.".
  when rc = 24 then
    zedlmsg = "Error. The TSO subcommand is not APF authorized."
  when rc = 28 then
    zedlmsg = "Error. The user has hit the attention key."
  otherwise
    return
end
if isispf then                                  /* ISPF? Msg on panel*/
   do
     zedsmsg = "Press PF1 !!"
     "ISPEXEC SETMSG MSG(ISRZØØ1)"
     exit
   end
else                                            /* Batch? Normal way */
   do
     say zedlmsg
     exit
   end
RETURN
```

Manoel Augusto Cunha
Systems Programmer
Companhia de Seguros Bonança (Portugal)

Simplified charge-back system

BACKGROUND

We used to have a MICS database for our charge-back system to keep track of batch processing. However, to reduce costs, we recently dropped the MICS and SAS software products. As we would still like to charge our users, we have developed a simplified version of a charge-back system.

A SELCOPY program is used to extract the SMF batch job termination record – SELCOPY is a utility program from Compute (Bridgend), you can use an alternative if you want. A REXX program then reads the raw information and extracts the job name, date, CPU time, and EXCP I/Os. Data is sorted by cost centre, job name, and date. The sorted data is then fed into two COBOL programs to generate a charge-back report: PMBAT80 accumulates charges on a per job per day basis and PMBAT90 prints a summary total for all cost centres for the month. We also use OGL and PPFA to enhance the presentation of the reports.

To ensure all users submit jobs with the proper charge code, TSO user exit IKJEFF10 has been written. If the JCL is submitted without a proper charge code, the submission will be rejected.

TSO USER EXIT – IKJEFF10

```
IKJEFF1Ø CSECT
* LINK TO SYS1.LINKLIB & USE OMEGAMON TO MAKE IT EFFECTIVE WITHOUT IPL.
* 1997 CHART OF ACCOUNTS.
* ----------------------
* MAGAZINES:
* ----------
* 18Ø94ØØ - CENTRE
* 181Ø1ØØ - BODYSHOP
* 181Ø2ØØ - CDN. UNDERWRITER
* .
* 2853ØØ - NORTHERN MINER ONLINE
* 2854ØØ - NORTHERN MINER CONF.
* 2854ØØ - NORTHERN MINER DENV.
* INFORMATION PRODUCTS:
* --------------------
* 3818ØØ - DAILY OIL BULLETIN
* .
```

```
* .
* 3868000 - CDN. DIR. SCHOOLS
* 3869600 - ALMANAC & SOURCE/96
* 3869700 - ALMANAC & SOURCE/97
* 3875000 - ADMIN EXDEX
* ENTRY POINTS -
*         IKJEFF10
* INPUT REGISTER
*    0     - UNPREDICTABLE
*    1     - ADDRESS OF AN EIGHT WORD PARAMETER LIST
*    2-12  - UNPREDICTABLE
*   13     - ADDRESS OF A REGISTER SAVE AREA
*   14     - RETURN ADDRESS
*   15     - EXIT ENTRY POINT ADDRESS
* OUTPUT
*    REGISTER 15 MUST CONTAIN ONE OF THE FOLLOWING RETURN CODES:
*    0     - CONTINUE PROCESSING
*    4     - INVOKE THE EXIT AGAIN TO OBTAIN ANOTHER STATEMENT
*    8     - THE SUBMIT PROCESSOR DISPLAYS MESSAGE IKJ56283I, AND
*          - INVOKES THE EXIT AGAIN
*   12     - THE SUBMIT PROCESSOR DISPLAY IKJ56280A, OBTAINS A RESPONSE
*          - FROM THE USER, AND INVOKES THE EXIT AGAIN.
*   16     - END PROCESSING OF THE SUBMIT COMMAND.
* WORK REGISTER
*    0     - N/A
*    1     - WTO/GETMAIN
*    2-3   - N/A
*    4-7   - N/A
*    8-10  - BXLE LOOP CONTROL
*   11-12  - N/A
*   12     - ADDRESS TO PARAMETER LIST
*   13     - REGISTER SAVE AREA
*   14     - RETURN ADDRESS
*   15     - SET RETURN CODE
         USING *,R15
         B     HERE                    BRANCH AROUND MODULE NAME
         DC    CL8'IKJEFF10'           MODULE NAME
HERE     STM   R14,R12,12(R13)         STORE CALLER'S REGS
START    BALR  R11,0                   CONTINUE WITH NEXT INSTRUCTION
         USING *,R11                   BASE
         L     R12,0(R1)               BASE
         USING PARMS,R12               ADDRESS TO PARMS DSECT
         DROP  R15                     DROP R15
CHECK99  CLC   WORD6,=F'99'            IF PREVIOUS CC=99 THEN
         BE    OKAY                    IF NOT, GO TO OKAY
CHECK8   CLC   WORD6,=F'8'             IF PREVIOUS CC=8 THEN
         BNE   PROCESS                 IF NOT, GO TO PROCESS
         MVC   WORD6,=F'0'             RESET WORD 6 TO ZERO
         FREEMAIN EU,LV=70,A=VSADDR    ISSUE FREEMAIN FOR MESSAGE
         LM    R14,R12,12(R13)         RESTORE REGISTERS
```

```
         LA    R15,16                  OK, SET CC=16
         BR    R14                     RETURN TO CALLER
*   PROCESS THE JOB CARD - CHECK ACCOUNTING INFORMATION.                *
PROCESS  EQU   *
         L     R3,WORD1                LOAD WORD1 ADDRESS (ACCT#)
         LA    R4,TABLE2               LOAD USER TABLE
         LA    R5,TABLEND2             LOAD END OF USER TABLE
USERID   CLC   Ø(7,R4),21(R3)          CHECK FOR VALID USER ID
*        WTO   'ACCT',ROUTCDE=2,DESC=(4)
         BE    OKAY                    IF FOUND, SET CC TO 16
         A     R4,=F'7'                INCREMENT USER TABLE BY 7
         CR    R4,R5                   CHECK IF END OF USER TABLE
         BH    NOMATCH                 IF SO, BRANCH TO NOMATCH
         B     USERID
NOMATCH  EQU   *
*        WTO   'NOMATCH',ROUTCDE=2,DESC=(4)
         GETMAIN EU,LV=7Ø,A=VSADDR     PREPARE FOR TERMINAL MESSAGE
         MVC   WORD2,VSADDR            STORE VSADDR TO WORD2
         L     R3,VSADDR               STORE VSADDR TO R3
         MVC   Ø(7Ø,R3),MSG1           MOVE MSG CONTEXT TO VS
         MVC   WORD6,=F'8'             NO MATCH, SET CC=8
         LM    R14,R12,12(R13)         RESTORE REGISTERS
         LA    R15,8                   ERROR, SET CC=8
*        MVC   Ø(R3),MSG1              THIS WILL DUMP
*        B     EXIT
         BR    R14
OKAY     EQU   *
*        WTO   'OKAY',ROUTCDE=2,DESC=(4)
         MVC   WORD6,=F'99'            MATCH, SET CC=99
         LM    R14,R12,12(R13)         RESTORE REGISTERS
         LA    R15,Ø                   OK, SET CC=16
EXIT     BR    R14                     RETURN
*   DEFINE CONSTANTS AND TABLES FOR VALIDITY CHECKS                     *
MSG1     DS    ØCL7Ø
         DC    XL2'46'                 MESSAGE LEN DEC 7Ø.
         DC    C'JOB SUBMISSION ERROR - INVALID ACC'
         DC    C'OUNT NUMBER SUPPLIED              '
VSADDR   DS    F
TABLE2   EQU   *
         DC    C'Ø165ØØ'
         DC    C'Ø167ØØ'
         DC    C'Ø167188'
         DC    C'1ØØ94ØØ'
         DC    C'1Ø1ØØØ'
         .
         .
         DC    C'8667Ø7'
         DC    C'8667Ø8'
         DC    C'8667Ø9'
         DC    C'8667188'
         DC    C'8667199'
```

```
         DC    C'8672500'
         DC    C'8681000'
         DC    C'8962400'
         DC    C'8963200'
         DC    C'8967300'
         DC    C'9647100'
TABLEND2 EQU   *-7
PARMS    DSECT
WORD1    DS    F
WORD2    DS    F
WORD3    DS    F
WORD4    DS    F
WORD5    DS    F
WORD6    DS    F
WORD7    DS    F
WORD8    DS    F
         END
```

Since the IKJEFF10 exit checks parameters in specific positions, you have to set up your JCL as below or you can modify the above exit to suit your installation.

```
//ITECS00R JOB (9480,8667100,TEC0000000),' Sample job card    ',
//         MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=ITECS00,TIME=1440
```

JCL TO RUN THE CHARGE-BACK SYSTEM

```
//ITECS00A JOB (9480,8667100,TEC0000001),
//             'D. TANG     ',TIME=1440,
//             CLASS=P,MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=ITECS00
//OUT01   OUTPUT PRMODE=PAGE,PAGEDEF=PMD1,FORMDEF=PMD1
//JOBLIB   DD DSN=SBICG.WORK.LOADLIB,DISP=SHR
//         DD DSN=SBICG.PROD.LOADLIB,DISP=SHR
//         DD DSN=PITEC.SELCOPY.LOADLIB,DISP=SHR
//         DD DSN=PITEC.EASYTREV.LOADLIB,DISP=SHR
//****************************************** STEP 01 OF 07 **
//* SELECT SMF TYPE 30, SUBTYPE 5 TERMINATION RECORD.       *
//*********************************************************
//SELTTOD  EXEC PGM=SELCOPY
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//DISK10   DD DSN=PMN00.SMFS.WEEK(0),UNIT=CART,
//            DISP=OLD
//DISKOUT  DD DSN=ITECS00.SMFS.MONTH,
//            DISP=(MOD,CATLG,DELETE),
//*           DCB=(RECFM=VB),
//            DCB=(RECFM=VB,LRECL=32752,BLKSIZE=32756),
//            UNIT=SYSDA,SPACE=(TRK,(45,45),RLSE)
//SYSIN    DD *
READ DISK10 RECFM=VB
```

```
IF POS 6 = X'1E'                    * RECORD TYPE 30
AND POS 41,44 <> X'00000000'        * NO BLANK RECORD
AND POS 23,24 =  X'0005'            * SUBTYPE RECORD 5 (TERMINATION)
AND POS 19,22 = 'JES2'              * ONLY JES2 FOR BATCH.
THEN WRITE DISKOUT
/*
//*************************************** STEP 02 OF 07 **
//* EXECUTE REXX TO EXTRACT RECORD FOR PMBAT80              *
//*************************************************************
//REXX010 EXEC PGM=IRXJCL,PARM='R@SMF1'
//SYSEXEC  DD  DSN=ITECS00.MVS.CNTL,DISP=SHR
//INPUT    DD  DSN=ITECS00.SMFS.MONTH,DISP=SHR
//SYSTSIN  DD  DUMMY
//SYSTSPRT DD  SYSOUT=X
//OUTPUT   DD  DSN=ITECS00.SMFS.OUT,DISP=(MOD,CATLG,CATLG),
//         UNIT=3390,SPACE=(TRK,(10,10),RLSE),
//         DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)
/*
//*************************************** STEP 04 OF 07 **
//* SORT RECORD BY COST CENTER, JOBNAME, MONTH, DAY         *
//*************************************************************
//MICA0108 EXEC PGM=SORT,REGION=2M
//SORTIN   DD DSN=ITECS00.SMFS.OUT,DISP=(OLD,DELETE)
//*ORTIN   DD DSN=ITECS00.SMFS.OUT,DISP=OLD
//SORTOUT  DD DSN=P$$$$.BATALL.WORK.SORTOUT,
//         DISP=(NEW,CATLG),
//         UNIT=SYSDA,SPACE=(CYL,(10,10),RLSE),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000)
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
   SORT FIELDS=(5,7,A,12,8,A,3,2,A,1,2,A),
   FORMAT=CH,DYNALLOC=(SYSDA,3)
/*
//*************************************** STEP 05 OF 07 **
//* IDCAMS TO CREATE VSAM SUMMARY FILE                      *
//*************************************************************
//MICA0109 EXEC PGM=IDCAMS,REGION=1024K
//DDNAME1  DD VOL=SER=PROD21,UNIT=3390,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DELETE ITECS00.VSAM.SUMMARY
  DEFINE CLUSTER(NAME(ITECS00.VSAM.SUMMARY) -
         VOLUME(PROD21)                                  -
         FILE(DDNAME1) -
         FSPC(10 10) -
         INDEXED -
         RECORDSIZE(61 61) -
         KEYS(7 0) -
         REUSE -
         SHR (2 3) -
         CYL(10 10)) -
```

```
         DATA(NAME(ITECSØØ.VSAM.SUMMARY.DATA)) -
         INDEX(NAME(ITECSØØ.VSAM.SUMMARY.INDEX))
/*
//*************************************** STEP Ø6 OF Ø7 *
//* PRODUCE BATCH MONTHLY REPORT -                      *
//* 2 REPORTS PRODUCED -                                *
//* OUTPUT = 2 COPIES                                   *
//*****************************************************
//MICAØ11Ø EXEC PGM=PMBAT8Ø
//BATJOBO  DD DSN=P$$$$.BATALL.WORK.SORTOUT,DISP=(OLD,DELETE)
//*ATJOBO  DD DSN=P$$$$.BATALL.WORK.SORTOUT,DISP=OLD
//SUMBATO  DD DSN=ITECSØØ.VSAM.SUMMARY,DISP=OLD
//*UTPUT   DD SYSOUT=(F,,PMD1),OUTPUT=(*.OUTØ1),COPIES=2
//OUTPUT   DD SYSOUT=(F,,PMD1),OUTPUT=(*.OUTØ1),COPIES=1
//SYSIN    DD DSN=PMICA.PDS.DAILY(DATE),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSDBOUT DD SYSOUT=*
//ABENDAID DD SYSOUT=*
//*
//*************************************** STEP Ø7 OF Ø7 *
//* PRODUCE BATCH MONTHLY REPORT -                      *
//* 2 REPORTS PRODUCED -                                *
//* OUTPUT = 2 COPIES                                   *
//*****************************************************
//PMBAT9Ø  EXEC PGM=PMBAT9Ø
//SUMBAT   DD DSN=ITECSØØ.VSAM.SUMMARY,DISP=OLD
//*UTPUT   DD SYSOUT=(F,,PMD1),OUTPUT=(*.OUTØ1),COPIES=2
//OUTPUT   DD SYSOUT=(F,,PMD1),OUTPUT=(*.OUTØ1),COPIES=1
//SYSIN    DD DSN=PMICA.PDS.DAILY(DATE),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSDBOUT DD SYSOUT=*
//ABENDAID DD SYSOUT=*
//*
//
```

REXX PROGRAM TO EXTRACT SMF TERMINATION RECORD

```
/* REXX ---------------------------------------------------------------*/
/*      Program: r@smf1                  Program Type: MAINLINE       */
/*       Author: dave tang               Create Date: feb 14/97       */
/*  Description: extra smf record type 3Ø to produce in house         */
/*               chargeback report. This program takes input from     */
/*               data extracted by selcopy program. Only subtype 5    */
/*               (termination record) is expected.                    */
/*  Environment: TSO ..............................................   */
/*        Parms: none                                                  */
/*        Logic: read the program...                                   */
/*--------------------------------------------------------------------*/
```

```
EndOfFile = 0
Counter = 0
Do While (¬EndOfFile)
  Counter = Counter + 1
  "execio 1 diskr input"
  If rc = 0 Then
    do
      parse pull Record
      call a00_process
    end
  else
     EndOfFile = 1
end
"execio 0 diskr input 0 (finis"
"execio 0 diskr output 0 (finis"
return
a00_process:
/* ---------------------------------------------------------------- */
/* FBOffset is set to 3 because this is a variable record. We have  */
/* to bypass the record descriptor block.                           */
/* ---------------------------------------------------------------- */
FBOffset = 3
SMF30IOF = 32 - FBOffset
/* ---------------------------------------------------------------- */
/* We are only interested in information contained in IOF, UOF, and */
/* COF. The information is using a triplet method.                  */
/* ---------------------------------------------------------------- */
SMF30IOF = c2d(substr(Record,SMF30IOF,4)) - FBOffset
SMF30UOF = 40 - FBOffset
SMF30UOF = c2d(substr(Record,SMF30UOF,4)) - FBOffset
SMF30COF = 56 - FBOffset
SMF30COF = c2d(substr(Record,SMF30COF,4)) - FBOffset
UserID = SMF30IOF + 108
UserID = substr(Record,UserID,8)
/* ---------------------------------------------------------------- */
/* Offset is hard coded here and is the relative offset from the    */
/* SMF variable.                                                    */
/* ---------------------------------------------------------------- */
UserID = SMF30IOF + 108
UserID = substr(Record,UserID,8)
JobName = substr(Record,SMF30IOF,8)
StartInitDate = SMF30IOF + 60
StartInitDate = substr(Record,StartInitDate,4)
StartInitDate = c2x(StartInitDate)
call B00_convert_date
ChannelEXCP = SMF30UOF + 4
ChannelEXCP = substr(Record,ChannelEXCP,4)
ChannelEXCP = c2d(ChannelEXCP)
TCB = SMF30COF + 4
TCB = substr(Record,TCB,4)
TCB = c2d(TCB)
```

```
SRB = SMF30COF+8
SRB = substr(Record,SRB,4)
SRB = c2d(SRB)
CPU = TCB + SRB
/* ---------------------------------------------------------------- */
/* Format field here and get ready to write out the record          */
/* ---------------------------------------------------------------- */
Userid = left(Userid,8)
JobName = left(Jobname,8)
ChannelEXCP = right(ChannelEXCP,8,'0')
call C00_get_CostCenter
CPU = right(CPU,8,'0')
DD  = right(DD,2,'0')
Month = right(Month,2,'0')
Rec1 = DD || Month || CostCenter || JobName || '    ' || CPU
Rec1 = Rec1 || ChannelEXCP
Queue Rec1
"execio 1 diskw output"
return
B00_convert_date:
/* ---------------------------------------------------------------- */
/* SMF30STD is stored in the Julian date format, this routine will  */
/* convert this date to MMDD.                                       */
/* ---------------------------------------------------------------- */
StartInitDate = substr(StartInitDate,3,5)
Year = substr(StartInitDate,1,2)
FullYear = 19 || Year
Days = substr(StartInitDate,3,3)
JulianMM ='031028031030031030031031030031030031'
JulianDD ='031059090120151181212243273304334365'
JulianDD2='031060091121152182213244274305335366'
LeapYear = FullYear // 4
/* ---------------------------------------------------------------- */
/* Check for leap year, if so, use JulianDD2                        */
/* ---------------------------------------------------------------- */
If LeapYear = 0 then
   JulianDD = JulianDD2
StopFlag = 'N'
Month = 0
Start = 1
do until StopFlag = 'Y'
   Month = Month + 1
   MM = substr(JulianDD,Start,3)
/* ---------------------------------------------------------------- */
/* Keep going until MM > Days. The new MM will be the Month variable*/
/* and DD will be the different Days and table value (PreviousMM)   */
/* ---------------------------------------------------------------- */
   if Days > MM then
      PreviousMM = MM
   else
      do
```

