170 MVS

November 2000

update

In this issue

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

North American office
Xephon/QNA
PO Box 350100,
Westminster, CO 80035-0100
USA
Telephone: (303) 410 9344
Fax: (303) 438 0290

Contributions
Articles published in *MVS Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*, or you can download a copy from www.xephon.com/contnote.html.

Editor
Jaime Kaminski

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

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

Subscriptions and back-issues
A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Printed in England.

z900 and z/OS – some considerations

INTRODUCTION

IBM's October 2000 announcement of the eServer branding, the z900 processor, and usage-based pricing is possibly the most significant development since IBM unveiled the System/390 a decade ago. The announcements exactly match the predictions made in the article *OS/390 strategy overview* in *MVS Update* issue 167, August 2000. This can be used as an introduction to the strategic implications of the new processors and operating systems.

THE NEW NOMENCLATURE

The first part of the announcement was the rebranding of the entire IBM server range. The new name given to all IBM server types is the 'eServer'. This is then followed by the range type, so the Netfinity PCs, based on Intel chips, become the eServer xSeries, with the 'X' representing the 'X-Architecture'. The AS/400 becomes the iSeries, 'I' representing 'integrated'. The RS/6000 becomes the pSeries, with the 'P' representing 'performance', and the System/390 becomes the zSeries, with the 'Z' indicating 'zero downtime'. In the same way that the System/390 is called the S/390 (or 390) in common parlance, the zSeries 900 will simply be the z900. The operating systems will follow the same nomenclature with OS/390 becoming z/OS, running the z/Architecture.

The obvious importance of the new naming structure is the creation of a unified IBM server brand, rather than the disparate and confusing naming structure that has resulted from the diverse heritage of the IBM servers. This rebranding is such a significant goal that IBM expects to spend $325 million promoting the eServer line in the next five quarters. The rebranding will be important in the acquisition of market share in the high-end e-business server market. However, the legal challenge to IBM's use of the eServer name by Technauts of North Carolina may result in some modification to the name, although this is unlikely.

THE Z900

The first new server to be announced under the eServer badge was the zSeries 900 (code-named the Freeway), the successor to the System/390 . The new z/Architecture is the end result of a two-year, one billion dollar project to re-orient the System/390 architecture towards the needs of e-business and e-commerce.

The principal difference between the System/390 and z900 is the latter's use of 64-bit addressing. The new z/Architecture also has a massively improved I/O subsystem to support the increased number of processors and larger main memory. With an I/O bandwidth of 24 gigabits per second, this is triple the G6 figure. The z900 also has significant enhancements to TCP/IP communications through the use of high speed interconnects called 'HiperSockets'. This allows TCP/IP traffic to travel between partitions within a single z900 at memory speed rather than network speed, creating a network in a box. The new Gigabit Ethernet feature also achieves line speed of one gigabit per second. The end result is a massive increase in the speed of communication within the server, between servers, and to the end users.

Another new feature of the z/Architecture is the Intelligent Resource Director (IRD). This controls LPAR CPU management, dynamic channel path management, and channel subsystems priority queuing. In essence the IRD manages the dynamic rerouting of CPU resources and I/O paths, and prioritizes queueing so that these resources are automatically directed where they are needed most. And self-healing capabilities mean the system continuously monitors itself and can automatically detect and correct errors. It does this through the Internet by dialing up and checking in to an IBM technology centre at specified times. Hardware and software capacity can be upgraded on-the-fly with a click of the mouse.

There is native FICON attachment to Magstar tape, and FICON directors for Magstar tape and the IBM Enterprise Storage Server. New and enhanced capacity-on-demand capabilities enable the system to scale from one to 16 processors on-the-fly and without disruption. It also provides 2,000 SSL transactions per second.

The new family of servers covers 26 CMOS models, ranging from one

way to 16 way, utilizing from three to twenty processors. These are available as new systems or as upgrades from Generation 5 and 6 models. The System/390 coupling facility is upgradeable to the z900 as well.