```
        DD = Days - PreviousMM
        StopFlag = 'Y'
       end
    Start = Start + 3
end
return
C00_get_CostCenter:
Name = substr(JobName,1,3)
/* ---------------------------------------------------------------- */
/* Check jobname to assgn cost center for charge back purpose       */
/* ---------------------------------------------------------------- */
select
  when Name = 'AAD' then CostCenter = '1863700'
  when Name = 'AAP' then CostCenter = '8167710'
  when Name = 'APR' then CostCenter = '8167710'
  when Name = 'AGL' then CostCenter = '8167760'
  when Name = 'ASE' then CostCenter = '4062100'
  when Name = 'CCI' then CostCenter = '1868900'
  when Name = 'CDM' then CostCenter = '1868900'
  when Name = 'CIR' then CostCenter = '1868900'
  when Name = 'CRD' then CostCenter = '1868900'
  when Name = 'CGP' then CostCenter = '1868900'
  when Name = 'DFS' then CostCenter = '8667100'
  when Name = 'DFP' then CostCenter = '8667100'
  when Name = 'ITE' then CostCenter = '8667100'
  when Name = 'IPS' then CostCenter = '8667100'
  when Name = 'IOP' then CostCenter = '8667100'
  when Name = 'MOV' then CostCenter = '8667100'
  when Name = 'MSC' then CostCenter = '5665500'
  when Name = 'MSE' then CostCenter = '4062100'
  when Name = 'MSM' then CostCenter = '8667100'
  when Name = 'SER' then CostCenter = '4062100'
  when Name = 'PCC' then CostCenter = '1868900'
  when Name = 'PMC' then CostCenter = '1868900'
  when Name = 'IIC' then CostCenter = '1868900'
  when Name = 'NAT' then CostCenter = '1868900'
  when Name = 'CME' then CostCenter = '3867000'
  when Name = 'CMP' then CostCenter = '3867000'
  when Name = 'MPL' then CostCenter = '3865500'
  when Name = 'PLM' then CostCenter = '3865500'
  when Name = 'MSD' then CostCenter = '3865500'
  when Name = 'PMA' then CostCenter = '8667100'
  when Name = 'PDB' then CostCenter = '8667100'
  when Name = 'P$S' then CostCenter = '8667100'
  when Name = 'PMN' then CostCenter = '8667100'
  when Name = 'RMM' then CostCenter = '8667100'
  when Name = 'TDB' then CostCenter = '8667100'
  when Name = 'UDB' then CostCenter = '8667100'
  when Name = 'PST' then CostCenter = '8667100'
  when Name = 'PPR' then CostCenter = '8667100'
  when Name = 'PIP' then CostCenter = '8667100'
```

```
  when Name = 'PMV' then CostCenter = '8667100'
  when Name = 'PPG' then CostCenter = '8667100'
  when Name = 'PSY' then CostCenter = '8667100'
  when Name = 'XSE' then CostCenter = '8667100'
  when substr(Jobname,1,2) = 'WO' then CostCenter = '3867000'
  when substr(Jobname,1,2) = 'MT' then CostCenter = '1868900'
otherwise
  CostCenter = '9999999'
end
return
```

PROGRAM TO PRODUCE DETAIL REPORT

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.     PMBAT80.
       AUTHOR.         DAVE TANG.
       DATE-WRITTEN.   xx-xx-xx.
       REMARKS.
           READ EXTRACTED SMF RECORD TYPE 30, SUBTYPE 5 AND CREATE A
           CHARGE BACK REPORT.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. IBM-4341.
       OBJECT-COMPUTER. IBM-4341.
       SPECIAL-NAMES. C01 IS NEW-PAGE.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
            SELECT FD-BATJOBO     ASSIGN TO SYS010-DA-FBA1-S-BATJOBO.
            SELECT FD-SUMBATO     ASSIGN TO SYS010-DA-FBA1-DA-SUMBATO
                   FILE STATUS    IS WS-SUMBATO-STATUS
                   RECORD KEY     IS FD-SUM-BATCH-COSTCTR
                   ORGANIZATION   IS INDEXED
                   ACCESS         IS DYNAMIC.
            SELECT FD-PARM-CARD   ASSIGN TO UT-S-SYSIN.
            SELECT FD-OUT-FILE    ASSIGN TO SYS030-UR-1403-S-OUTPUT.
       DATA DIVISION.
       FILE SECTION.
       FD  FD-BATJOBO
           LABEL RECORDS ARE STANDARD
           BLOCK  CONTAINS 0 RECORDS
           RECORD CONTAINS 80 CHARACTERS
           DATA RECORD IS FD-BATJOBO-REC.
       01  FD-BATJOBO-REC.
           05 FD-BATJOBO-KEY     PIC X(19).
           05 FD-BATJOBO-JESNO   PIC X(04).
           05 FD-BATCH-CPUTIME   PIC 9(08).
           05 FD-BATCH-EXCP      PIC 9(08).
           05 FILLER             PIC X(41).
       FD  FD-SUMBATO
           LABEL RECORDS ARE STANDARD
```

```
    BLOCK  CONTAINS Ø RECORDS
    DATA RECORD IS FD-SUM-BATCH-REC.
Ø1  FD-SUM-BATCH-REC.
    Ø5 FD-SUM-BATCH-COSTCTR    PIC X(Ø7).
    Ø5 FD-SUM-BATCH-CPUTIME.
       1Ø  FD-SUM-BATCH-HH     PIC 9(Ø3).
       1Ø  FILLER              PIC X(Ø1).
       1Ø  FD-SUM-BATCH-MM     PIC 9(Ø2).
       1Ø  FILLER              PIC X(Ø1).
       1Ø  FD-SUM-BATCH-SS     PIC 9(Ø2).
       1Ø  FILLER              PIC X(Ø1).
       1Ø  FD-SUM-BATCH-SS1ØØ  PIC 9(Ø2).
    Ø5 FD-SUM-BATCH-CPUTIME-CHG PIC 9(Ø9)V99.
    Ø5 FD-SUM-BATCH-EXCP       PIC 9(Ø9).
    Ø5 FD-SUM-BATCH-EXCP-CHG   PIC 9(Ø9)V99.
    Ø5 FD-SUM-BATCH-TOTAL-CHG  PIC 9(Ø9)V99.
FD  FD-PARM-CARD
    LABEL RECORDS ARE STANDARD
    BLOCK  CONTAINS Ø RECORDS
    DATA RECORD IS FD-PARM-REC.
Ø1  FD-PARM-REC.
    Ø5 FD-DATE           PIC 9(Ø2).
    Ø5 FD-YEAR           PIC 9(Ø2).
    Ø5 FILLER            PIC X(76).
FD  FD-OUT-FILE
    LABEL RECORDS ARE STANDARD
    RECORD CONTAINS 133 CHARACTERS
    DATA RECORD IS FD-OUT-REC.
Ø1  FD-OUT-REC.
    Ø5 CCC         PIC X(1).
    Ø5 FILLER      PIC X(132).
WORKING-STORAGE SECTION.
Ø1  FILLER                    PIC X(4Ø) VALUE
    'MICSP W.S STARTS HERE'.
Ø1  WS-MISC-WORK.
    Ø5  WS-BATJOBO-STATUS     PIC X(Ø2) VALUE 'ØØ'.
    Ø5  WS-SUMBATO-STATUS     PIC X(Ø2) VALUE 'ØØ'.
    Ø5  WS-EOF-FLAG           PIC X(Ø1) VALUE SPACES.
    Ø5  WS-LINE-CNT           PIC 9(Ø2) VALUE 66.
    Ø5  WS-LINE-CNT-2         PIC 9(Ø2) VALUE 66.
    Ø5  WS-SPACES.
        1Ø WS-SPACES-CC       PIC X(Ø1) VALUE SPACES.
        1Ø FILLER             PIC X(133) VALUE SPACES.
    Ø5  WS-FIRST-RECORD       PIC X(Ø1) VALUE 'Y'.
    Ø5  WS-WORK1              PIC 9(Ø9).
    Ø5  WS-WORK2              PIC 9(Ø9).
    Ø5  WS-WORK3              PIC X(12) VALUE 'ØØØ:ØØ:ØØ.ØØ'.
    Ø5  WS-WORK-CHARGE        PIC 9(Ø9)V99.
    Ø5  WS-CPU-RATE           PIC 9V9999.
    Ø5  WS-EXCP-RATE          PIC 9V9999.
    Ø5  WS-TOTAL-CPUTIME.
```

```
    10 WS-TOTAL-SS100   PIC 9(11) VALUE 0.
    10 WS-TOTAL-EXCP     PIC 9(09) VALUE 0.
    10 WS-TOTAL-CHG      PIC 9(09)V99 VALUE 0.
    10 WS-TOTAL-CPUTIME-CHG   PIC 9(09)V99 VALUE 0.
    10 WS-TOTAL-EXCP-CHG      PIC 9(09)V99 VALUE 0.
    10 WS-TOTAL1              PIC 9(09)V99 VALUE 0.
    10 WS-TOTAL2              PIC 9(09)V99 VALUE 0.
    10 WS-TOTAL3              PIC 9(09)V99 VALUE 0.
    10 WS-TOTAL4              PIC 9(09)V99 VALUE 0.
    10 WS-GTOTAL-HH      PIC 9(09) VALUE 0.
    10 WS-GTOTAL-MM      PIC 9(09) VALUE 0.
    10 WS-GTOTAL-SS      PIC 9(09) VALUE 0.
    10 WS-GTOTAL-SS100   PIC 9(11) VALUE 0.
    10 WS-GTOTAL-EXCP     PIC 9(09) VALUE 0.
    10 WS-GTOTAL-CHG      PIC 9(09)V99 VALUE 0.
05  WS-PREV-COSTCTR      PIC X(07) VALUE SPACES.
05  WS-HEAD1.
    10 WS-HEAD1-CC       PIC X(01) VALUE '1'.
    10 WS-HEAD-MONTH     PIC X(09).
    10 FILLER            PIC X(01) VALUE SPACES.
    10 WS-HEAD-YEAR      PIC X(02).
    10 FILLER            PIC X(120) VALUE SPACES.
05  WS-HEAD2.
    10 WS-HEAD2-CC       PIC X(01) VALUE '2'.
    10 WS-HEAD-COSTCTR   PIC X(07).
    10 FILLER            PIC X(125) VALUE SPACES.
05  WS-HEAD3.
    10 WS-HEAD3-CC        PIC X(01) VALUE '3'.
    10 FILLER             PIC X(14) VALUE 'IBM_BATCH_JOBS'.
    10 FILLER             PIC X(03) VALUE SPACES.
    10 FILLER             PIC X(02) VALUE 'DD'.
    10 FILLER             PIC X(01) VALUE '/'.
    10 FILLER             PIC X(02) VALUE 'MM'.
    10 FILLER             PIC X(02) VALUE SPACES.
    10 FILLER             PIC X(11) VALUE '   CPU_TIME'.
    10 FILLER             PIC X(01) VALUE SPACES.
    10 FILLER             PIC X(09) VALUE '      I/O'.
    10 FILLER             PIC X(01) VALUE SPACES.
    10 FILLER             PIC X(19) VALUE SPACES.
    10 FILLER             PIC X(03).
    10 FILLER-ACC3-HEAD   PIC X(09) VALUE 'ACCT_NO_3'.
    10 FILLER             PIC X(12).
    10 FILLER             PIC X(13) VALUE '       CHARGE'.
05  WS-HEAD4.
    10 WS-HEAD4-CC        PIC X(01) VALUE ' '.
    10 FILLER             PIC X(14) VALUE SPACES.
    10 FILLER             PIC X(03) VALUE SPACES.
    10 FILLER             PIC X(02) VALUE SPACES.
    10 FILLER             PIC X(01) VALUE SPACES.
    10 FILLER             PIC X(02) VALUE SPACES.
    10 FILLER             PIC X(01) VALUE SPACES.
```

```
    10 FILLER             PIC X(11) VALUE SPACES.
    10 FILLER             PIC X(01) VALUE SPACES.
    10 FILLER             PIC X(09) VALUE SPACES.
    10 FILLER             PIC X(02) VALUE SPACES.
    10 FILLER             PIC X(09) VALUE SPACES.
    10 FILLER             PIC X(01) VALUE SPACES.
    10 FILLER             PIC X(09) VALUE SPACES.
    10 FILLER             PIC X(27) VALUE SPACES.
05  WS-PRT-JOB.
    10 WS-PRT-CC          PIC X(01) VALUE SPACES.
    10 WS-PRT-JOBNAME     PIC X(08).
    10 FILLER             PIC X(09) VALUE SPACES.
    10 WS-PRT-DAY         PIC X(02).
    10 FILLER             PIC X(01) VALUE '/'.
    10 WS-PRT-MONTH       PIC X(02).
    10 FILLER             PIC X(02).
    10 WS-PRT-CPUTIME     PIC X(11).
    10 FILLER             PIC X(01).
    10 WS-PRT-EXCP        PIC ZZZZZZZZ9.
    10 FILLER             PIC X(01).
    10 FILLER             PIC X(09) VALUE SPACES.
    10 FILLER             PIC X(01).
    10 FILLER             PIC X(09) VALUE SPACES.
    10 FILLER             PIC X(24).
    10 WS-PRT-CHARGE      PIC $ZZZZZZZZ9.99.
05  WS-PRT-SUBTOTAL.
    10 WS-PRT-SUB-CC      PIC X(01) VALUE SPACES.
    10 WS-PRT-SUB-FILLER  PIC X(18) VALUE 'SUBTOTAL'.
    10 FILLER             PIC X(05) VALUE SPACES.
    10 WS-PRT-SUB-HH      PIC Z99.
    10 FILLER             PIC X(1) VALUE ':'.
    10 WS-PRT-SUB-MM      PIC 9(2).
    10 FILLER             PIC X(1) VALUE ':'.
    10 WS-PRT-SUB-SS      PIC 9(2).
    10 FILLER             PIC X(1) VALUE '.'.
    10 WS-PRT-SUB-SS100   PIC 9(2).
    10 FILLER             PIC X(01).
    10 WS-PRT-SUB-EXCP    PIC ZZZZZZZZ9.
    10 FILLER             PIC X(44).
    10 WS-PRT-SUB-CHARGE  PIC $ZZZZZZZZ9.99.
05  WS-BATCH-JOBO.
    07 WS-BATCH-KEY.
       10 WS-BATCH-DAY.
          15  WS-BATCH-DAY1     PIC X(01).
          15  WS-BATCH-DAY2     PIC X(01).
       10 WS-BATCH-MONTH     PIC 9(02).
       10 WS-BATCH-COSTCTR   PIC X(07).
       10 WS-BATCH-JOBNAME   PIC X(08).
    07 WS-BATCH-JESNO     PIC X(4).
    07 WS-BATCH-CPUTIME   PIC 9(08).
    07 WS-BATCH-EXCP      PIC 9(08).
```

```
           07 FILLER              PIC X(41).
       05  WS-TEM-CPU-TIME.
           10 WS-TEM-HH       PIC 9(09) VALUE 0.
           10 WS-TEM-MM       PIC 9(09) VALUE 0.
           10 WS-TEM-SS       PIC 9(09) VALUE 0.
           10 WS-TEM-SS100    PIC 9(09) VALUE 0.
       05  WS-RET-CPU-TIME.
           10 WS-RET-HH       PIC 9(09) VALUE 0.
           10 WS-RET-MM       PIC 9(09) VALUE 0.
           10 WS-RET-SS       PIC 9(09) VALUE 0.
           10 WS-RET-SS100    PIC 9(09) VALUE 0.
       05  W1-DETAIL.
           07 W1-BATCH-DAY        PIC X(02).
           07 W1-BATCH-MONTH      PIC 9(02).
           07 W1-BATCH-COSTCTR    PIC X(07).
           07 W1-BATCH-JOBNAME    PIC X(08).
           07 W1-BATCH-JESNO      PIC X(4).
           07 W1-BATCH-CPUTIME.
              09 W1-BATCH-HH      PIC 9(02) VALUE 0.
              09 FILLER           PIC X(01) VALUE ':'.
              09 W1-BATCH-MM      PIC 9(02) VALUE 0.
              09 FILLER           PIC X(01) VALUE ':'.
              09 W1-BATCH-SS      PIC 9(02) VALUE 0.
              09 FILLER           PIC X(01) VALUE ':'.
              09 W1-BATCH-SS100   PIC 9(02) VALUE 0.
           07 W1-BATCH-EXCP       PIC 9(09)  VALUE 0.
           07 W1-ACCUM-CPUTIME.
              09 W1-ACCUM-HH      PIC 9(09) VALUE 0.
              09 W1-ACCUM-MM      PIC 9(09) VALUE 0.
              09 W1-ACCUM-SS      PIC 9(09) VALUE 0.
              09 W1-ACCUM-SS100   PIC 9(09) VALUE 0.
           07 W1-PRT-CHARGE       PIC 9(09)V99  VALUE 0.
       05  WS-OK-FLAG             PIC X(01) VALUE SPACES.
       05  WS-MONTH-TAB.
           10  FILLER             PIC X(09) VALUE 'JANUARY'.
           10  FILLER             PIC X(09) VALUE 'FEBRUARY'.
           10  FILLER             PIC X(09) VALUE 'MARCH'.
           10  FILLER             PIC X(09) VALUE 'APRIL'.
           10  FILLER             PIC X(09) VALUE 'MAY'.
           10  FILLER             PIC X(09) VALUE 'JUNE'.
           10  FILLER             PIC X(09) VALUE 'JULY'.
           10  FILLER             PIC X(09) VALUE 'AUGUST'.
           10  FILLER             PIC X(09) VALUE 'SEPTEMBER'.
           10  FILLER             PIC X(09) VALUE 'OCTOBER'.
           10  FILLER             PIC X(09) VALUE 'NOVEMBER'.
           10  FILLER             PIC X(09) VALUE 'DECEMBER'.
       05  WS-MONTH-TABLE REDEFINES WS-MONTH-TAB
                                  PIC X(09) OCCURS 12.
       05  WS-PREV-KEY            PIC X(19).
       05  WS-PREV-JESNO          PIC X(04).
   01  FILLER                     PIC X(40) VALUE
```

```
     'MICSP WORKING STORAGE ENDED '.
 PROCEDURE DIVISION.
 0000-MAINLINE.
     OPEN INPUT  FD-BATJOBO, FD-PARM-CARD
          OUTPUT FD-SUMBATO,
                 FD-OUT-FILE.
     READ FD-PARM-CARD
          AT END DISPLAY 'CONTROL CARD MISSING'
          STOP RUN.
     MOVE WS-MONTH-TABLE(FD-DATE) TO WS-HEAD-MONTH.
     MOVE FD-YEAR                 TO WS-HEAD-YEAR.
* ------------------------------------------------------------ *
* SET CPU AND IO CHARGE RATE HERE.                             *
* $4382.39 PER CPU HOUR, I/O RATE $2 PER 1000.                 *
* ------------------------------------------------------------ *
     MOVE 1.2173                  TO WS-CPU-RATE.
     COMPUTE WS-EXCP-RATE = 2 / 1000.
     PERFORM 1000-PRT-BATJOB THRU 1000-EXIT
             UNTIL WS-EOF-FLAG = 'Y'.
     PERFORM 1007-PRINT-DETAIL.
     PERFORM 1050-SUB-TOTAL.
     PERFORM 9000-GRAND-TOTAL.
     CLOSE FD-BATJOBO, FD-SUMBATO, FD-OUT-FILE, FD-PARM-CARD.
     STOP RUN.
  1000-PRT-BATJOB.
     READ FD-BATJOBO INTO WS-BATCH-JOBO
          AT END MOVE 'Y' TO WS-EOF-FLAG
                 GO TO 1000-EXIT.
     IF WS-FIRST-RECORD = 'Y'
        MOVE WS-BATCH-COSTCTR TO WS-PREV-COSTCTR, WS-HEAD-COSTCTR
        MOVE WS-BATCH-KEY     TO WS-PREV-KEY
        MOVE WS-BATCH-JESNO   TO WS-PREV-JESNO
        MOVE 'N'              TO WS-FIRST-RECORD.
* ------------------------------------------------------------ *
* IF THERE IS A CHANGE IN THE COST CENTRE, PRINT SUBTOTAL      *
* ------------------------------------------------------------ *
     IF WS-BATCH-COSTCTR NOT = WS-PREV-COSTCTR
        PERFORM 1007-PRINT-DETAIL
        PERFORM 1050-SUB-TOTAL
        MOVE WS-BATCH-COSTCTR TO WS-PREV-COSTCTR, WS-HEAD-COSTCTR
        MOVE WS-BATCH-KEY     TO WS-PREV-KEY
        MOVE WS-BATCH-JESNO   TO WS-PREV-JESNO
        PERFORM 1002-HEADING-1.
* ------------------------------------------------------------ *
* IF THIS IS A DIFFERENT JOB, PRINT THE DETAIL LINE            *
* ------------------------------------------------------------ *
     IF WS-BATCH-KEY NOT = WS-PREV-KEY
        PERFORM 1007-PRINT-DETAIL
        MOVE WS-BATCH-KEY     TO WS-PREV-KEY.
* ------------------------------------------------------------ *
* MULTIPLE JOBS WITH THE SAME JOB NAME WILL BE MERGED INTO ONE  *
```

```
* ENTRY IF THEY RUN ON THE SAME DAY.                            *
* -------------------------------------------------------------- *
     MOVE WS-BATCH-DAY        TO W1-BATCH-DAY.
     MOVE WS-BATCH-MONTH      TO W1-BATCH-MONTH.
     MOVE WS-BATCH-JOBNAME    TO W1-BATCH-JOBNAME.
* -------------------------------------------------------------- *
* COMPUTING VARIABLE TO STORE ALL CHARGES.                       *
* -------------------------------------------------------------- *
     COMPUTE W1-BATCH-EXCP = W1-BATCH-EXCP + WS-BATCH-EXCP.
     COMPUTE W1-ACCUM-SS100 = W1-ACCUM-SS100 + WS-BATCH-CPUTIME.
     COMPUTE WS-TOTAL1 = WS-BATCH-CPUTIME * WS-CPU-RATE / 100.
     COMPUTE WS-TOTAL2 = WS-BATCH-EXCP * WS-EXCP-RATE.
     COMPUTE WS-WORK-CHARGE = WS-TOTAL1 + WS-TOTAL2.
     COMPUTE W1-PRT-CHARGE = W1-PRT-CHARGE + WS-WORK-CHARGE.
     COMPUTE WS-TOTAL-CPUTIME-CHG = WS-TOTAL-CPUTIME-CHG +
                                    WS-TOTAL1.
     COMPUTE WS-TOTAL-EXCP-CHG    = WS-TOTAL-EXCP-CHG +
                                    WS-TOTAL2.
     COMPUTE WS-TOTAL-CHG = WS-TOTAL-CHG + WS-WORK-CHARGE.
     COMPUTE WS-TOTAL-EXCP   = WS-TOTAL-EXCP + WS-BATCH-EXCP.
 1000-EXIT.
     EXIT.
 1007-PRINT-DETAIL.
* -------------------------------------------------------------- *
* PRINT DETAIL INFORMATION FOR THE SAME JOB.                     *
* -------------------------------------------------------------- *
     MOVE ZERO                TO WS-TEM-HH, WS-TEM-MM, WS-TEM-SS.
     MOVE W1-ACCUM-SS100      TO WS-TEM-SS100.
     COMPUTE WS-TOTAL-SS100  = WS-TOTAL-SS100 + W1-ACCUM-SS100.
     PERFORM 1004-REFORMAT-CPU.
     MOVE WS-RET-HH           TO W1-BATCH-HH.
     MOVE WS-RET-MM           TO W1-BATCH-MM.
     MOVE WS-RET-SS           TO W1-BATCH-SS.
     MOVE WS-RET-SS100        TO W1-BATCH-SS100.
     MOVE W1-BATCH-DAY        TO WS-PRT-DAY.
     MOVE W1-BATCH-MONTH      TO WS-PRT-MONTH.
     MOVE W1-BATCH-JOBNAME    TO WS-PRT-JOBNAME.
     MOVE W1-BATCH-CPUTIME    TO WS-PRT-CPUTIME.
     MOVE W1-BATCH-EXCP       TO WS-PRT-EXCP.
     MOVE W1-PRT-CHARGE       TO WS-PRT-CHARGE.
     IF WS-LINE-CNT > 46
        PERFORM 1002-HEADING-1.
     IF WS-LINE-CNT = 0
        MOVE '4'              TO WS-PRT-CC
     ELSE
        MOVE ' '              TO WS-PRT-CC.
     WRITE FD-OUT-REC FROM WS-PRT-JOB
           AFTER POSITIONING WS-PRT-CC.
     ADD 1                    TO WS-LINE-CNT.
     MOVE SPACES              TO W1-BATCH-DAY.
     MOVE 0                   TO W1-BATCH-MONTH.
```

```
           MOVE SPACES              TO W1-BATCH-JOBNAME.
           MOVE Ø                   TO W1-BATCH-HH
                                       W1-BATCH-MM
                                       W1-BATCH-SS
                                       W1-BATCH-SS1ØØ.
           MOVE Ø                   TO W1-BATCH-EXCP.
           MOVE Ø                   TO W1-ACCUM-HH
                                       W1-ACCUM-MM
                                       W1-ACCUM-SS
                                       W1-ACCUM-SS1ØØ
                                       W1-PRT-CHARGE.
       1ØØ7-EXIT.
           EXIT.
       1ØØ2-HEADING-1.
           MOVE ZERO                TO WS-LINE-CNT.
           MOVE SPACES              TO FILLER-ACC3-HEAD.
           WRITE FD-OUT-REC FROM WS-HEAD1 AFTER POSITIONING WS-HEAD1-CC.
           WRITE FD-OUT-REC FROM WS-HEAD2 AFTER POSITIONING WS-HEAD2-CC.
           WRITE FD-OUT-REC FROM WS-HEAD3 AFTER POSITIONING WS-HEAD3-CC.
           WRITE FD-OUT-REC FROM WS-HEAD4 AFTER POSITIONING WS-HEAD4-CC.
       1ØØ2-EXIT-1.
           EXIT.
      * ------------------------------------------------------------ *
      * CPU EXTRACTED FROM SMF RECORD ARE STORED IN 1ØØTH OF A SECOND *
      * THIS ROUTINE WILL CONVERT IT TO HH:MM:SS:1ØØ FORMAT. THIS     *
      * ROUTINE ALSO REFORMATS THE ACCUMULATED CPU.                   *
      * ------------------------------------------------------------ *
       1ØØ4-REFORMAT-CPU.
      * ------------------------------------------------------------ *
      * IF 1ØØTH SECOND IS > 99, THEN CONVERT IT TO SECONDS.          *
      * ------------------------------------------------------------ *
           IF WS-TEM-SS1ØØ > 99
              DIVIDE WS-TEM-SS1ØØ BY 1ØØ GIVING WS-WORK1
                     REMAINDER WS-WORK2
              MOVE WS-WORK2         TO WS-RET-SS1ØØ
            ELSE
              MOVE Ø                TO WS-WORK1
              MOVE WS-TEM-SS1ØØ     TO WS-RET-SS1ØØ.
            COMPUTE WS-TEM-SS = WS-TEM-SS + WS-WORK1.
      * ------------------------------------------------------------ *
      * IF SECONDS IS > 59 THEN CONVERT IT TO MINUTES.                *
      * ------------------------------------------------------------ *
            IF WS-TEM-SS > 59
              DIVIDE WS-TEM-SS BY 6Ø GIVING WS-WORK1
                     REMAINDER WS-WORK2
              MOVE WS-WORK2         TO WS-RET-SS
            ELSE
              MOVE Ø                TO WS-WORK1
              MOVE WS-TEM-SS        TO WS-RET-SS.
            COMPUTE WS-TEM-MM = WS-TEM-MM + WS-WORK1.
```

```
* ------------------------------------------------------------ *
* IF MINUTE IS > 59 THEN CONVERT IT TO HOURS.                  *
* ------------------------------------------------------------ *
      IF WS-TEM-MM > 59
         DIVIDE WS-TEM-MM BY 60 GIVING WS-WORK1
                REMAINDER WS-WORK2
         MOVE WS-WORK2        TO WS-RET-MM
      ELSE
         MOVE 0               TO WS-WORK1
         MOVE WS-TEM-MM       TO WS-RET-MM.
      COMPUTE WS-TEM-HH = WS-TEM-HH + WS-WORK1.
      MOVE WS-TEM-HH          TO WS-RET-HH.
 1004-EXIT.
     EXIT.
* ------------------------------------------------------------ *
* PRINT SUB-TOTAL AND ACCUMULATE GRAND TOTAL.                  *
* ------------------------------------------------------------ *
 1050-SUB-TOTAL.
     MOVE ZERO               TO WS-TEM-HH, WS-TEM-MM, WS-TEM-SS.
     EXHIBIT NAMED WS-TOTAL-SS100.
     MOVE WS-TOTAL-SS100     TO WS-TEM-SS100.
* ------------------------------------------------------------ *
* ACCUMULATE GRAND TOTALS.
* ------------------------------------------------------------ *
     ADD WS-TOTAL-SS100      TO WS-GTOTAL-SS100.
     PERFORM 1004-REFORMAT-CPU.
     MOVE WS-RET-HH          TO FD-SUM-BATCH-HH,
                                WS-PRT-SUB-HH.
     MOVE WS-RET-MM          TO FD-SUM-BATCH-MM,
                                WS-PRT-SUB-MM.
     MOVE WS-RET-SS          TO FD-SUM-BATCH-SS,
                                WS-PRT-SUB-SS.
     MOVE WS-RET-SS100       TO FD-SUM-BATCH-SS100,
                                WS-PRT-SUB-SS100.
     MOVE WS-PREV-COSTCTR    TO FD-SUM-BATCH-COSTCTR.
     MOVE WS-TOTAL-EXCP      TO WS-PRT-SUB-EXCP,
                                FD-SUM-BATCH-EXCP.
     MOVE WS-TOTAL-CPUTIME-CHG TO FD-SUM-BATCH-CPUTIME-CHG.
     MOVE WS-TOTAL-EXCP-CHG    TO FD-SUM-BATCH-EXCP-CHG.
     MOVE WS-TOTAL-CHG         TO WS-PRT-SUB-CHARGE,
                                  FD-SUM-BATCH-TOTAL-CHG.
     WRITE FD-OUT-REC FROM WS-SPACES AFTER POSITIONING
                           WS-SPACES-CC.
     WRITE FD-OUT-REC FROM WS-PRT-SUBTOTAL AFTER POSITIONING
                           WS-PRT-SUB-CC.
     ADD WS-TOTAL-EXCP       TO WS-GTOTAL-EXCP.
     ADD WS-TOTAL-CHG        TO WS-GTOTAL-CHG.
* ------------------------------------------------------------ *
* CREATE SUMMARY RECORD FOR PROGRAM PMBAT90 TO PRINT TOTAL BY  *
* COST CENTRE.                                                 *
* ------------------------------------------------------------ *
```

```
           WRITE FD-SUM-BATCH-REC.
           IF WS-SUMBATO-STATUS NOT = '00'
              DISPLAY 'ERROR ON WRITE SUMMARY, JOB TERMINATED, RC= '
                      WS-SUMBATO-STATUS FD-SUM-BATCH-REC
              DISPLAY 'ERROR ON WRITE SUMMARY, JOB TERMINATED, RC= '
                      WS-SUMBATO-STATUS FD-SUM-BATCH-REC UPON CONSOLE
              CLOSE FD-BATJOBO, FD-SUMBATO, FD-OUT-FILE, FD-PARM-CARD
              STOP RUN.
           MOVE 0                TO WS-TOTAL-EXCP,
                                    WS-TOTAL-CHG,
                                    WS-TOTAL-SS100,
                                    WS-TOTAL-CPUTIME-CHG,
                                    WS-TOTAL-EXCP-CHG.
           ADD 2                 TO WS-LINE-CNT.
           ADD 2                 TO WS-LINE-CNT-2.
       1005-EXIT.
           EXIT.
      * ------------------------------------------------------------ *
      * SET UP GRAND TOTAL                                           *
      * ------------------------------------------------------------ *
       9000-GRAND-TOTAL.
           MOVE 'TOTAL'             TO WS-HEAD-COSTCTR.
           PERFORM 1002-HEADING-1.
           MOVE ZERO                TO WS-TEM-HH, WS-TEM-MM, WS-TEM-SS.
           MOVE WS-GTOTAL-SS100     TO WS-TEM-SS100.
           PERFORM 1004-REFORMAT-CPU.
           MOVE WS-RET-HH           TO WS-PRT-SUB-HH.
           MOVE WS-RET-MM           TO WS-PRT-SUB-MM.
           MOVE WS-RET-SS           TO WS-PRT-SUB-SS.
           MOVE WS-RET-SS100        TO WS-PRT-SUB-SS100.
           MOVE 'GRAND TOTAL'       TO WS-PRT-SUB-FILLER.
           MOVE WS-GTOTAL-EXCP      TO WS-PRT-SUB-EXCP.
           MOVE WS-GTOTAL-CHG       TO WS-PRT-SUB-CHARGE.
           WRITE FD-OUT-REC FROM WS-SPACES AFTER POSITIONING
                                 WS-SPACES-CC.
           WRITE FD-OUT-REC FROM WS-PRT-SUBTOTAL AFTER POSITIONING
                                 WS-PRT-SUB-CC.
       9000-EXIT.
           EXIT.
```

PROGRAM TO PRODUCE SUMMARY REPORT

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.     PMBAT90.
       AUTHOR.         DAVE TANG.
       DATE-WRITTEN.   xx-xx-xx.
       REMARKS.
           READ SUMMARY RECORD FOR EACH COST CENTRE.
       ENVIRONMENT DIVISION.
```

```
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-4341.
OBJECT-COMPUTER. IBM-4341.
SPECIAL-NAMES. C01 IS NEW-PAGE.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
     SELECT FD-SUMBAT      ASSIGN TO SYS010-DA-FBA1-DA-SUMBAT
            FILE STATUS    IS WS-SUMBAT-STATUS
            RECORD KEY     IS FD-SUM-BATCH-COSTCTR
            ORGANIZATION   IS INDEXED
            ACCESS         IS SEQUENTIAL.
     SELECT FD-PARM-CARD   ASSIGN TO UT-S-SYSIN.
     SELECT FD-OUT-FILE    ASSIGN TO SYS030-UR-1403-S-OUTPUT.
DATA DIVISION.
FILE SECTION.
FD  FD-SUMBAT
    LABEL RECORDS ARE STANDARD
    BLOCK  CONTAINS 0 RECORDS
    DATA RECORD IS FD-SUM-BATCH-REC.
01  FD-SUM-BATCH-REC.
    05 FD-SUM-BATCH-COSTCTR    PIC X(07).
    05 FD-SUM-BATCH-CPU.
       10  FD-SUM-BATCH-HH     PIC 9(03).
       10  FILLER              PIC X(01).
       10  FD-SUM-BATCH-MM     PIC 9(02).
       10  FILLER              PIC X(01).
       10  FD-SUM-BATCH-SS     PIC 9(02).
       10  FILLER              PIC X(01).
       10  FD-SUM-BATCH-SS100  PIC 9(02).
    05 FD-SUM-BATCH-CPU-CHG    PIC 9(09)V99.
    05 FD-SUM-BATCH-EXCP       PIC 9(09).
    05 FD-SUM-BATCH-EXCP-CHG   PIC 9(09)V99.
    05 FD-SUM-BATCH-TOTAL-CHG  PIC 9(09)V99.
FD  FD-PARM-CARD
    LABEL RECORDS ARE STANDARD
    BLOCK  CONTAINS 0 RECORDS
    DATA RECORD IS FD-PARM-REC.
01  FD-PARM-REC.
    05 FD-DATE           PIC 9(02).
    05 FD-YEAR           PIC 9(02).
    05 FILLER            PIC X(76).
FD  FD-OUT-FILE
    LABEL RECORDS ARE STANDARD
    RECORD CONTAINS 133 CHARACTERS
    DATA RECORD IS FD-OUT-REC.
01  FD-OUT-REC.
    05 CCC         PIC X(1).
    05 FILLER      PIC X(132).
WORKING-STORAGE SECTION.
01  FILLER                   PIC X(40) VALUE
```