At the heart of the z900 is the IBM multichip module (MCM), which contains 35 chips, one clock, four memory bus adapters, two system controllers, and eight L2 cache devices, copper and silicon on insulator technology, and 2.5 billion transistors. The new boxes have increased the maximum number of central processors in the symmetrical processing complex from twelve in the G6 to sixteen in the z900. It promises greater than 2,500 MIPs on 16 processors, and 300 million transactions a day (9 billion when clustered).

The new boxes have increased the maximum number of central processors in the symmetrical processing complex from twelve in the G6 to sixteen in the z900. IBM claims this will provide a 50-60% performance increase over the G6 model ZZ7, whilst performance increases in the order of 20-30% are attainable at the uniprocessor level.

64-BIT ARCHITECTURE

The 64-bit Generation 7 will mark as big a change in the mainframe world as the 3081 did twenty years ago. Currently, most 64-bit processors are found in high-end Unix enterprise servers such as Compaq's Alpha, Intel's IA-64, IBM's PowerPC, HP's PA-RISC, and Sun's UltraSPARC.

E-business is the largest driver for 64-bit computing because of its huge long-term requirements for bandwidth and capacity. One of the most important scaling factors in a system is the systems addressing architecture. A 32-bit system can address 4GB of data. A 64-bit system can address 4 billion times that amount.

Applications that are compute-intensive or require high levels of data manipulation (such as certain Internet applications, data warehouses, and concurrent transaction processing) are driving the move to 64-bit addressing. Oracle, Sybase, and Informix databases among others now support it.

Z/OS

The z900 will have its own 64-bit operating system called z/OS. OS/390 Version 2 Release 10 was released on 29 September 2000 and marks the end of the OS/390 line. The new z900 box will be able to support operating systems as far back as OS/390 Version 2 Release 6 in 31-bit mode and z/OS can run on G5 and G6 machines in 31-bit mode. This is because the z/Architecture is a tri-modal architecture capable of executing in 24-bit, 31-bit, or 64-bit addressing modes. Operating systems and middleware products have been modified to exploit the new capabilities of the z/Architecture. Users will gain benefits immediately by the elimination of the overhead of central storage to expanded storage page movement and the relief provided for those constrained by the 2GB real storage limitations.

Users have two possible upgrade paths to 64-bit operations. They can either use OS/390 Version 2 Release 10 and upgrade this with a Product Upgrade Package (PUP), which will take OS/390 up to 64-bit operations. Alternatively, they can go straight to z/OS Version 1 Release 1 (which would be the equivalent of OS/390 Version 3 Release 11). Both the PUP and z/OS will be available in March 2001, following the standard six-month release cycle of OS/390.

64-bit addressing provides considerable benefits for z/OS. These can be divided into improvements to real memory, integer arithmetic, and virtual memory (see *MVS Update*, issue 167, August 2000, page 6). The implementation of 64-bit z/Architecture eliminates any bottlenecks associated with a lack of addressable memory by making the addressing capability virtually unlimited (16 exabytes compared to the current capability of 2 gigabytes). Both DB2 Version 6 (with PTF) and IMS Version 7 are enhanced to exploit 64-bit real storage above 2 GB. Additionally, access methods such as BSAM, QSAM, etc, VSAM for extended format datasets, Hierarchical File System (HFS), and Extended Remote Copy (XRC) have been enhanced to exploit 64-bit real storage above 2GB.

The highest level of compliance to the 64-bit specifications is achieved when applications are written for a 64-bit address space and take advantage of the performance boost provided by the underlying hardware and operating system. Both DB2 Version 6 (with PTF) and IMS Version 7 are enhanced to exploit 64-bit real storage above 2GB.

However, at the present it is unlikely that many enterprises will require the boost in performance that this will entail.

z/OS also has an installation wizard designed to reduce the skill requirements needed for set-up. This follows the trend in OS/390 of increasing the ease of use.