```
    'MICSP W.S STARTS HERE'.
01  WS-MISC-WORK.
    05  WS-SUMBAT-STATUS     PIC X(02) VALUE '00'.
    05  WS-EOF-FLAG          PIC X(01) VALUE SPACES.
    05  WS-LINE-CNT          PIC 9(02) VALUE 66.
    05  WS-WORK1             PIC 9(09).
    05  WS-WORK2             PIC 9(09).
    05  WS-SPACES.
        10 WS-SPACES-CC      PIC X(01) VALUE SPACES.
        10 FILLER            PIC X(133) VALUE SPACES.
    05  WS-HEAD1.
        10 WS-HEAD1-CC       PIC X(01) VALUE '1'.
        10 WS-HEAD-MONTH     PIC X(09).
        10 FILLER            PIC X(01) VALUE SPACES.
        10 WS-HEAD-YEAR      PIC X(02).
        10 FILLER            PIC X(120) VALUE SPACES.
    05  WS-HEAD2.
        10 WS-HEAD2-CC       PIC X(01) VALUE '2'.
        10 WS-HEAD-COSTCTR   PIC X(07).
        10 FILLER            PIC X(125) VALUE SPACES.
    05  WS-HEAD3.
        10 WS-HEAD3-CC        PIC X(01) VALUE '3'.
        10 FILLER             PIC X(14) VALUE 'COST_CENTER   '.
        10 FILLER             PIC X(03) VALUE SPACES.
        10 FILLER             PIC X(05) VALUE '  '.
        10 FILLER             PIC X(02) VALUE SPACES.
        10 FILLER             PIC X(11) VALUE '   CPU_TIME'.
        10 FILLER             PIC X(01) VALUE SPACES.
        10 FILLER             PIC X(09) VALUE '      I/O'.
        10 FILLER             PIC X(01) VALUE SPACES.
        10 FILLER             PIC X(18) VALUE SPACES.
        10 FILLER             PIC X(13) VALUE '   CPU_CHG'.
        10 FILLER             PIC X(1)  VALUE ' '.
        10 FILLER             PIC X(13) VALUE '  EXCP_CHG'.
        10 FILLER             PIC X(1)  VALUE ' '.
        10 FILLER             PIC X(10) VALUE ' TOTAL_CHG'.
    05  WS-HEAD4.
        10 WS-HEAD4-CC        PIC X(01) VALUE ' '.
        10 FILLER             PIC X(132) VALUE SPACES.
    05  WS-PRT-JOB.
        10 WS-PRT-CC          PIC X(01) VALUE SPACES.
        10 WS-PRT-COSTCTR     PIC X(08).
        10 FILLER             PIC X(09) VALUE SPACES.
        10 FILLER             PIC X(05).
        10 FILLER             PIC X(02).
        10 WS-PRT-CPU.
           15  WS-PRT-HH      PIC Z99.
           15  FILLER         PIC X(01) VALUE ':'.
           15  WS-PRT-MM      PIC 9(02).
           15  FILLER         PIC X(01) VALUE ':'.
```

```
           15  WS-PRT-SS       PIC 9(Ø2).
           15  FILLER          PIC X(Ø1) VALUE ':'.
           15  WS-PRT-SS1ØØ    PIC 9(Ø2).
       1Ø FILLER               PIC X(Ø1).
       1Ø WS-PRT-EXCP          PIC ZZZZZZZZ9.
       1Ø FILLER               PIC X(15) VALUE SPACES.
       1Ø WS-PRT-CPU-CHG       PIC $ZZZZZZZZ9.99.
       1Ø FILLER               PIC X(1) VALUE SPACES.
       1Ø WS-PRT-EXCP-CHG      PIC $ZZZZZZZZ9.99.
       1Ø FILLER               PIC X(Ø1) VALUE SPACES.
       1Ø WS-PRT-CHG           PIC $ZZZZZZZZ9.99.
Ø5     WS-PRT-SUBTOTAL.
       1Ø WS-PRT-SUB-CC        PIC X(Ø1) VALUE SPACES.
       1Ø FILLER               PIC X(18) VALUE '  TOTAL '.
       1Ø FILLER               PIC X(Ø6) VALUE SPACES.
       1Ø WS-PRT-SUB-HH        PIC Z99.
       1Ø FILLER               PIC X(1) VALUE ':'.
       1Ø WS-PRT-SUB-MM        PIC 9(2).
       1Ø FILLER               PIC X(1) VALUE ':'.
       1Ø WS-PRT-SUB-SS        PIC 9(2).
       1Ø FILLER               PIC X(1) VALUE '.'.
       1Ø WS-PRT-SUB-SS1ØØ     PIC 9(2).
       1Ø FILLER               PIC X(1) VALUE SPACES.
       1Ø WS-PRT-SUB-EXCP      PIC ZZZZZZZZ9.
       1Ø FILLER               PIC X(15) VALUE SPACES.
       1Ø WS-PRT-SUB-CPU-CHG PIC $ZZZZZZZZ9.99.
       1Ø FILLER               PIC X(1) VALUE SPACES.
       1Ø WS-PRT-SUB-EXCP-CHG PIC $ZZZZZZZZ9.99.
       1Ø FILLER               PIC X(Ø1) VALUE SPACES.
       1Ø WS-PRT-SUB-CHG       PIC $ZZZZZZZZ9.99.
Ø5     WS-TEM-CPU-TIME.
       1Ø WS-TEM-HH            PIC 9(Ø9) VALUE Ø.
       1Ø WS-TEM-MM            PIC 9(Ø9) VALUE Ø.
       1Ø WS-TEM-SS            PIC 9(Ø9) VALUE Ø.
       1Ø WS-TEM-SS1ØØ         PIC 9(Ø9) VALUE Ø.
Ø5     WS-RET-CPU-TIME.
       1Ø WS-RET-HH            PIC 9(Ø9) VALUE Ø.
       1Ø WS-RET-MM            PIC 9(Ø9) VALUE Ø.
       1Ø WS-RET-SS            PIC 9(Ø9) VALUE Ø.
       1Ø WS-RET-SS1ØØ         PIC 9(Ø9) VALUE Ø.
Ø5     WS-TOTAL-CPU-TIME.
       1Ø WS-TOTAL-HH          PIC 9(Ø9) VALUE Ø.
       1Ø WS-TOTAL-MM          PIC 9(Ø9) VALUE Ø.
       1Ø WS-TOTAL-SS          PIC 9(Ø9) VALUE Ø.
       1Ø WS-TOTAL-SS1ØØ       PIC 9(Ø9) VALUE Ø.
Ø5     WS-TOTAL-CPU            PIC 9(Ø9) VALUE Ø.
Ø5     WS-TOTAL-CPU-CHG        PIC 9(Ø9) VALUE Ø.
Ø5     WS-TOTAL-EXCP           PIC 9(Ø9) VALUE Ø.
Ø5     WS-TOTAL-EXCP-CHG       PIC 9(Ø9) VALUE Ø.
Ø5     WS-TOTAL-TOTAL-CHG      PIC 9(Ø9) VALUE Ø.
```

```
    Ø5  WS-OK-FLAG            PIC X(Ø1) VALUE SPACES.
    Ø5  WS-MONTH-TAB.
        1Ø  FILLER            PIC X(Ø9) VALUE 'JANUARY'.
        1Ø  FILLER            PIC X(Ø9) VALUE 'FEBRUARY'.
        1Ø  FILLER            PIC X(Ø9) VALUE 'MARCH'.
        1Ø  FILLER            PIC X(Ø9) VALUE 'APRIL'.
        1Ø  FILLER            PIC X(Ø9) VALUE 'MAY'.
        1Ø  FILLER            PIC X(Ø9) VALUE 'JUNE'.
        1Ø  FILLER            PIC X(Ø9) VALUE 'JULY'.
        1Ø  FILLER            PIC X(Ø9) VALUE 'AUGUST'.
        1Ø  FILLER            PIC X(Ø9) VALUE 'SEPTEMBER'.
        1Ø  FILLER            PIC X(Ø9) VALUE 'OCTOBER'.
        1Ø  FILLER            PIC X(Ø9) VALUE 'NOVEMBER'.
        1Ø  FILLER            PIC X(Ø9) VALUE 'DECEMBER'.
    Ø5  WS-MONTH-TABLE REDEFINES WS-MONTH-TAB
                              PIC X(Ø9) OCCURS 12.
Ø1  WS-BATCH-REC.
    Ø5 WS-BATCH-COSTCTR        PIC X(Ø7).
    Ø5 WS-BATCH-CPU.
       1Ø  WS-BATCH-HH         PIC 9(Ø3).
       1Ø  FILLER              PIC X(Ø1).
       1Ø  WS-BATCH-MM         PIC 9(Ø2).
       1Ø  FILLER              PIC X(Ø1).
       1Ø  WS-BATCH-SS         PIC 9(Ø2).
       1Ø  FILLER              PIC X(Ø1).
       1Ø  WS-BATCH-SS1ØØ      PIC 9(Ø2).
    Ø5 WS-BATCH-CPU-CHG PIC 9(Ø9)V99.
    Ø5 WS-BATCH-EXCP           PIC 9(Ø9).
    Ø5 WS-BATCH-EXCP-CHG       PIC 9(Ø9)V99.
    Ø5 WS-BATCH-TOTAL-CHG      PIC 9(Ø9)V99.
Ø1  FILLER                     PIC X(4Ø) VALUE
    'MICSP WORKING STORAGE ENDED '.
PROCEDURE DIVISION.
ØØØØ-MAINLINE.
    OPEN INPUT  FD-SUMBAT, FD-PARM-CARD
         OUTPUT FD-OUT-FILE.
    READ FD-PARM-CARD
         AT END DISPLAY 'CONTROL CARD MISSING'
         STOP RUN.
    MOVE WS-MONTH-TABLE(FD-DATE) TO WS-HEAD-MONTH.
    MOVE FD-YEAR                 TO WS-HEAD-YEAR.
    PERFORM 1ØØØ-PRT-SUMMARY THRU 1ØØØ-EXIT
            UNTIL WS-EOF-FLAG = 'Y'.
    PERFORM 2ØØØ-TOTAL.
    CLOSE FD-SUMBAT, FD-OUT-FILE, FD-PARM-CARD.
    STOP RUN.
 1ØØØ-PRT-SUMMARY.
    READ FD-SUMBAT INTO WS-BATCH-REC
         AT END MOVE 'Y' TO WS-EOF-FLAG
                GO TO 1ØØØ-EXIT.
```

```
           MOVE WS-BATCH-COSTCTR      TO WS-PRT-COSTCTR.
           MOVE WS-BATCH-HH           TO WS-PRT-HH.
           MOVE WS-BATCH-MM           TO WS-PRT-MM.
           MOVE WS-BATCH-SS           TO WS-PRT-SS.
           MOVE WS-BATCH-SS100        TO WS-PRT-SS100.
           MOVE WS-BATCH-CPU-CHG      TO WS-PRT-CPU-CHG.
           MOVE WS-BATCH-EXCP         TO WS-PRT-EXCP.
           MOVE WS-BATCH-EXCP-CHG     TO WS-PRT-EXCP-CHG.
           MOVE WS-BATCH-TOTAL-CHG    TO WS-PRT-CHG.
           COMPUTE WS-TOTAL-HH = WS-TOTAL-HH + WS-BATCH-HH.
           COMPUTE WS-TOTAL-MM = WS-TOTAL-MM + WS-BATCH-MM.
           COMPUTE WS-TOTAL-SS = WS-TOTAL-SS + WS-BATCH-SS.
           COMPUTE WS-TOTAL-SS100 = WS-TOTAL-SS100 +
                                    WS-BATCH-SS100.
           COMPUTE WS-TOTAL-SS100 = WS-TOTAL-SS100 +
                                    WS-BATCH-SS100.
           COMPUTE WS-TOTAL-CPU-CHG = WS-BATCH-CPU-CHG +
                                        WS-TOTAL-CPU-CHG
           COMPUTE WS-TOTAL-EXCP = WS-BATCH-EXCP + WS-TOTAL-EXCP.
           COMPUTE WS-TOTAL-EXCP-CHG = WS-BATCH-EXCP-CHG +
                                       WS-TOTAL-EXCP-CHG.
           COMPUTE WS-TOTAL-TOTAL-CHG = WS-BATCH-TOTAL-CHG +
                                        WS-TOTAL-TOTAL-CHG.
           IF WS-LINE-CNT > 46
              PERFORM 1002-HEADING-1.
           IF WS-LINE-CNT = 0
              MOVE '4'                TO WS-PRT-CC
           ELSE
              MOVE ' '                TO WS-PRT-CC.
           WRITE FD-OUT-REC FROM WS-PRT-JOB
                 AFTER POSITIONING WS-PRT-CC.
           ADD 1                      TO WS-LINE-CNT.
       1000-EXIT.
           EXIT.
       1002-HEADING-1.
           MOVE ZERO                  TO WS-LINE-CNT.
           WRITE FD-OUT-REC FROM WS-HEAD1 AFTER POSITIONING WS-HEAD1-CC.
           WRITE FD-OUT-REC FROM WS-HEAD2 AFTER POSITIONING WS-HEAD2-CC.
           WRITE FD-OUT-REC FROM WS-HEAD3 AFTER POSITIONING WS-HEAD3-CC.
           WRITE FD-OUT-REC FROM WS-HEAD4 AFTER POSITIONING WS-HEAD4-CC.
       1002-EXIT-1.
           EXIT.
       1004-REFORMAT-CPU.
      * ------------------------------------------------------------ *
      * IF 100TH SECOND IS > 99, THEN CONVERT IT TO SECONDS.         *
      * ------------------------------------------------------------ *
           IF WS-TEM-SS100 > 99
               DIVIDE WS-TEM-SS100 BY 100 GIVING WS-WORK1
                      REMAINDER WS-WORK2
               MOVE WS-WORK2          TO WS-RET-SS100
```

```
      ELSE
         MOVE Ø               TO WS-WORK1
         MOVE WS-TEM-SS1ØØ    TO WS-RET-SS1ØØ.
      COMPUTE WS-TEM-SS = WS-TEM-SS + WS-WORK1.
* ------------------------------------------------------------- *
* IF SECONDS IS > 59 THEN CONVERT IT TO MINUTES.                *
* ------------------------------------------------------------- *
      IF WS-TEM-SS > 59
         DIVIDE WS-TEM-SS BY 6Ø GIVING WS-WORK1
                REMAINDER WS-WORK2
         MOVE WS-WORK2        TO WS-RET-SS
      ELSE
         MOVE Ø               TO WS-WORK1
         MOVE WS-TEM-SS       TO WS-RET-SS.
      COMPUTE WS-TEM-MM = WS-TEM-MM + WS-WORK1.
* ------------------------------------------------------------- *
* IF MINUTE IS > 59 THEN CONVERT IT TO HOURS.                   *
* ------------------------------------------------------------- *
      IF WS-TEM-MM > 59
         DIVIDE WS-TEM-MM BY 6Ø GIVING WS-WORK1
                REMAINDER WS-WORK2
         MOVE WS-WORK2        TO WS-RET-MM
      ELSE
         MOVE Ø               TO WS-WORK1
         MOVE WS-TEM-MM       TO WS-RET-MM.
      COMPUTE WS-TEM-HH = WS-TEM-HH + WS-WORK1.
      MOVE WS-TEM-HH          TO WS-RET-HH.
 1ØØ4-EXIT.
     EXIT.
* ------------------------------------------------------------- *
* PRINT SUB-TOTAL AND ACCUMULATE GRAND TOTAL.                   *
* ------------------------------------------------------------- *
 2ØØØ-TOTAL.
     MOVE WS-TOTAL-HH         TO WS-TEM-HH.
     MOVE WS-TOTAL-MM         TO WS-TEM-MM.
     MOVE WS-TOTAL-SS         TO WS-TEM-SS.
     MOVE WS-TOTAL-SS1ØØ      TO WS-TEM-SS1ØØ.
     PERFORM 1ØØ4-REFORMAT-CPU.
     MOVE WS-RET-HH           TO WS-TOTAL-HH.
     MOVE WS-RET-MM           TO WS-TOTAL-MM.
     MOVE WS-RET-SS           TO WS-TOTAL-SS.
     MOVE WS-RET-SS1ØØ        TO WS-TOTAL-SS1ØØ.
     MOVE WS-TOTAL-HH         TO WS-PRT-SUB-HH.
     MOVE WS-TOTAL-MM         TO WS-PRT-SUB-MM.
     MOVE WS-TOTAL-SS         TO WS-PRT-SUB-SS.
     MOVE WS-TOTAL-SS1ØØ      TO WS-PRT-SUB-SS1ØØ.
     MOVE WS-TOTAL-EXCP       TO WS-PRT-SUB-EXCP.
     MOVE WS-TOTAL-CPU-CHG TO WS-PRT-SUB-CPU-CHG.
     MOVE WS-TOTAL-EXCP-CHG    TO WS-PRT-SUB-EXCP-CHG.
     MOVE WS-TOTAL-TOTAL-CHG   TO WS-PRT-SUB-CHG.
```

```
           WRITE FD-OUT-REC FROM WS-SPACES AFTER POSITIONING
                                 WS-SPACES-CC.
           WRITE FD-OUT-REC FROM WS-PRT-SUBTOTAL AFTER POSITIONING
                                 WS-SPACES-CC.
       1005-EXIT.
           EXIT.
```

THE REPORTS

To enhance the look of our report, we choose to use AFP to print on our laser printer. If AFP is not available at your shop, you don't need OGL and PPFA. However, you will have to modify PMBAT80 and PMBAT90 to user ANSI characters and create FCBs.

Overlay OGL PMD1 for laser page printing

```
//ITECS00A JOB (5110,8167100,ITECS0000),
//             'PMD1 OVERLAY        ',TIME=1440,REGION=1024K,
//             CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=ITECS00
//*
//* this is for job account detail page.
//*
//STEP1   EXEC PGM=DZIOVRLY,REGION=400K
//OUTPUT1 OUTPUT FORMDEF=OGL
//SYSPRINT DD SYSOUT=X
//SAMPLE   DD SYSOUT=9,DCB=(RECFM=VBM,LRECL=8205,BLKSIZE=8209),
//            OUTPUT=*.OUTPUT1
//OVRLIB   DD DSN=SYS1.OVERLIB,DISP=OLD
//FONTDD   DD DSN=SYS1.FONTLIBC,DISP=SHR
//*        DD DSN=SYS1.FONTLIB,DISP=SHR
//SYMBOLIC DD DUMMY
//SEGDD    DD DSN=SYS1.PSEGLIB,DISP=SHR
//SYSIN    DD *
-'GETTING STARTED'
   SETUNITS 1 IN 1 IN;
   OVERLAY PMD1 SIZE 8.5'IN 11.0 IN OFFSET 0.0 IN 0.0 IN;
   CONTROL REPLACE ALL;
   font t055fc;
   font t0759c;
   -'TEXT'
      POSITION 0.8 in 1.0 in;
      settext line t0759c 'Cost'
              line t0759c 'Centre:     _____________________';
      POSITION 3.4 in 1.0 in;
      settext line t0759c ' '
              line t0759c 'User:     _____________________';
      POSITION 5.7 in 1.0 in;
      settext line t0759c 'Account'
              line t0759c 'Manager:     _____________________';
```