TCP/IP NETWORKING ENHANCEMENTS

z/OS can provide near continuous availability for TCP/IP applications and their users with two key features in z/OS: Sysplex Distributor and VIPA Non-disruptive Takeover.

Virtual IP Address Non-disruptive Takeover

VIPA represents an IP address that is not tied to a specific hardware adapter address.The benefit is that, if an adapter fails, the IP protocol can find an alternate path to the same software, be it the TCP/IP services on z900 or an application. VIPA Takeover introduced in OS/390 Release 8 supports movement to a backup IP stack on a different server in a Parallel Sysplex in case of a failure of the primary IP stack.VIPA Non-disruptive Takeover enhances the initial Release 8 functions, providing VIPA takeback support. This allows the movement of workload back from the alternate to the primary IP stack.

Sysplex Distributor

Introduced in OS/390 R10,Sypslex Distributor is a software-only means of distributing IP workload across a Parallel Sypslex cluster. Client connections appear to be connected to a single IP address yet the connections are outed to servers on different z900 or S/390 servers.In addition to load balancing, Sysplex Distributor simplifies the task of moving applications within a Parallel Sysplex environment.

WORKLOAD LICENSE CHARGES (WLC)

A key part of the IBM announcement is the forthcoming availability of Workload License Charges (Usage Based Pricing) with the z900. This is a crucial part of the z900 package. Users have long complained

that the current pricing models penalize those with unused capacity or those wishing to migrate large non-System/390 applications to the mainframe. Capacity-based pricing models force them to pay for software based on the size of their overall mainframe complex rather than on its actual use. These high up-front software costs have been detrimental to IBM because they have prevented the company from significantly expanding its user base. The new pricing model means the cost will be based on the number of MIPS consumed. There are in fact several different pricing models now available:

- *Full server capacity* – a monthly licence fee based on the total capacity of the server.
- *Defined capacity* – a monthly licence fee based on the defined capacity within one or more logical partitions.
- *Flat pricing* – a monthly licence fee charged at a flat rate, regardless of the server size or use.
- *One time charges* – upfront licence payment based on the number of users or the numbr of processors in the server, followed by an annual maintenance contract.

There are considerable benefits with Workload License Charges for e-businesses. Users can grow their business one workload at a time, if they do encounter a spike of activity. It will not affect their bill, and it is possible to purchase insurance for e-business surges.

The key consideration for users is that metering will require both hardware and software support. Users wishing to benefit from these changes will require a z900 system running z/OS in 64-bit mode. The Licence Manager component of z/OS deals with the Workload License Charges.

Usage-based pricing or 'Software Value Pricing' will be of considerable benefit to users who will be able to pay for the capacity used, not the total system capacity. The financial benefits to users of usage-based pricing will be enormous and will probably mean that there will be a rapid move to the new processor.

However, the overall effectiveness of the model is based on the ISVs being willing to adopt the pricing scheme. The new licensing is

immediately supported by BMC, Candle, Computer Associates, Compuware, Isogon, SAGA, and Software AG. IBM appears to be positioning its own products to provide direct alternatives to these ISV products should they refuse to bring their pricing into line.

Other support comes from CRM and ERP vendors PeopleSoft, SAP, and Siebel, plus Lawson, IMI, Temenos, and Trilogy. And the z900 supports Linux off the shelf, immediately running thousands of Linux applications. Also, Rogue Wave and Rational are among the new software companies announcing specific Linux products for the z900.

This contrasts with Amdahl, which has been offering a similar capability with its Multiple Server Facility for over two years. But so far, few of the major independent software vendors have changed their pricing practices to complement this.

The pricing model (in conjunction with IBM's Java, Linux, and Unix initiatives) provides further indications that IBM sees the future of the System/390 platform as the basis for applications rather than as a stand-alone operating system.