```
   -'DRAWBOX'
      POSITION Ø.6 IN .25 IN;
      DRAWBOX 7.5 IN 1Ø IN MEDIUM;
      POSITION Ø.6 IN .25 IN;
      DRAWBOX 7.5 IN 1 IN MEDIUM
      withtext top left
        line tØ55fc '               User Support - '
             tØ55fc 'Detail Report for :';
      POSITION Ø.6 IN 1.25 IN;
      DRAWBOX 7.5 IN Ø.3 IN MEDIUM SHADE LIGHT
      withtext top center
        line tØ55fc 'OPERATIONS / PRODUCTION';
      POSITION Ø.6 IN 9.45 IN;
      DRAWBOX 7.5 IN Ø.8 IN MEDIUM
      withtext center left
        line tØ759c '  Rates:      IBM cpu:  $4382.39 per hour   '
             tØ759c '     DEC connect:    '
             tØ759c '  $27.ØØ per connect hour'
             tØ759c '          Dev. Charge:     $43Ø per day'
        line tØ759c '                                                '
        line tØ759c '                IBM I/O:  $2.ØØ per 1ØØØ       ';
```

PPFA to format report

```
//ITECSØØA JOB (948Ø,Ø167ØØ,TECØØØØØ1),
//             '                    ',TIME=144Ø,
//             CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=ITECSØØ
//*
//STEP1    EXEC PGM=AKQPPFA,PARM='SIZE=128K'
//STEPLIB  DD DSN=SYS1.PPFA.AKQMODØ,DISP=SHR
//SYSPRINT DD SYSOUT=*
//FORMLIB  DD DSN=SYS1.PPFAFORM,DISP=SHR
//PAGELIB  DD DSN=SYS1.PPFAPAGE,DISP=SHR
//SYSIN    DD *
FORMDEF PMD1
         OFFSET Ø.Ø IN Ø.Ø IN
         REPLACE YES;
   COPYGROUP F2PMD1
         DUPLEX NO;
         OVERLAY PMD1;
         SUBGROUP COPIES 1
                  OVERLAY PMD1;
   SETUNITS 1 IN 1 IN;
PAGEDEF PMD1
         REPLACE YES;
   FONT  TØ759C;
   FONT  TØ55FC;
   FONT  ST15;
   FONT  GU15;
   PAGEFORMAT PAGE1
              WIDTH 8.5
```

```
            HEIGHT 11
            DIRECTION ACROSS;
        SETUNITS LINESP 8 LPI;
        PRINTLINE CHANNEL 1
                  REPEAT 1
                  FONT T055FC
                  POSITION 5.5 IN 0.5 IN;
        PRINTLINE CHANNEL 2
                  REPEAT 1
                  FONT T0759C
                  POSITION 1.3 IN 1.1 IN;
        PRINTLINE CHANNEL 3
                  REPEAT 2
                  FONT GU15
                  POSITION 0.8 IN 2.0 IN;
        SETUNITS LINESP 7 LPI;
        PRINTLINE CHANNEL 4
                  REPEAT 59
                  FONT ST15
                  POSITION 0.8 IN 2.4 IN;
/*
```

Dave Tang
Manager, Systems Engineering
Southam Inc (Canada)

Register and PSW display

DESCRIPTION

When developing Assembler code, one is often faced with the requirement to be able to display the register contents and PSW without having to do a SNAP dump or cause an abend to get a dump. This routine will do both things without altering the contents of any of the registers, not even register 0 and 1 (obviously registers 14 and 15 will be altered by the LINK macro). The storage will be displayed as a WTO message, which really makes it easy to locate.

The program is called without having to set up any register contents or parameters, simply by coding LINK EP=SHOWREGS. The program is reentrant and can be called from an on-line environment, even from JES2 exits etc.

SAMPLE OUTPUT

```
LABEL1  LA      R1Ø,LABEL2
        LINK    EP=SHOWREGS
LABEL2  LA      R5,1(R5)

   +*****************************************************
   +R00:00000070, R01:00005F38, R02:00005FA8, R03:00000000
   +R04:008D6D78, R05:008F2A98, R06:008C0FF8, R07:FD000000
   +R08:008F2D48, R09:00000000, R10:12A00F96, R11:008F2A98
   +R12:92A00F46, R13:00005F38, R14:80FBAC10, R15:92A00AF0
   +PSW=078D2000 92A00F96
   +*****************************************************
```

PROGRAM SOURCE CODE FOR SHOWREGS

```
*************************************************************************
* This module displays the contents of registers and the PSW as a WTO.
* It does not modify any registers other than R14 and R15 which are
* lost anyway because of the LINK.
* Example:
*      LABEL1    LA    R1Ø,LABEL2    (This could be any instruction)
*                LINK  EP=SHOWREGS
*      LABEL2    B     MOVEDATA      (This could be any instruction)
* The register contents as at LABEL1 and the PSW as at LABEL2 will be
* displayed by the routine.
SHOWREGS CSECT
SHOWREGS AMODE 31
SHOWREGS RMODE ANY
         BAKR  R14,Ø               .Save Caller's Status
         LR    R4,RØ               .BAKR/PR does not protect RØ & R1
         LR    R5,R1               .BAKR/PR does not protect RØ & R1
         BALR  R12,Ø
         USING Storage,12
Storage  LA    R3,GetMSize         .Size of storage to get and clear
         STORAGE OBTAIN,LENGTH=(3),LOC=ANY,BNDRY=DBLWD
         LR    R2,R1               .Point to getmained area
         LA    R3,GetMSize         .Length of area
         XR    R9,R9               .Byte to propagate into area
         MVCL  R2,R8               .Propagate binary zeroes
         LR    R13,R1
         USING GetMArea,R13        .Addressability to getmained area
         STM   R4,R5,SaveRØR1      .Preserve contents of RØ & R1
         EREG  RØ,R11              .Contents of RØ-R11 as at BAKR
         STM   RØ,R11,StackRgs     .Save it
         LR    R1Ø,R12             .Preserve our base register
         LR    R11,R13             .Preserve our savearea register
         EREG  R12,R15             .As they were at the BAKR
         LR    R6,R12              .Preserve the old R12 value
         LR    R7,R13              .Preserve the old R13 value
         LR    R12,R1Ø             .Restore our base register
         LR    R13,R11             .Restore our savearea register
```

```
         STM   R6,R7,StackRgs+48   .Contents of R12 & R13 as at BAKR
         ST    R14,StackRgs+56     .Contents of R14 as at BAKR
         ST    R15,StackRgs+60     .Contents of R15 as at BAKR
         MVC   LeftByts(64),StackRgs
         NC    LeftByts(64),=64X'F0' Turn off the second part bytes
         TR    LeftByts(64),FrstByte
         MVC   RghtByts(64),StackRgs
         NC    RghtByts(64),=64X'0F' Turn off the first part bytes
         TR    RghtByts(64),SecByte
         MVC   WTOArea(WTOLeng),WTO
         WTO   '**************************************************',X
               ROUTCDE=11
         LA    R9,LeftByts
         LA    R10,RghtByts
         LA    R4,Regs             .Character displays of registers
         LA    R5,4                .4 Rows of WTO messages
NextRow  EQU   *
         LA    R6,4                .4 Entries per row
FrstDgt  LA    R8,WTOArea+5        .Where first digit goes
         MVC   0(2,R8),0(R4)       .Register's number
         LA    R4,2(R4)
         LA    R8,3(R8)
Entries  LA    R7,4                .4x2 Bytes per register
NextChar EQU   *
         MVC   0(1,R8),0(R9)
         LA    R8,1(R8)            .Point to next character (target)
         MVC   0(1,R8),0(R10)
         LA    R8,1(R8)            .Point to next character (target)
         LA    R9,1(R9)            .Point to next character (source)
         LA    R10,1(R10)          .Point to next character (source)
         BCT   R7,NextChar
         MVC   3(2,R8),0(R4)       .Register number
         LA    R4,2(R4)            .Next entry in register name list
         LA    R8,6(R8)            .Where next register's info starts
         BCT   R6,Entries
         SH    R4,=H'2'            .Reduce by 2
DoWTO    LA    R1,WTOArea
         WTO   MF=(E,(1))          .WTO next line of register info
         BCT   R5,NextRow          .Do for each of the 4 rows
         XR    R1,R1               .Address of PSA
         USING PSA,R1
         L     R1,PSATOLD          .Address of current TCB
         DROP  R1
         USING TCB,R1
         L     R1,TCBRBP           .Address of current RB
         DROP  R1
         USING RBBASIC,R1
         ICM   R1,7,RBLINKB        .Address of previous PRB
         ICM   R1,8,=X'00'         .RB address is 3-byte address
         MVC   KeepPSW,RBOPSW      .Preserve the PSW to analyse
         MVC   LeftByts(8),RBOPSW  .Make OLDPSW printable
         NC    LeftByts(8),=64X'F0' Turn off the second part bytes
```

```
         TR    LeftByts(8),FrstByte
         MVC   RghtByts(8),RBOPSW  .Make OLDPSW printable
         NC    RghtByts(8),=64X'0F' Turn off the first part bytes
         TR    RghtByts(8),SecByte
         MVC   PSWWTOA(PSWWTOL),PSWWTO
         LA    R1,LeftByts         .Where the 1st half of each byte is
         LA    R2,RghtByts         .Where the 2nd half of each byte is
         LA    R3,PSWWTOA+8        .Where we want to move the data to
         LA    R4,8                .8 Bytes in the PSW
PSWLoop  MVC   0(1,R3),0(R1)       .Move first half of byte
         LA    R3,1(R3)            .Bump up target pointer
         MVC   0(1,R3),0(R2)       .Move second half of byte
         LA    R3,1(R3)            .Bump up target pointer
         LA    R1,1(R1)            .Bump up first-half-of-byte pointer
         LA    R2,1(R2)            .Bump up second-half-of-byte pointer
         CH    R4,=H'5'            .Halfway?
         BNE   PSWLoopX            .No
         LA    R3,1(R3)            .Yes, leave a blank
PSWLoopX BCT   R4,PswLoop          .Do for each of the 8 bytes
         LA    R1,PSWWTOA          .Point to PSW WTO message area
         WTO   MF=(E,(1))          .WTO the PSW as at entry
         WTO '*************************************************', X
               ROUTCDE=11
Return   EQU   *                   .Pick up return code
         LA    R3,GetMSize         .Size of area to free
         LR    R2,R13              .Address of area to free
         LM    R4,R5,SaveR0R1      .Old values of R0 & R1
         L     R5,SaveR1
         STORAGE RELEASE,LENGTH=(R3),ADDR=(R2)
         XR    R15,R15             .Copy return code
         LR    R0,R4
         LR    R1,R5
ToCaller PR    ,                   .=>Caller
SecByte  DC    X'F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6'
FrstByte DS    0CL240
         DC    X'F0',15X'00',X'F1',15X'00',X'F2',15X'00',X'F3'
         DC    15X'00',X'F4',15X'00',X'F5',15X'00',X'F6',15X'00',X'F7'
         DC    15X'00',X'F8',15X'00',X'F9',15X'00',X'C1',15X'00',X'C2'
         DC    15X'00',X'C3',15X'00',X'C4',15X'00',X'C5',15X'00',X'C6'
*
WTO      WTO 'Rxx:xxxxxxxx, Rxx:xxxxxxxx, Rxx:xxxxxxxx, Rxx:xxxxxxxx', X
               ROUTCDE=11,MF=L
WTOLeng  EQU   *-WTO
PSWWTO   WTO   'PSW=xxxxxxxx xxxxxxxx',ROUTCDE=11,MF=L
PSWWTOL  EQU   *-PSWWTO
REGS     DC    C'000102030405060708091011121314 15'
         LTORG
GetmArea DSECT
SaveArea DS    18F
StackRgs DS    16F
LeftByts DS    16F
RghtByts DS    16F
```

```
SaveRØR1 DS    D
KeepPSW  DS    F
WTOArea  DS    CL(WTOLeng)
PSWWTOA  DS    CL(PSWWTOL)
GetMSize EQU   *-GetMarea
         IHAPSA
         IHARB
         IKJTCB
         END
```

A A Keyser
Systems Programmer
Houghton Consulting Services Pty Ltd (Australia)

Shared pages

OVERVIEW

Virtual addressing permits an addressing range that is greater than the central storage capabilities of the MVS system. The potentially large number of address spaces provides the system with a large virtual addressing capacity. Shared pages is a new function which was introduced in MVS/ESA 5.2.0. It permits more than one virtual storage page to simultaneously share the same system resources. This can significantly reduce storage requirements in many types of application as multiple virtual pages can use the same real storage frame, the same expanded frame, or the same slot on auxiliary DASD.

IARVSERV

Shared pages is provided by the IARVSERV macro services (RSM Virtual Storage Services). This macro provides the interface to implement data sharing among different virtual storage areas.

The terminology used in the IBM reference manuals when explaining the shared pages concept is as follows:

Source area — The data to be shared is called the source. This refers to the actual source data in the virtual storage that contains the data.

Target area	The target area is used to describe the virtual storage area where the source data is made available as shareable.
Sharing group	The source and its corresponding target form a sharing group. A sharing group can consist of several target areas, all using the same source data.
Sharing pages	A page (4K) is called a sharing page if it is a member of a sharing group.
Sharing programs	Programs that access the source or target areas.

The services that are provided by the IARVSERV macro area:

SHARE	The SHARE parameter requests that a source of data is made available through a given virtual storage area (target).
UNSHARE	The UNSHARE parameter requests that the specified virtual storage area (target) no longer shares storage.
CHANGEACCESS	The CHANGEACCESS parameter requests that the type of access to the specified virtual storage is changed.

Using the IARVSERV macro service, virtual storage can be shared by multiple address spaces and data spaces. The following areas cannot be used:

- Hiperspace
- VIO window
- V=R region
- PSA.

The target area cannot contain page-protected or page-fixed pages.

There are six types of data sharing and each is called a specific view of the data. This view is the way the program accesses the target virtual storage and is specified through the TARGET_VIEW parameter.

Read-only view (READONLY)

Read-only view specifies that the target data may not be modified.

Shared-write view (SHAREDWRITE)

Shared-write view specifies that the target data can be read and modified through the view.

Copy-on-write view (UNIQUEWRITE)

Copy-on-write (UNIQUEWRITE) specifies that the target can be used to read shared data and to retain a private copy of the shared data should the source or any target get altered.

Copy-on-write view (TARGETWRITE)

Copy-on-write view (TARGETWRITE) specifies that the target can be used to read shared data and retain a private copy of the shared data if this view of the shared data is altered.

The copy-on-write attribute is available when suppression-on-protection is present on the processor. Suppression-on-protection is available on the following models:

- S/390 9672 processors
- 9021 711 based models
- 9121 511 based models
- 9221 211 based models.

Like-source view (LIKESOURCE)

Like-source view specifies that the view type for the new target area is to be the same as the current view of the source.

Hidden view (HIDDEN)

Hidden-view specifies that the data in the target area will be inaccessible until the view type is changed to READONLY, SHAREDWRITE, UNIQUEWRITE, or TARGETWRITE.

The virtual storage areas that are to be shared are specified using the RANGLIST parameter. The RANGLIST parameter points to a number of entries where each entry is 28 bytes long. A mapping of each

entry is provided through the IBM mapping macro IARVRL. The NUMRANGE parameter specifies the number of entries in the supplied RANGLIST.

The maximum number of shared pages for a program in problem state with PSW key 8-15 is 32. This number includes both the source and targets, so that the actual number of unique pages is 16. The number can be dynamically changed by using the SMF exit SYS.IEFUSI (step initiation exit). The maximum number that can be specified is (2**31)-1.

EXPLOITATION BY MVS

UCB Virtual Storage Constraint Relief (VSCR) allows UCBs to be defined in 31-bit storage above the 16-megabyte line. Because of the large amount of third-party software and user code that relies on old interfaces that use 24-bit UCB pointers (eg TIOT UCB address – TIOEFRSR), IBM has needed to maintain this compatibility. This has been provided through the captured UCB.

The captured UCB is created by exploiting the RSM shared pages support. The UCBs that are moved above the 16-megabyte line can now be accessed through a 24-bit window, when they cannot be accessed directly. Captured UCBs are created and destroyed during device allocation and deallocation and reside in private LSQA storage. IOS provides the mechanism to create and destroy captured UCBs through a new programming interface IOSCAPU.

RMF SUPPORT

RMF provides the following features to report shared pages or shared page groups.

Monitor I	Measurements in the paging activity report and the workload activity report.
Monitor II	The shared storage page-in-rate related to each address space (ASD/ASDJ).
Monitor III	The shared storage page-in rate that is related to each address space is included in the GROUP, STORS, STORF, and STORJ reports.

IPCS

The IPCS RSMDATA subcommand with the SHRDATA parameter provide a detailed report on the status of IARVSERV data sharing. The MVS DISPLAY command also displays information about shared data.

EXAMPLE

The Assembler source (SHAREDPG) in this article provides an example of two types of data sharing using the IARVSERV macro:

1 Sharing a 2-page virtual storage area within the same address space (option 1).

2 Sharing a 10-page virtual storage area between two data spaces (option 2).

I have included an ABEND macro for the first example, so that the source and target areas can be examined in the dump that is produced.

For the second example, I issue an SDUMPX macro to produce an SVC dump containing the batch address space and the two data spaces (IARVSERV1 and IARVSRV2) which are created. The SVC dump can be examined under IPCS to verify the data space storage which is shared. The IPCS browse option (option 1) can be used to accomplish this by specifying the data space names on the browse option panel:

```
. Address Space ===>    ASID(X'nn')  DSPNAME(IARVSERVn)
```

The program is invoked as follows.

Option 1

```
//OPT1     EXEC PGM=SHAREDPG,PARM='1'
//SYSPRINT DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
```

Option 2

```
//OPT2     EXEC PGM=SHAREDPG,PARM='2'
//SYSPRINT DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
```

Because CSA storage is required for the data space list used by the DUMPX macro, the program must be link-edited into an authorized

library. The minimum authorization that is required to issue the IARVSERV macro is problem state with a PSW key that allows access to the source, target, or both, depending on the value specified through the TARGET_VIEW parameter.

SHAREDPG SOURCE CODE

```
         TITLE 'MVS/EAS 5.2 SHARED PAGES EXAMPLE'
*--------------------------------------------------------------------*
* NAME:          SHAREDPG                                            *
* AUTHOR:        REM PERRETTA                                        *
* LANGUAGE:      IBM ASM/370                                         *
* PURPOSE:       THIS ROUTINE WILL SET UP AND THEN TEST THE SHARED    *
*                PAGE FACILITY WHICH WAS INTRODUCED IN MVS/ESA 5.2.   *
*                IT HAS THE FOLLOWING OPTIONS WHICH ARE SPECIFIED     *
*                THROUGH THE JOB PARM FACILITY:                      *
*                1. EXEC PGM=SHAREDPG,PARM='1'                       *
*                   SHARE PAGES BETWEEN STORAGE AREAS IN THE SAME    *
*                   ADDRESS SPACE.                                   *
*                   AN ABEND MACRO IS ISSUED TO OBTAIN A DUMP.       *
*                2. EXEC PGM=SHAREDPG,PARM='2'                       *
*                   SHARE PAGES BETWEEN TWO DATA SPACES.             *
*                   AN SDUMPX MACRO IS ISSUED TO SVC DUMP THE TWO    *
*                   DATA SPACES SO THAT THEY CAN BE BROWSED USING    *
*                   IPCS.                                            *
* INVOCATION:                                                        *
* FROM THE REXX EXEC:                                                *
*              VARIOUS                                               *
* INPUT PARAMETERS:                                                  *
*       OPTION FLAG                                                  *
*       1 BYTE CHARACTER VALUE AS FOLLOWS:                           *
*       C'1' =  SHARED PAGES USING OBTAINED STORAGE                  *
*       C'2' =  SHARED PAGES USING DATA SPACES.                      *
* R15 ON RETURN IS SET AS FOLLOWS:                                   *
*     0           NO PARMS                                         *
*     4           PARM LENGTH > 1                                  *
*     8           NO TABLE ENTRIES RETURNED                        *
*     12          INVALID OPTION SPECIFIED                         *
* THE FOLLOWING ABEND CODES ARE ISSUED                               *
*     ABEND 100       IARVSERV SHARE ERROR                         *
*     ABEND 200       IARVSERV UNSHARE ERROR                       *
*     ABEND 300       DSPSERV ERROR                                *
*     ABEND 400       DSPSERV ERROR                                *
*     ABEND 500       ALESERV ERROR                                *
*     ABEND 500       ALESERV ERROR                                *
*     ABEND 700       DSPSERV ERROR                                *
*     ABEND 800       DSPSERV ERROR                                *
*     ABEND 900       DUMPX   ERROR                                *
*--------------------------------------------------------------------*
```

```
ZERO     EQU   X'ØØ'                    ZERO
SPACE    EQU   C' '                     SPACE
SIGNF    EQU   X'FØ'                    POSITIVE SIGN
OPT1     EQU   C'1'                     SHARED PAGES BY STORAGE OBTAIN
OPT2     EQU   C'2'                     SHARED PAGES BY DATA SPACES
SHAREDPG CSECT
SHAREDPG AMODE 31
SHAREDPG RMODE ANY
         BAKR  R14,Ø                    SAVE CALLER'S ARS + GPRS
*                                       IN THE LINKAGE STACK
         USING SHAREDPG,R12             INFORM THE ASSEMBLER
         LAE   R12,Ø(R15,Ø)             SET UP PROGRAM BASE REGISTER
         L     R9,Ø(,R1)                PARAMETER @
         USING INPPARM,R9               INFORM THE ASSEMBLER
*.....................................................................
* HOUSEKEEPING                                                       .
*.....................................................................
STOREGET EQU   *
         L     R8,=AL4(WORKALEN)        WORK AREA LENGTH
GETWORK  STORAGE OBTAIN,                GET STORAGE                    X
               LENGTH=(R8),             LENGTH                         X
               ADDR=(R1Ø),              @ OF STORAGE                   X
               SP=Ø,KEY=8,              SUBPOOL AND KEY                X
               LOC=BELOW,               BELOW THE 16M LINE             X
               COND=NO,                 UNCONDITIONAL                  X
               RELATED=(FREEWORK,'FREE WORK AREA')
         LAE   R13,Ø(R1Ø,Ø)             @ THE WORKAREA
         USING SAVEAREA,R13             INFORM THE ASSEMBLER
         LA    RØ,SAVEAREA              @ THE WORKAREA
         ICM   R1,B'1111',=AL4(WORKALEN) LENGTH
         SR    R14,R14                  ZERO FILL
         SR    R15,R15                  ZERO FILL
         MVCL  RØ,R14                   CLEAR THE AREA
         MVC   PREVSA,=C'F1SA'          PUT ACRONYM INTO SAVEAREA
*                                       TO INDICATE STATUS SAVED ON
*                                       THE LINKAGE STACK.
         CLC   PARMLEN,=X'ØØØØ'         ANY PARMS?
         BE    RETURN1                  NO-
         CLC   PARMLEN,=X'ØØØ1'         PARM LEN > 1?
         BNE   RETURN2                  NO-
*.....................................................................
* PROCESS THE USER OPTION                                            .
*.....................................................................
TESTOPT  EQU   *
         CLI   OPTION,OPT1              OPTION 1?
         BNE   OPTN2                    NO-
         BAS   R2,OPTION1               LET'S DO IT
         XR    R1Ø,R1Ø                  ZERO RETURN CODE
         B     CLEANUP                  LET'S RETURN
OPTN2    EQU   *
         CLI   OPTION,OPT2              OPTION 1?
         BNE   INVOPT                   NO-
```

```
         BAS   R2,OPTION2              LET'S DO IT
         B     CLEANUP                 LET'S RETURN
RETURN1  EQU   *
         LA    R1Ø,4(Ø,Ø)              NO PARMS
         B     CLEANUP                 LET'S RETURN
RETURN2  EQU   *
         LA    R1Ø,8(Ø,Ø)              PARM LENGTH > 1?
         B     CLEANUP                 LET'S RETURN
INVOPT   EQU   *
         LA    R1Ø,12(Ø,Ø)             INVALID OPTION
         B     CLEANUP                 LET'S RETURN
*.....................................................................
* FREE THE OBTAINED STORAGE AND EXIT                                 .
*.....................................................................
CLEANUP  EQU   *
         LAE   R1,Ø(R13,Ø)              ADDRESS TO FREE
         L     R9,=AL4(WORKALEN)        WORK AREA LENGTH
FREEWORK STORAGE RELEASE,               RELEASE STORAGE                X
               ADDR=(R1),               ADDRESS TO GIVE BACK           X
               LENGTH=(R9),             LENGTH OF STORAGE              X
               SP=Ø,KEY=8,              SUBPOOL AND KEY                X
               COND=NO,                 UNCONDITIONAL                  X
               RELATED=(GETWORK,'GET WORK AREA')
EXIT     EQU   *
         LR    R15,R1Ø                  SET RC
         PR                             RESTORE CALLER'S ARS
*                                       GPRS 2-14 AND RETURN
*                                       TO CALLER
         TITLE 'SHARED PAGES BY STORAGE OBTAIN'
*.....................................................................
* SHARE STORAGE BETWEEN STORAGE AREAS IN THE SAME ADDRESS SPACE      .
*.....................................................................
OPTION1  EQU   *
         LA    R4,VRLAREA               @ THE VRL ENTRY
         USING VRL,R4                   INFORM THE ASSEMBLER
         XC    VRL(VRLLEN),VRL          CLEAR THE VRL
         XC    SOURCEALET,SOURCEALET    PASN
         XC    TARGETALET,TARGETALET    PASN
GETSHR1  STORAGE OBTAIN,                GET SHARED SOURCE STORAGE      X
               LENGTH=SHAREDAREALEN,    STORAGE LENGTH                 X
               SP=Ø,KEY=8,              SUBPOOL Ø KEY 8                X
               LOC=RES,                 LOCATION = RESIDENCY           X
               COND=NO,                 UNCONDITIONAL                  X
               RELATED=(FREESHR1,'FREE SOURCE SHARED AREA 1')
         STCM  R1,B'1111',SOURCE        SAVE THE SOURCE ADDRESS
GETSHR2  STORAGE OBTAIN,                GET TARGET STORAGE             X
               LENGTH=SHAREDAREALEN,    STORAGE LENGTH                 X
               SP=Ø,KEY=8,              SUBPOOL AND KEY                X
               LOC=BELOW,               BELOW THE LINE                 X
               COND=NO,                 UNCONDITIONAL                  X
               RELATED=(FREESHR2,'FREE TARGET AREA 2')
         STCM  R1,B'1111',TARGET        SAVE THE TARGET ADDRESS
```

```
         ICM   RØ,B'1111',SOURCE       SOURCE @
         L     R1,=AL4(SHAREDAREALEN)  LENGTH OF SHARED AREA
         LA    R14,=C'R'               INIT CHARACTER
         LA    R15,1(Ø,Ø)              INITIAL LENGTH TO MOVE
         ICM   R15,B'1ØØØ',=C'R'       PAD CHARACTER
         MVCL  RØ,R14                  INIT THE SOURCE AREA
         ICM   R5,B'1111',SOURCE       SOURCE @
         STCM  R5,B'1111',VRLSVSA      STORE
         ICM   R5,B'1111',SOURCEALET   SOURCE ALET
         STCM  R5,B'1111',VRLSALET     STORE
         ICM   R5,B'1111',=AL4(SHAREDAREALEN/4Ø96) SHARED AREA LENGTH
         STCM  R5,B'1111',VRLNUMPG     NUMBER OF PAGES
         ICM   R5,B'1111',TARGET       TARGET @
         STCM  R5,B'1111',VRLTVSA      STORE
         ICM   R5,B'1111',TARGETALET   TARGET ALET
         STCM  R5,B'1111',VRLTALET     STORE
         STCM  R4,B'1111',VRLADDR      STORE
*.....................................................................
* ISSUE IARVSERV TO SHARE THE DATA                                   .
*.....................................................................
         IARVSERV SHARE,               SHARE SOURCE WITH TARGET        X
               RANGLIST=VRLADDR,       STORAGE ADDRESSES               X
               TARGET_VIEW=READONLY,   READ ONLY                       X
               PLISTVER=MAX,           MAX PARAMETER LIST              X
               MF=(E,IARVSRV,COMPLETE) EXECUTE FORM
         LTR   R15,R15                 SHARED OK?
         BNZ   ABEND1ØØ                NO-
*.....................................................................
* UNCOMMENT THE FOLLOWING ABEND SO THAT THE SOURCE AND TARGET AREA   .
* CAN BE CHECKED(SOURCE,TARGET).                                     .
*.....................................................................
         ABEND 999,DUMP                CHECK THE TARGET AND SOURCE
*.....................................................................
* AT THIS POINT, WE ARE NOW SHARING DATA BETWEEN THE SOURCE AND      .
* TARGET.                                                            .
*.....................................................................
* LETS RELEASE SHARED AREA                                           .
*.....................................................................
         IARVSERV UNSHARE,             UNSHARE SOURCE WITH TARGET      X
               RANGLIST=VRLADDR,       STORAGE ADDRESSES               X
               RETAIN=NO,              RELEASE THE STORAGE             X
               PLISTVER=MAX,           MAX PARAMETER LIST              X
               MF=(E,IARVSRV,COMPLETE) EXECUTE FORM
         LTR   R15,R15                 UNSHARE OK?
         BNZ   ABEND2ØØ                NO-
*.....................................................................
* FREE THE SOURCE AND TARGET AREAS                                   .
*.....................................................................
         ICM   R1,B'1111',SOURCE       SOURCE ADDRESS
FREESHR1 STORAGE RELEASE,              RELEASE STORAGE                 X
               LENGTH=SHAREDAREALEN,   STORAGE LENGTH                  X
               SP=Ø,KEY=8,             SUBPOOL AND KEY                 X
```

```
              ADDR=(R1),              ADDRESS TO FREE                X
              COND=NO,                UNCONDITIONAL                  X
              RELATED=(GETSHR1,'GET SOURCE AREA')
         ICM  R1,B'1111',TARGET       TARGET ADDRESS
FREESHR2 STORAGE RELEASE,             RELEASE STORAGE                X
              LENGTH=SHAREDAREALEN,   STORAGE LENGTH                 X
              SP=0,KEY=8,             SUBPOLL AND KEY                X
              ADDR=(R1),              ADDRESS TO FREE                X
              COND=NO,                UNCONDITIONAL                  X
              RELATED=(GETSHR1,'GET TARGET AREA')
         DROP R4                      INFORM THE ASSEMBLER
         BR   R2                      RETURN TO CALLER
         TITLE 'SHARED PAGES BY DATA SPACES'
*.....................................................................
* SHARE STORAGE BETWEEN TWO DATA SPACES                              .
*.....................................................................
OPTION2  EQU  *
         LA   R8,VRLAREA              @ THE VRL ENTRY
         USING VRL,R8                 INFORM THE ASSEMBLER
         XC   VRL(VRLLEN),VRL         CLEAR THE VRL
         XC   SOURCEALET,SOURCEALET   PASN
         XC   TARGETALET,TARGETALET   PASN
*.....................................................................
* CREATE DATA SPACE 1                                                .
*.....................................................................
         DSPSERV CREATE,              CREATE A DATA SPACE            X
              STOKEN=DSPTOKEN1,       STOKEN OF NEW DATA SPACE       X
              NAME=DATASPACENAME1,    DATA SPACE NAME                X
              BLOCKS=DSPSIZE,         DATA SPACE SIZE                X
              ORIGIN=DATASPACESTART1, ORIGIN OF DATA SPACE           X
              SCOPE=SINGLE,           SINGLE                         X
              DREF=NO,                NO DREF STORAGE                X
              GENNAME=NO,             WE WILL NAME THE DATA SPACE    X
              MF=(E,DSP1,COMPLETE)    EXECUTE FORM
         LTR  R15,R15                 DATA SPACE CREATED?
         BNZ  ABEND300                NO-
*.....................................................................
* CREATE DATA SPACE 2                                                .
*.....................................................................
         DSPSERV CREATE,              CREATE A DATA SPACE            X
              STOKEN=DSPTOKEN2,       STOKEN OF NEW DATA SPACE       X
              NAME=DATASPACENAME2,    DATA SPACE NAME                X
              BLOCKS=DSPSIZE,         DATA SPACE SIZE                X
              ORIGIN=DATASPACESTART2, ORIGIN OF DATA SPACE           X
              SCOPE=SINGLE,           SINLGE                         X
              DREF=NO,                NO DREF STORAGE                X
              GENNAME=NO,             WE WILL NAME THE DATA SPACE    X
              MF=(E,DSP2,COMPLETE)    EXECUTE FORM
         LTR  R15,R15                 DATA SPACE CREATED?
         BNZ  ABEND400                NO-
         MVC  ALETA1,ALSRVLST         ALET PARAMETER AREA
```

```
*.....................................................................
* ADD AN ENTRY TO THE PASN ACCESS LIST                               .
*.....................................................................
         ALESERV ADD,                  ADD TO THE PASN ACCESS LIST     X
               STOKEN=DSPTOKEN1,       STOKEN NAME                     X
               ACCESS=PUBLIC,          MAKE IT PUBLIC                  X
               AL=PASN,                PUT IT ON THE PASN-AL           X
               ALET=DSP1ALET,          ALET OF NEW DATA SPACE          X
               MF=(E,ALETA1)           EXECUTE FORM
         LTR   R15,R15                 ALET RETURNED?
         BNZ   ABEND5ØØ                NO-
*.....................................................................
* ADD AN ENTRY TO THE PASN ACCESS LIST                               .
*.....................................................................
         MVC   ALETA2,ALSRVLST         ALET PARAMETER AREA
         ALESERV ADD,                  ADD TO THE PASN ACCESS LIST     X
               STOKEN=DSPTOKEN2,       STOKEN NAME                     X
               ACCESS=PUBLIC,          MAKE IT PUBLIC                  X
               AL=PASN,                PUT IT ON THE PASN-AL           X
               ALET=DSP2ALET,          ALET OF NEW DATA SPACE          X
               MF=(E,ALETA2)           EXECUTE FORM
         LTR   R15,R15                 ALET RETURNED?
         BNZ   ABEND6ØØ                NO-
*.....................................................................
* GET INTO AR MODE AND CONSTRUCT THE IARSERV RANGE LIST              .
*.....................................................................
ARMODE   EQU   *
         SAC   512                     GET INTO AR MODE
         ICM   R4,B'1111',DATASPACESTART1 DATA SPACE 1 ORIGIN
         ICM   R5,B'1111',DSP1ALET     DATA SPACE 1 ALET
         SAR   R4,R5                   GPR4/AR4
         ICM   R6,B'1111',DSPSIZE      DATA SPACE SIZE
INITDTSP EQU   *
         MVC   Ø(L'DATASPACEINIT,R4),DATASPACEINIT INIT FIRST 16 BYTES
*                                      OF 4K
         AL    R4,=F'4Ø96'             NEXT 4K
         BCT   R6,INITDTSP             DO WHILE R6 > Ø?
         ICM   R5,B'1111',DATASPACESTART1 SOURCE @
         STCM  R5,B'1111',VRLSVSA      STORE
         ICM   R5,B'1111',DSP1ALET     SOURCE ALET
         STCM  R5,B'1111',VRLSALET     STORE
         ICM   R5,B'1111',DSPSIZE      DATA SPACE SIZE IN PAGES
         STCM  R5,B'1111',VRLNUMPG     NUMBER OF PAGES
         ICM   R5,B'1111',DATASPACESTART2 TARGET @
         STCM  R5,B'1111',VRLTVSA      STORE
         ICM   R5,B'1111',DSP2ALET     TARGET ALET
         STCM  R5,B'1111',VRLTALET     STORE
         STCM  R8,B'1111',VRLADDR      STORE
*.....................................................................
* ISSUE IARVSERV TO SHARE THE DATA                                   .
*.....................................................................
```

```
         IARVSERV SHARE,               SHARE SOURCE WITH TARGET        X
               RANGLIST=VRLADDR,       STORAGE ADDRESSES               X
               TARGET_VIEW=READONLY,   READ ONLY                       X
               PLISTVER=MAX,           MAX PARAMETER LIST              X
               MF=(E,IARVSRV,COMPLETE) EXECUTE FORM
         LTR   R15,R15                 SHARED OK?
         BNZ   ABEND100                NO-
         SAC   0                       PRIMARY MODE
         MVC   MDESETX,MDESETL1        MOVE FOR EXECUTE FORM
         MODESET MF=(E,MDESETX)        SUPV STATE KEY 0
GCSASTOR STORAGE OBTAIN,               STORAGE FOR DATA SPACE LIST     X
               LENGTH=DLISTLEN,        DATA SPACE LIST LEN             X
               SP=241,KEY=8,           CSA KEY 8                       X
               LOC=RES,                GET STORAGE AS PER RESIDENCY    X
               COND=NO,                UNCONDITIONAL                   X
               RELATED=(FCSASTOR,'FREE CSA STORAGE')
         LR    R8,R1                   ADDRESS THE AREA
         USING DLISTARA,R8             INFORM THE ASSEMBLER
         MVC   DLISTARA,DSPACELIST     MOVE THE DSPLIST
         MVC   SDUMPXA,SDUMP           SDUMPX PARAMETER LIST
         XC    WAITECB,WAITECB         CLEAR ECB
         LA    R9,WAITECB              ECB @
*.....................................................................
* ISSUE THE SDUMPX COMMAND TO PRODUCE AN SVC DUMP OF THE DATASPACES.
*.....................................................................
         SDUMPX PLISTVER=3,                                            X
               DSPLIST=(R8),                                           X
               ECB=((R9),WRITE),                                       X
               SDATA=(ALLPSA,NUC,SQA,RGN,LSQA,IO,CSA),                 X
               SUSPEND=NO,                                             X
               TYPE=FAILRC,                                            X
               HDR='IARVSERV DATA SPACE TEST',                         X
               ID='SHAREDPG DATA SPACE TEST',                          X
               MF=(E,SDUMPXA)
         LTR   R15,R15                 SDUMP OK?
         BNZ   ABEND900                NO-
*.....................................................................
* WAIT FOR THE DUMP TO BE PROCESSED                                  .
*.....................................................................
         WAIT  ECB=WAITECB,            WAIT FOR SDUMPX TO COMPLETE     X
               LINKAGE=SVC             SVC ENTRY
FCSASTOR STORAGE RELEASE,              FREE CSA STORAGE                X
               LENGTH=DLISTLEN,        STORAGE LENGTH                  X
               SP=241,KEY=8,           CSA KEY 8                       X
               ADDR=(R8),              CSA AREA TO RELEASE             X
               COND=NO,                UNCONDITIONAL                   X
               RELATED=(GDSSTOR,'FREE DATA SPACE LIST')
*.....................................................................
* AS THIS POINT, WE ARE NOW SHARING DATA BETWEEN THE SOURCE AND     .
* TARGET DATA SPACES                                                 .
*.....................................................................
```