It is likely that the usage-based pricing model will spread in the enerprise server marketplace; for example HP announced that it also intends to support a pay-per-use computing model, which it calls 'Utility Computing'. However, unlike IBM's solution, there is uncertainty about exactly how this will work. The exact pricing model is not available and the accounting software to support this is not expected to ship until the first quarter of 2001. The ability to measure application usage will also facilitate more usage-based pricing schemes across IBM in the future.

THE Z900 AND E-BUSINESS

Less than a decade ago IT analysts were predicting the demise of the mainframe. Even by mid- to late 1990, when it became clear that the mainframe would not vanish, analysts still predicted that there would be a slow-down in the rate of mainframe capacity increase towards the end of the century. Again this did not happen and now users are experiencing massive increases in capacity requirements for the future. This is being fuelled primarily by the rapid global move to

24x7 e-commerce and the continued trend today for company mergers and takeovers increasing the size of businesses.

The transition from simple e-business transactions via Web pages to true end-to-end e-business transactions without intervention will massively increase the capacity requirements of machines.

A whole plethora of factors affect whether an e-commerce site is successful or not, but two elements are primarily a hardware issue. These are support for wildly unpredictable user demands and support for secure transactions.

- Unpredictable user demands are the scourge of e-commerce sites. Television or other media advertising can result in massive surges of users – often hundreds of times the daily average – viewing a company's Web pages. Supporting these spikes in demand is a key area of concern for e-commerce sites. The price of failure is the loss of both customers and credibility, which can impact on the bottom line – a company's share price.
- A rarely-mentioned requirement of electronic commerce sites is the ability to support large volumes of secure transactions. Potential customers wishing to purchase items using credit cards require secure transactions – which are supported by the Secure Sockets Layer (SSL). Dealing with large numbers of these secure encrypted transactions can slow down some apparently 'powerful' machines to a crawl.

Over the next couple of years, these increasing demands will dramatically highlight which systems have the scalability to support the new e-business requirements.

It is problems like this that the new large Unix servers are desperately attempting to overcome. These vendors can increase the performance and availability of a system that looks good on paper but is not always sufficient in the real world. Users need to think of how their systems are likely to grow in this unpredictable environment.

Many global organizations, especially e-business companies, require information systems to be in operation 24 hours per day, 365 days per year. Extended downtime can have a disastrous impact on the business.

Even a few minutes of system unavailability (planned or unplanned) may be extremely disruptive and expensive.

THE COMPETITIVE LANDSCAPE

The IBM mainframe announcements should be viewed in a wider perspective. IBM is also reorganizing its sales forces to sell by customer type rather than product. These moves indicate a significant realignment of IBM strategy towards the e-business server marketplace competing against Sun and to a lesser extent HP, rather than Hitachi and Amdahl, IBM's traditional rivals in the mainframe arena.

For years mainframes have been the most scalable, and robust computing solution available to business users. The IBM System/390 architecture has 35 years of heritage to support it. The number of vendors competing for this high-end business user space is increasing. In September 2000 both Hewlett-Packard (HP) and Sun Microsystems announced or released their offerings for this sector. On 12 September HP announced HP 9000 Superdome server, aimed at the top-end of the Web server market, plugging a gap in its product set.

As the hardware offerings from the competing vendors become closer, competitive advantage will have to be sought in other areas, namely the value added services that these companies can provide, and IBM Global Services is the leader in that arena.

CONCLUSIONS

The z900 running z/OS provides the IT industry with massive levels of performance and scalability that will sustain the most strenuous e-commerce demands. Additionally, the environmental and administrative benefits of massive server consolidation could be a crucial point in the favour of the z900. The z900 package provides an unparalleled instrument for conducting both e-business and traditional workloads.