```
* LETS CLEAN UP                                                  .
*.....................................................................
         MVC   MDESETX,MDESETL2         MOVE FOR EXECUTE FORM
         MODESET MF=(E,MDESETX)         PROB STATE KEY 8
*.....................................................................
* LETS UNSHARE THE STORAGE ACROSS THE DATA SPACES                 .
*.....................................................................
         IARVSERV UNSHARE,              UNSHARE SOURCE WITH TARGET      X
               RANGLIST=VRLADDR,        STORAGE ADDRESSES               X
               RETAIN=NO,               RELEASE THE STORAGE             X
               PLISTVER=MAX,            MAX PARAMETER LIST              X
               MF=(E,IARVSRV,COMPLETE) EXECUTE FORM
         LTR   R15,R15                  UNSHARE OKAY?
         BNZ   ABEND200                 NO-
*.....................................................................
* DELETE THE DATA SPACES                                          .
*.....................................................................
         DSPSERV DELETE,                DELETE DATA SPACE 1             X
               STOKEN=DSPTOKEN1,        STOKEN OF DATA SPACE            X
               MF=(E,DSP1,COMPLETE)     EXECUTE FORM
         LTR   R15,R15                  DATA SPACE DELETED?
         BNZ   ABEND700                 NO-
         DSPSERV DELETE,                DELETE DATA SPACE 2             X
               STOKEN=DSPTOKEN2,        STOKEN OF DATA SPACE            X
               MF=(E,DSP2,COMPLETE)     EXECUTE FORM
         LTR   R15,R15                  DATA SPACE DELETED?
         BNZ   ABEND800                 NO-
         DROP  R8                       INFORM THE ASSEMBLER
         BR    R2                       RETURN TO CALLER
         TITLE 'ABEND ROUTINES'
ABEND100 EQU   *
         LR    R5,R15                   IARVSERV RETURN CODE
         LR    R6,R0                    IARVSERV REASON CODE
         ABEND 100,DUMP                 IARVSERV SHARE ERROR
ABEND200 EQU   *
         LR    R5,R15                   IARVSERV RETURN CODE
         LR    R6,R0                    IARVSERV REASON CODE
         ABEND 200,DUMP                 IARVSERV UNSHARE ERROR
ABEND300 EQU   *
         LR    R5,R15                   DSPSERV RETURN CODE
         LR    R6,R0                    DSPSERV REASON CODE
         ABEND 300,DUMP                 DSPSERV ERROR
ABEND400 EQU   *
         LR    R5,R15                   DSPSERV RETURN CODE
         LR    R6,R0                    DSPSERV REASON CODE
         ABEND 400,DUMP                 DSPSERV ERROR
ABEND500 EQU   *
         LR    R5,R15                   ALESERV RETURN CODE
         ABEND 500,DUMP                 ALESERV ERROR
ABEND600 EQU   *
         LR    R5,R15                   ALESERV RETURN CODE
         ABEND 500,DUMP                 ALESERV ERROR
```

```
ABEND700 EQU   *
         LR    R5,R15                  DSPSERV RETURN CODE
         LR    R6,R0                   DSPSERV REASON CODE
         ABEND 700,DUMP                DSPSERV ERROR
ABEND800 EQU   *
         LR    R5,R15                  DSPSERV RETURN CODE
         LR    R6,R0                   DSPSERV REASON CODE
         ABEND 800,DUMP                DSPSERV ERROR
ABEND900 EQU   *
         STORAGE RELEASE,              FREE CSA STORAGE                X
               LENGTH=DLISTLEN,        STORAGE LENGTH                  X
               SP=241,KEY=8,           CSA KEY 8                       X
               ADDR=(R8),              CSA AREA TO RELEASE             X
               COND=NO,                UNCONDITIONAL                   X
               RELATED=(GDSSTOR,'FREE DATA SPACE LIST')
         LR    R5,R15                  DUMPX   RETURN CODE
         ABEND 900,DUMP                DUMPX   ERROR
         TITLE 'LTORG'
*.....................................................................
* LTORG                                                              .
*.....................................................................
         LTORG
         TITLE 'NON-DYNAMIC STORAGE'
*.....................................................................
* STATIC STORAGE DEFINITIONS                                        .
*.....................................................................
DATASPACENAME1 DC  CL8'IARVSRV1'       DATA SPACE NAME 1
DATASPACENAME2 DC  CL8'IARVSRV2'       DATA SPACE NAME 2
DATASPACEINIT  DC  CL16'IARVSERVDATASPCE' DATA SPACE INIT
DSPSIZE        DC  F'10'               DATA SPACE SIZE 10 X 4096
ALSRVLST ALESERV MF=L
ALSRVLLN       EQU *-ALSRVLST          ALSERV PARAMETER LENGTH
SDUMP    SDUMPX HDR='IARVSERV DATA SPACE TEST',                        X
               PLISTVER=3,                                           X
               SDATA=(ALLPSA,ALLNUC,SQA,RGN,LSQA,IO,CSA),            X
               SUSPEND=NO,                                           X
               DSPLIST=DSPACELIST,                                   X
               TYPE=FAILRC,                                          X
               ID='SHAREDPG DATA SPACE TEST',                        X
               MF=L
SDUMPLEN EQU   *-SDUMP
MDESETL1 MODESET KEY=ZERO,MODE=SUP,MF=L  LIST FORM OF MODESET
MSETLEN1 EQU    *-MDESETL1             LENGTH OF PARAMETER LIST
MDESETL2 MODESET KEY=NZERO,MODE=PROB,MF=L  LIST FORM OF MODESET
MSETLEN2 EQU    *-MDESETL2             LENGTH OF PARAMETER LIST
DSPACELIST     DS  0X
               DC  AL4(36)
               DC  CL16'SHTS001AIARVSRV1'
               DC  CL16'SHTS001AIARVSRV2'
DLISTLEN       EQU *-DSPACELIST
         TITLE 'WORKAREA DSECT'
```

```
*.....................................................................
* DYNAMIC STORAGE DEFINITIONS                                        .
*.....................................................................
WORKAREA DSECT
SAVEAREA DS    CL72                    SAVEAREA
PREVSA   EQU   SAVEAREA+4,4            @ OF PREVIOUS SAVEAREA
WAITECB  DS    F                       SDUMPX WAIT ECB
SOURCE   DS    AL4                     SOURCE @
TARGET   DS    AL4                     TARGET @
VRLADDR  DS    AL4                     VRL @
SOURCEALET DS  AL4                     SOURCE ALET
TARGETALET DS  AL4                     TARGET ALET
DSP1ALET DS    AL4                     ALET FOR DATA SPACE 1
DSP2ALET DS    AL4                     ALET FOR DATA SPACE 2
DSPTOKEN1  DS  D                       DATA SPACE TOKEN 1
DATASPACESTART1 DS AL4                 DATA SPACE ORIGN 1
DSPTOKEN2  DS  D                       DATA SPACE TOKEN 2
DATASPACESTART2 DS AL4                 DATA SPACE ORIGN 2
VRLAREA  DS    CL(VRLLEN)              VRL
         IARVSERV PLISTVER=MAX,                                      X
               MF=(L,IARVSRV)
         DSPSERV PLISTVER=MAX,                                       X
               MF=(L,DSP1)
         DSPSERV PLISTVER=MAX,                                       X
               MF=(L,DSP2)
ALETA1   DS    CL(ALSRVLLN)            ALSERV MACRO AREA 1
ALETA2   DS    CL(ALSRVLLN)            ALSERV MACRO AREA 2
SDUMPXA  DS    CL(SDUMPLEN)            SDUMPX AREA
MDESETX  DS    CL(MSETLEN1)            MODESET AREA
WORKALEN EQU   *-WORKAREA              WORK AREA LENGTH
         TITLE 'INPUT PARM'
INPPARM  DSECT ,
PARMLEN  DS    XL2                     PARM LENGTH
OPTION   DS    C                       INPUT OPTION
         TITLE 'SHARED SOURCE AREA'
SHAREDSOURCEAREA DSECT
SHAREDAREA     DS   CL(4Ø96*2)         SHARED AREA
SHAREDAREALEN  EQU  *-SHAREDAREA       SHARED AREA LENGTH
         TITLE 'DSPLIST AREA'
DLISTARA DS    CL(DLISTLEN)            SDUMPX DSPLIST AREA
         TITLE 'RSM VIRTUAL RANGE LIST ENTRY'
         IARVRL
         END   SHAREDPG
```

Rem Perretta
Senior Systems Programmer (UK)

A binary search subroutine

INTRODUCTION

The program explained in this article was developed by me to accelerate the search process in some batch programs which used very large internal data tables. We often have to use data tables in our programs to validate records or to get other information.

Typical processing involves an application program reading records from an input file and, for each one, starting a search process in a resident table. This kind of work can make the process very inefficient when very large files and/or tables are to be processed.

Suppose you have 10,000 records in a file and, for every record, you need to look up information in a 1,000-element table. If you do it sequentially, you must consider the following scenarios:

1 The information for each record exists in the table and, if you are lucky, will be located near the beginning of the table. In this case the search code won't have to be executed very many times to retrieve the information.

2 The information may reside at the end of the table. This is not so good. The search code will be executed many times to retrieve the information.

3 The information does not exist in the table. This is terrible. The search code will execute 1,000 times in order to find that the information is not available.

Another way to look for information residing in a table is to use the 'binary search' method. In this method, the table to be scanned has its 'search key field' sorted in ascending order. Initially, the central element of the table is chosen to start the search process. The information is compared to the 'central element' and, if it is greater than the 'central element', a 'new central element' is established between the 'current central' and the 'last element' of the table. If the information is less than the 'central element', a 'new central element' is established between the 'current element' and the 'first element' of

the table and so on, until the information matches or not. We can say the number of times the search code will be executed to retrieve any information is:

$$n = \frac{\log (x)}{\log (2)}$$

where x is the number of elements in the table.

The worst case to retrieve information in a 1,000 elements table is:

$$n = \frac{\log (1000)}{\log (2)} = 9.965 \Longrightarrow 10$$

which means that the search code will only be executed 10 times!

THE BSEARCH PROGRAM

The BSEARCH program is intended to work as a subroutine, so it can be called by other programming languages like Assembler, COBOL, or PL/I. BSEARCH allows you to search in sequential and binary mode. To use binary mode, the table must be sorted in ascending order.

The main control block in BSEARCH is the work area. The work area is defined by the calling program and is initialized the first time BSEARCH is called. In COBOL, the work area looks like this:

```
01  WORKAREA.
          05  WA-TABPTR     PIC 9(8) COMP.
          05  WA-ROWS       PIC 9(8) COMP.
          05  WA-COLS       PIC 9(8) COMP.
          05  WA-ARGADD     PIC 9(8) COMP.
          05  WA-ARGPOS     PIC 9(4) COMP.
          05  WA-ARGLEN     PIC 9(4) COMP.
          05  WA-RETCD      PIC 9(8) COMP.
          05  WA-INDEX      PIC 9(8) COMP.
          05  WA-MODE       PIC X(1).
          05  WA-FLAGS      PIC X(1).
          05  FILLER        PIC 9(4) COMP.
          05  WA-FELADR     PIC 9(8) COMP.
          05  WA-CELADR     PIC 9(8) COMP.
          05  WA-LELADR     PIC 9(8) COMP.
          05  WA-CLCINST    PIC X(6).
          05  FILLER        PIC 9(4) COMP.
          05  WA-WROWS      PIC 9(8) COMP.
          05  WA-WCOLS      PIC 9(8) COMP.
```

To call BSEARCH you must provide a parameter list like this:

```
77  ARG             PIC X(8).
77  MODUS           PIC X(3) VALUE 'BIN'.
01  ARGPOS          PIC 9(4) COMP.
01  ARGLEN          PIC 9(4) COMP.
```

Before you call BSEARCH for the first time, you must put some values into certain fields in the work area:

1 Move the search mode to WA-MODE field.

```
wa-mode=0 ---> sequential search
wa-mode=1 ---> binary search
```

2 Move the number of table elements to WA-ROWS field.

3 Move the length of table element to WA-COLS field.

4 Call BSEARCH for the first time in order to initialize the work area:

```
CALL    'BSEARCH' USING WORKAREA TABLE-NAME.
```

where TABLE-NAME is the name of the table you are working on. Now you are ready to call BSEARCH in order to search but remember that the table must be sorted in ascending order if you wish to search in binary mode.

Now you must provide the parameters' values:

1 Set the search mode moving BIN or SEQ to the MODUS field. The MODUS field is a parameter.

2 Move the argument position to ARGPOS field.

3 Move the argument length to ARGLEN field.

4 Move the argument value to ARG field.

5 Call BSEARCH:

```
call    'bsearch' using workarea
                        arg
                        argpos
                        arglen
                        modus.
```

A new search can be made by repeating instructions 2 to 4.

Notes

WA-INDEX in the work area will hold the number of the table element relative to zero (ie WA-INDEX=0 points to element number 1,WA-INDEX=5 points to element number 6 and so on).

SOURCE CODE FOR BSEARCH

```
*----    THIS ROUTINE SETS THE RETURN CODE FIELD AND THE ELEMENT  ----*
*        INDEX AS RESULT OF THE SEARCH AS FOLLOW:                      *
*        RETURN CODE     ELEMENT INDEX CONTAINS   IT MEANS             *
*         00000000       ELEMENT NUMBER           ARGUMENT FOUND       *
*         00000008       ZEROS                    ARGUMENT NOT FOUND   *
*        THE ELEMENT INDEX IS THE INDEX TO GIVE THE CALLER THE         *
*        ABILITY TO ACCESS THE MATRIX ELEMENT. SO, THE CALLER MUST     *
*        USE THE ELEMENT INDEX TO GET THE MATRIX ELEMENT HE/SHE WANTS  *
*        TO WORK.                                                      *
*        WARNING:                                                      *
*        THERE IS NO LOCAL SAVE AREA IN THIS PROGRAM SINCE IT DOES NOT*
*----    CALL ANOTHER PROGRAM.                                    ----*
BSEARCH  CSECT
BSEARCH  AMODE 31
BSEARCH  RMODE ANY
         STM   14,12,12(13)            . SAVE ALL REGISTERS
         LR    12,15                   . SET BASE ENTRY POINT ADDRESS
         USING BSEARCH,12              . PROGRAM ADDRESSABILITY
         L     11,Ø(1)                 . R11 POINTS TO WORK AREA
         USING WA,11                   . WORK AREA ADDRESSABILITY
*----    AT FIRST TIME, THIS CODE MOVES THE MATRIX ADDRESS        ----*
*        TO WORK AREA AND ENABLES THE BRANCH INSTRUCTION AT            *
*        LABEL "FIRST" TO BRANCH TO "SECOND" AT SECOND TIME            *
*        THE ROUTINE IS CALLED.                                        *
*        INPUT PARAMETERS:  R1 ----> FULL1 : POINTER TO WORK AREA      *
*----                               FULL2 : POINTER TO MATRIX    ----*
         TM    WAFLAGS,WATWICE         . IF NOT THE FIRST TIME
         BO    SECOND                  . THEN SKIP THIS CODE, ELSE
         OI    WAFLAGS,WATWICE         . SET FOR SECOND TIME
         L     2,4(1)                  . GET MATRIX ADDRESS
         L     1,Ø(1)                  . POINT TO WORK AREA
         ST    2,Ø(1)                  . STORE MATRIX ADDRESS IN W/A
         LM    14,12,12(13)            . RESTORE ALL REGISTERS
         BSM   Ø,14
*----  - AT SECOND TIME, PERFORM NORMAL PROCCESSING.              ----*
*      - FIRST, MOVES THE CALLER SPECIFIED VALUES TO WORK AREA,        *
*        RESETS THE RETURN CODE VALUE AND ELEMENT INDEX.               *
*      - PREPARES THE COMPARE INSTRUCTION TO WORK WITH THE            *
*        REQUIRED VALUES OF ARGUMENT POSITION AND ARGUMENT LENGTH.     *
*      - ESTABLISHES POINTERS TO ARGUMENT, MATRIX, AND SETS THE       *
*        ARGUMENT LENGTH OF COMPARE INSTRUCTION.                       *
*      - SAVES NUMBER OF COLUMNS AND ROWS IN WORK FIELD.              *
*      - COMPUTES THE LAST ELEMENT ADDRESS.                           *
```