A DASD volume display utility

INTRODUCTION

The following program provides a useful and quick DASD volume display that includes all volumes matching a specified mask (from one to six characters). The display includes the unit number, model type, and, if SMS managed, the SMS storage group name. The program is invoked via the following REXX:

```
/* REXX */

arg parm
if parm = '' then
   exit
parm_length = length(parm)
if parm_length > 6 then
   do
      say parm' is too long !'
      exit
   end
asterisk = pos('*',parm)
if asterisk ¬= Ø then
   if (parm_length > asterisk) | (asterisk = 1) then
      do
         say '"'parm'" is an invalid mask'
         exit
      end
   else
      parm = substr(parm,1,parm_length-1)
"call 'your-dataset-name(VOLINFO)' '"parm"'"
```

VOLINFO

```
VOLINFO  CSECT
VOLINFO  AMODE 31
VOLINFO  RMODE 24
         USING VOLINFO,R15
         B     A_Start
         DC    CL8'VOLINFO '
         DC    CL8'&SYSDATE'
A_Start  DS    ØH
         BAKR  R14,Ø
         LR    R12,R15
         USING VOLINFO,R12
         DROP  R15
```

```
         LA    R13,Save_area
         L     R2,CVTPTR                  r2 -> cvt
         L     R3,CVTSYSAD-CVT(,R2)       r3 -> ucb of iplvol
         USING UCBOB,R3
         MVC   IPL_volume(6),UCBVOLI      save volser of iplvol
         LA    R3,UCB_copy                r3 -> ucb copy area
         L     R1,Ø(,R1)                  r1 -> parm
         SR    R2,R2
         ICM   R2,B'ØØ11',Ø(R1)           length
         BZ    Abend
         CHI   R2,6
         BE    Specific_volume            = 6 characters
         BL    Generic_scan               < 6 characters

Abend    ABEND 99

Generic_scan equ *
         BCTR  R2,Ø                       -1
         EX    R2,MVCØØ1                  volume mask
         EX    R2,MVCØØ2                  put in heading

Generic_scan_loop equ *
         UCBSCAN COPY,UCBAREA=UCB_copy,WORKAREA=UCB_scanwork,          X
               DEVCLASS=DASD,DYNAMIC=YES,                              X
               DCEAREA=DCE_copy,DCELEN=32,RANGE=ALL,DEVNCHAR=Unit_no

         LTR   R15,R15                    finished ?
         BNZ   End_of_scan                yes
         TM    UCBSTAT,UCBONLI            unit on-line ?
         BNO   Generic_scan_loop          no - get next
         EX    R2,CLCØØ1                  this volume matches ?
         BNE   Generic_scan_loop          no - get next
         CLI   H_flag,X'ØØ'
         BNE   Heading_already_done
         MVI   H_flag,X'FF'
         TPUT  Heading,L'Heading

Heading_already_done equ *
         MVC   TPUT_line,C' '
         MVC   TPUT_line+1(L'TPUT_line-1),TPUT_line
         BAL   R8,P_Display_results
         B     Generic_scan_loop

End_of_scan equ *
         CHI   R15,4                      scan finished OK ?
         BNE   Abend
         CLI   H_flag,X'FF'               heading done ?
         BNE   Volume_not_found           no - then not found
         B     Program_exit
MVCØØ1   MVC   Volume_mask(Ø),2(R1)
MVCØØ2   MVC   Heading+27(Ø),Volume_mask
```

```
CLCØØ1   CLC   Volume_mask(Ø),UCBVOLI
Specific_volume equ *
         MVC   Volser(6),2(R1)
         UCBSCAN COPY,UCBAREA=UCB_copy,WORKAREA=UCB_scanwork,          X
               DEVCLASS=DASD,DYNAMIC=YES,VOLSER=Volser,                X
               DCEAREA=DCE_copy,DCELEN=32,RANGE=ALL,DEVNCHAR=Unit_no