```
*      - NEXT STEP, TESTS THE SEARCH MODE AND BRANCHES TO THE         *
*        REQUIRED SEARCH CODE.                                        *
*        INPUT PARAMETERS:  R1 ----> FULL1: POINTER TO WORK AREA      *
*                                    FULL2: POINTER TO ARGUMENT FIELD *
*                                    FULL3: POINTER TO ARG POSITION   *
*                                    FULL4: POINTER TO ARG LENGTH     *
*                                    FULL5: POINTER TO MODE FIELD     *
*        WORK AREA:                                                   *
*        FOR A DETAILED VIEW OF WORK AREA, SEE THE "WA" DUMMY SECTION *
*        AT END OF THIS ASSEMBLY.                                     *
*        OUTPUT -------> OUTPUT FROM THIS ROUTINE IS A CONDITION      *
*                        CODE IN GENERAL REGISTER 15 AND IN THE       *
*                        WORK AREA RETURN CODE FIELD "WARC".          *
*                        IF RC=ØØ : "WAINDEX" FIELD IN THE WORK AREA  *
*                        CONTAINS THE NUMBER OF THE TABLE ELEMENT     *
*                        WHERE THE ARGUMENT WAS FOUND.                *
*                        IF RC NOT ØØ : THE ARGUMENT WAS NOT FOUND.   *
*                        IN THIS CASE, THE NUMBER IN THE "WAINDEX"    *
*----                    FIELD IS INVALID AND MUST NOT BE USED.   ----*
SECOND   EQU   *
*        MOVE PARAMETERS VALUES TO WORK AREA
         XC    WAINTER(2),WAINTER
         L     2,4(1)                  . GET ARGUMENT ADDRESS
         ST    2,WAARG                 . STORE IT ON W/A
         L     2,8(1)                  . GET POSITION FIELD ADDRESS
         LH    2,Ø(2)                  . GET POSITION VALUE
         STH   2,WAARGPOS              . STORE IT ON W/A
         L     2,12(1)                 . GET LENGTH FIELD ADDRESS
         LH    2,Ø(2)                  . GET LENGTH VALUE
         STH   2,WAARGLEN              . STORE IT ON W/A
         L     2,16(1)                 . GET MODE FIELD ADDRESS
         IC    2,Ø(2)                  . GET SEARCH MODE
         STC   2,WAMODE                . STORE IT ON W/A
         XC    WARC(4),WARC            . RESET RETURN CODE
         XC    WAINDEX(4),WAINDEX      . RESET ELEMENT INDEX
*        PREPARES COMPARE INSTRUCTION TO WORK WITH THE REQUIRED
*        ARGUMENT POSITION AND ARGUMENT LENGTH.
         MVC   WACLC(6),COMPARE        . MOVE CLC INSTRUCTION TO W/A
         LH    1,WAARGPOS              . LOAD ARGUMENT POSITION
         BCTR  1,Ø                     . TRANSFORM IT IN OFFSET
         ICM   2,3,WACLCOP2            . 2ND OPERAND BASE AND DISPL
         SRL   2,12                    . CLEAR OLD DISPLACEMENT
         SLL   2,12                    . RETURN
         OR    2,1                     . INSERT NEW DISPLACEMENT
         STCM  2,3,WACLCOP2            . STORE 2ND OPERAND DISPL
*        ESSTABLISHES POINTERS TO ARGUMENT, MATRIX AND SET THE
*        ARGUMENT LENGTH TO COMPARE INSTRUCTION
         LH    7,WAARGLEN              . LOAD ARGUMENT LENGTH
         BCTR  7,Ø                     . LENGTH ADJUSTMENT
         L     8,WAARG                 . LOAD THE ARGUMENT ADDRESS
*        SAVE COLLUNMS AND ROWS INTO A WORK FIELD
         MVC   WA#CL(4),WA#COL         . SAVE THE NUMBER OF COLUMNS
```

```
         MVC   WA#RW(4),WA#ROW         . SAVE THE NUMBER OF ROWS
*        COMPUTE LAST ELEMENT ADDRESS
         L     1,WA#RW                 . LOAD NUMBER OF ROWS
         MH    1,WA#CL+2               . MULTIPLY BY NUMBER OF COLUMNS
         A     1,WAMATRIX              . BYTE BEYOND LAST ELEMENT
         S     1,WA#CL                 . LAST ELEMENT ADDRESS
         ST    1,WALAST                . SAVE LAST ELEMENT ADDRESS
         MVC   WAFIRST(4),WAMATRIX     . SET FIRST ELEMENT
*        BRANCH TO REQUIRED SEARCH ROUTINE
         TM    WAMODE,WABIN            . IF BINARY MODE
         BO    BINARY                  . THEN BRANCH
*----    SEQUENTIAL SEARCH MODE                                          *
SEQUENT  EQU   *
         L     9,WAMATRIX              . POINT TO FIRST ELEMENT
         L     4,WA#ROW                . LOAD NUMBER OF ROWS
SEQLOOP  EQU   *
         LH    1,WAINTER
         LA    1,1(,1)
         STH   1,WAINTER
         EX    7,WACLC                 . COMPARE
         BE    BINFOUND                . BRANCH IF OK
         A     9,WA#COL                . POINT TO NEXT ELEMENT
         BCT   4,SEQLOOP               . VERIFY THE NEXT ELEMENT
         B     NOTFOUND
*----    BINARY SEARCH MODE                                          ----*
*        IN THIS CODE SEGMENT, THE FOLLOWING REGISTERS ARE USED          *
*        AS DESCRIBED:                                                   *
*        R7 - CONTAINS THE LENGTH OF CLC INSTRUCTION OPERANDS.           *
*        R8 - IS THE POINTER TO ARGUMENT FIELD                           *
*----    R9 - IS THE POINTER TO CENTRAL ELEMENT                      ----*
BINARY   EQU   *
*        THIS CODE COMPUTES THE ADDRESS OF CENTRAL ELEMENT IN
*        A GIVEN SEGMENT OF CURRENT MATRIX.
         SLR   4,4                     . CLEAR R4
         L     5,WA#RW                 . LOAD NUMBER OF ROWS
         LA    2,2                     . LOAD DIVISOR
         DR    4,2                     . DIVIDE ROW BY 2
         MH    5,WA#CL+2               . NUMBER OF BYTES TO CENTRE
         A     5,WAFIRST               . EFFECTIVE CENTRE ADDRESS
         ST    5,WACENTER              . SAVE CENTER ADDRESS
*        COMPARE THE ARGUMENT AGAINST THE MATRIX POSITION
         LH    1,WAINTER
         LA    1,1(,1)
         STH   1,WAINTER
         L     9,WACENTER              . LOAD ELEMENT ADDRESS
         EX    7,WACLC                 . COMPARE
         BE    BINFOUND                . IF EQUAL THEN GOBACK
         BH    BINHI                   . IF GREATER
*        IF ARGUMENT IS LESS THAN THAT IN THE MATRIX
*        THEN SET THE NEW LAST ELEMENT
BINLO    EQU   *
         L     1,WACENTER              . LOAD CENTRE ADDRESS
```

```
         S     1,WA#CL                  . POINT TO PREVIOUS ELEMENT
         ST    1,WALAST                 . SET IT AS THE LAST ELEMENT
         B     BINVROW
*        WHEN THE ARGUMENT IS GREATER THAN THAT IN THE MATRIX
*        THEN SET CENTRAL ELEMENT AS THE NEW FIRST ELEMENT
BINHI    EQU   *
         MVC   WAFIRST(4),WACENTER      . MAKE CENTRAL = FIRST
*        COMPUTE THE NEW NUMBER OF ROWS.
BINVROW  EQU   *
         L     5,WALAST                 . LOAD LAST ELEMENT
         S     5,WAFIRST                . COMPUTE NUMBER OF BYTES
         A     5,WA#CL                  . ADD ONE MORE ELEMENT
         SLR   4,4                      . CLEAR R4
         L     2,WA#CL                  . LOAD NUMBER OF COLUMNS
         DR    4,2                      . COMPUTE ROWS
         CH    5,=H'2'                  . IF 2 REMAINS, THEN
         BE    BIN1LST                  .
         ST    5,WA#RW                  . SAVE NEW NUMBER OF ROWS
         B     BINARY                   . RESTART
*        COMPARES THE TWO LAST ROWS TO SEE THEY MATCH
*        SEARCH REQUIREMENTS.
BIN1LST  EQU   *
         LH    1,WAINTER
         LA    1,1(,1)
         STH   1,WAINTER
         L     9,WAFIRST                . POINT TO FIRST ELEMENT
         EX    7,WACLC                  . COMPARE
         BE    BINFOUND                 . IF EQUAL, THEN GOBACK
         LH    1,WAINTER
         LA    1,1(,1)
         STH   1,WAINTER
         L     9,WALAST                 . POINT TO LAST ELEMENT
         EX    7,WACLC                  . COMPARE
         BE    BINFOUND                 . IF EQUAL, THEN GOBACK
*        IF THE TWO LAST ROWS DONT MATCH THE SEARCH REQUIREMENTS
*        THEN CLEAR THE ELEMENT INDEX, SET RETURN CODE TO 8 AND
*        RETURN TO CALLER
NOTFOUND EQU   *
         LA    15,8                     . SET RC=00000008
         ST    15,WARC                  . STORE IT IN W/A
         XC    WAINDEX(4),WAINDEX       . SET ELEMENT INDEX TO ZEROS
         LM    14,12,12(13)             . RESTORE ALL REGISTERS
         BSM   0,14                     . RETURN TO CALLER
*        IF ONE OF THE TWO LAST ROWS MATCHES THE SEARCH,
*        THEN LOOK FOR THE ELEMENT INDEX IT REPRESENTS,
*        STORE ITS VALUE IN THE ELEMENT INDEX FIELD, SET
*        RETURN CODE TO ZEROS, AND RETURN TO CALLER.
BINFOUND EQU   *
         SLR   8,8                      . CLEAR R8
         S     9,WAMATRIX               . CURRENT - FIRST
         L     2,WA#COL                 . LOAD NUMBER OF COLUMNS
         DR    8,2                      . COMPUTE THE ELEMENT INDEX
```

```
         ST    9,WAINDEX               . STORE ELEMENT INDEX IN W/A
         SLR   15,15                   . SET RC=00000000
         ST    15,WARC                 . STORE IT IN W/A
         LM    14,12,12(13)            . RESTORE ALL REGISTERS
         SLR   15,15                   . SET RC=00000000
         BSM   Ø,14                    . RETURN TO CALLER
*        OUT OF SEQUENCE COMPARE INSTRUCTION
         DS    ØH
COMPARE  CLC   Ø(Ø,8),Ø(9)             . D5LLBDDDBDDD
         LTORG
*----    WORK AREA DUMMY SECTION                                        *
WA       DSECT
WAMATRIX DS    A           BIN      +ØØ. POINTER TO MATRIX
WA#ROW   DS    A           BIN      +Ø4. MATRIX NUMBER OF ROWS
WA#COL   DS    A           BIN      +Ø8. MATRIX NUMBER OF COLUMNS
WAARG    DS    A           BIN      +12. POINTER TO ARGUMENT FIELD
WAARGPOS DS    AL2         BIN      +16. ARGUMENT FIELD POSITION
WAARGLEN DS    AL2         BIN      +18. ARGUMENT FIELD LENGTH
WARC     DS    A           BIN      +2Ø. RETURN CODE
WAINDEX  DS    A           BIN      +24. ELEMENT INDEX
WAMODE   DS    AL1         BIN      +28. SEARCH MODE
WASEQ    EQU   X'ØØ'                  .   SEQUENTIAL SEARCH MODE
WABIN    EQU   X'Ø1'                  .   BINARY SEARCH MODE
WAFLAGS  DS    AL1         BIN      +29. FLAGS
WAONCE   EQU   X'ØØ'                  .   FIRST TIME CALLED
WATWICE  EQU   X'Ø1'                  .   CALLED MORE THAN 1 TIME
         DS    AL2                  +3Ø. RESERVED
WAFIRST  DS    A           BIN      +32. FIRST   ELEMENT ADDRESS
WACENTER DS    A           BIN      +36. CENTRAL ELEMENT ADDRESS
WALAST   DS    A           BIN      +4Ø. LAST    ELEMENT ADDRESS
WACLC    DS    CL6                  +44. COMPARE INSTRUCTION
         ORG   WACLC
         DS    CL1             OP   +44. INSTRUCTION CODE
WACLCLEN DS    CL1             LL   +45. OPERAND'S LENGTH
WACLCOP1 DS    CL2             BDDD +46. FIRST OPERAND
WACLCOP2 DS    CL2             BDDD +48. SECOND OPERAND
         ORG
WAINTER  DS    CL2                  +5Ø. RESERVED
WA#RW    DS    A                    +52. NUMBER OF ROWS    - WORK
WA#CL    DS    A                    +56. NUMBER OF COLUMNS - WORK
         END
```

COBOL EXAMPLE

This COBOL example was tested under MVS/ESA 4.3.

```
 ID DIVISION.
         .
 DATA DIVISION.
 WORKING-STORAGE SECTION.
         .
```

```
*******************************************
* WORK AREA TO BE USED BY THE SEARCH ROUTINE*
*******************************************
       Ø1  WORKAREA.
           Ø5  WA-TABPTR    PIC 9(8) COMP.
           Ø5  WA-ROWS      PIC 9(8) COMP.
           Ø5  WA-COLS      PIC 9(8) COMP.
           Ø5  WA-ARGADD    PIC 9(8) COMP.
           Ø5  WA-ARGPOS    PIC 9(4) COMP.
           Ø5  WA-ARGLEN    PIC 9(4) COMP.
           Ø5  WA-RETCD     PIC 9(8) COMP.
           Ø5  WA-INDEX     PIC 9(8) COMP.
           Ø5  WA-MODE      PIC X(1).
           Ø5  WA-FLAGS     PIC X(1).
           Ø5  FILLER       PIC 9(4) COMP.
           Ø5  WA-FELADR    PIC 9(8) COMP.
           Ø5  WA-CELADR    PIC 9(8) COMP.
           Ø5  WA-LELADR    PIC 9(8) COMP.
           Ø5  WA-CLCINST   PIC X(6).
           Ø5  FILLER       PIC 9(4) COMP.
           Ø5  WA-WROWS     PIC 9(8) COMP.
           Ø5  WA-WCOLS     PIC 9(8) COMP.
*******************************************
* PARAMETERS FIELDS                       *
*******************************************
       77  ARG             PIC X(8).
       77  MODUS           PIC X(3) VALUE 'BIN'.
       Ø1  ARGPOS           PIC 9(4) COMP.
       Ø1  ARGLEN           PIC 9(4) COMP.
*******************************************
* TABLE DEFINITION                        *
*******************************************
       Ø1  ASM-INSTR.
          Ø5  FILLER PIC X(8) VALUE 'A       '.
          Ø5  FILLER PIC X(8) VALUE 'ACTR    '.
          Ø5  FILLER PIC X(8) VALUE 'AD      '.
          Ø5  FILLER PIC X(8) VALUE 'ADR     '.
          Ø5  FILLER PIC X(8) VALUE 'AE      '.
          Ø5  FILLER PIC X(8) VALUE 'AEJECT  '.
          Ø5  FILLER PIC X(8) VALUE 'AER     '.
          Ø5  FILLER PIC X(8) VALUE 'AGO     '.
          Ø5  FILLER PIC X(8) VALUE 'AH      '.
          Ø5  FILLER PIC X(8) VALUE 'AIF     '.
       Ø1  ASM-CODES REDEFINES ASM-INSTR.
          Ø5  INSTR     PIC X(8) OCCURS 1Ø TIMES.
       PROCEDURE DIVISION.
*******************************************
* THE FIRST CALL INITIALIZES THE WORK AREA. *
*******************************************
           CALL    'BSEARCH' USING WORKAREA
                                   ASM-INSTR.
```

```
*******************************************
* YOU MUST MOVE SOME DATA TO WORK AREA.   *
*******************************************
          MOVE     1       TO     WA-MODE.   (SEARCH MODE=BIN)
          MOVE    1Ø       TO     WA-ROWS.   (# OF ROWS)
          MOVE     8       TO     WA-COLS.   (# OF COLS)
*******************************************
* NOW YOU PROVIDE THE PARAMATERS VALUES.  *
*******************************************
          MOVE 'BIN'       TO     MODUS.     (SEARCH MODE=BIN)
          MOVE 'AGO'       TO     ARG.       (ARGUMENT TO SEARCH)
          MOVE     1       TO     ARGPOS.    (ARGUMENT POSITION)
          MOVE     3       TO     ARGLEN.    (ARGUMENT LENGTH)
*******************************************
* CALL SECOND TIME TO SEARCH.             *
*******************************************
          CALL    'BSEARCH' USING WORKAREA
                                  ARG
                                  ARGPOS
                                  ARGLEN
                                  MODUS.
*******************************************
* TEST THE RETURN CODE.                   *
*******************************************
          IF RETURN-CODE EQUAL Ø GO TO I-SHOW ELSE
                   DISPLAY '++ "' ARG '" NOT FOUND ++' UPON CONSOLE
                   GO TO FIM.
*******************************************
* DISPLAY THE SEARCH RESULT AND GOBACK.   *
*******************************************
       I-SHOW.
           ADD 1 TO WA-INDEX.
           DISPLAY 'VALUE FOUND=' INSTR(WA-INDEX) UPON CONSOLE.
       FIM.
           GOBACK.
```

Antonio Spinelli
Systems Programmer
Prodesp (Brazil)

BGS Systems Inc has announced Release 12.9 of BEST/1 Performance Assurance, its MVS performance management and modelling tool. Support for goal mode systems, modelling of tape devices, and reporting for SNA networks and Tandem systems has been added to the new version. There is a facility for goal mode users to create models of goal mode systems automatically. Users set a switch to create a model complete with workloads for every service class/period then use the 'what-if' functions of BEST/1 Datacenter for hardware planning and performance tuning. The new release also allows users to incorporate the activity of individual tape devices and their contribution to workload performance into models and predict the effect of installing faster tape devices.

For further information contact:
BGS Systems Inc, 128 Technology Center, Waltham, MA 02254-9111, USA
Tel: (617) 891 0000
Fax: (617) 890 0000 or
BGS Systems Ltd, Bridge Gate, 55-57 High Street, Redhill, RH1 1RX, UK
Tel: (01737) 778400
Fax: (01737) 779060.

* * *

Version 5.0 of Chicago Soft Ltd's MVS/QuickRef, ISPF-based quick reference tool has been hugely updated and now includes messages from the following third-party products: LMS (Sutnym Storage); SuperUtilities (CDB Software); BLOCKADE for MVS and BLOCKADE Enterprise Security Server (Blockade Systems); CA-GOALNET, CA-TELEVIEW, CA-EASYTRIEVE, and CA-Inter Test (Computer Associates); ENTERWEB (Macro 4); Naviplex (Landmark Systems); CenterStage/MVS and Quick Tune (Softworks); BETA 42 and BETA 45 (Beta Systems); TAPE2000 (SEA); Check Plus for DB2, PACLOG, PATROL DB-Log Master for DB2 for MVS, Reorg Plus for DB2, Coordinated Recovery Manager, Recovery Manager for IMS, and Authorization Interface Utility (BMC Software); and all products from Chaney Systems. A complete description of the syntax and usage rules for each element of HTML has also been added to the database.

For further information contact:
Chicago Soft Ltd, 45 Lyme Road #307, Hanover, NH 03755- 9867, USA
Tel: (603) 643 4002
Fax: (603) 643 4571 or
Tecfacs Ltd, 6 Forest Court, Oaklands Park, Wokingham, Berks, RG11 2FD, UK
Tel: (01734) 776645
Fax: (01734) 894461.

* * *

Legacy Tuning Products llc has added ISPF panels to Version 2.02 of JCLTune, its automated JCL analysis, reporting, and tuning tool. JCLTune captures SMF information, which it processes to determine how to modify JCL for better performance.

For further information Contact:
Legacy Tuning Products llc, 4061 Powder Mill Road, Suite 500, Calverton, MD 20705, USA
Tel: (301) 902 0355
Fax: (301) 902 0333.

xephon