         LTR   R15,R15
         BNZ   Check_rc
         BAL   R8,P_Display_results

Program_exit equ *
         SR    R15,R15
         PR                                return to caller

Check_rc equ *
         CHI   R15,4
         BNE   Abend
Volume_not_found equ *
         TPUT  Not_found_msg,L'Not_found_msg
         B     Program_exit

P_Display_results equ *
         MVC   TPUT_line(6),UCBVOLI        volser
         MVC   TPUT_line+7(4),Unit_no      unit address
         CLI   TPUT_line+7,C'Ø'
         BNE   *+8                         skip next
         MVI   TPUT_line+7,C' '            get rid of leading Ø
         MVC   TPUT_line+12(4),=C'339Ø'    assume either 338Ø or 339Ø
         CLI   UCBUNTYP,X'ØF'              type ?
         BE    *+8                         skip next
         MVI   TPUT_line+14,C'8'           338Ø !
         LA    RIØ,DCE_COPY
         USING DCE,RIØ
         LA    R4,DASD_model

P_Search_table_loop equ *
         CLC   DCEOBRDT(1),Ø(R4)           model type
         DROP  RIØ
         BE    P_Found_it
         LA    R4,3(,R4)                   r4 -> next
         CLI   Ø(R4),X'FF'
         BNE   P_Search_table_loop
         B     P_Check_for_SMS

P_Found_it equ *
         MVC   TPUT_line+16(2),1(R4)       model type

P_Check_for_SMS equ *
         TM    UCBFL5,UCBSMS               sms managed ?
         BNO   P_Check_for_IPL_volume
         CLC   SSOB_pointer,=F'Ø'
         BNE   P_setup_done
```

```
         LA    R9,SSIB_area                set up IEFJSSIB
         MVC   SSIBID-SSIB(4,R9),=C'SSIB'
         MVC   SSIBSSNM-SSIB(4,R9),=C'SMS '
         LHI   RØ,SSIBSIZE
         STH   RØ,SSIBLEN-SSIB(,R9)
         LA    R1Ø,SSOB_area               set up IEFSSOBH
         ST    R1Ø,SSOB_pointer
         OI    SSOB_pointer,X'8Ø'
         MVC   SSOBID-SSOB(4,R1Ø),=C'SSOB'
         ST    R9,SSOBSSIB-SSOB(,R1Ø)      SSOB ptr to SSIB
         LHI   RØ,SSOBHSIZ
         STH   RØ,SSOBLEN-SSOB(,R1Ø)
         LHI   RØ,SSOBSSMS                 function code
         STH   RØ,SSOBFUNC-SSOB(,R1Ø)
         LA    R11,SSSA_area               set up IEFSSSA
         USING IEFSSSA,R11
         ST    R11,SSOBINDV-SSOB(,R1Ø)     SSOB ptr to SSSA
         MVC   SSSAID,=CL4'SSSA'
         LHI   RØ,SSSALN+SSSA1LN
         STH   RØ,SSSALEN
         LHI   RØ,SSOBSSVR
         STH   RØ,SSSAVER
         LHI   RØ,SSSAACTV
         STH   RØ,SSSASFN
         MVI   SSSAIFLG,SSSANAUT
         MVI   SSSA1TYP,SSSA1VOL
         LHI   RØ,1
         ST    RØ,SSSA1CNT
         LHI   RØ,6
         STH   RØ,SSSA1NML

P_setup_done equ *
         MVC   SSSA1NAM(6),UCBVOLI         volser

         LA    R1,SSOB_pointer
         IEFSSREQ                          subsystem call

         LTR   R15,R15
         BNZ   Abend

         L     R1Ø,SSSA1PTR
         MVC   TPUT_line+19(8),VLDSTGRP-VLD(R1Ø)  storage group name
         B     P_Issue_TPUT

P_Check_for_IPL_volume equ *
         CLC   UCBVOLI(6),IPL_volume
         BNE   P_Issue_TPUT
         MVC   TPUT_line+19(14),=C'<— IPL volume'

P_Issue_TPUT equ *
```

```
         TPUT  TPUT_line,L'TPUT_line

         BR    R8                          return

Save_area        DC    18F'Ø'

SSOB_pointer     DC    F'Ø'

DASD_model       equ *
                 DC    X'1E',C'D '  3380/3390 model types
                 DC    X'21',C'J '
                 DC    X'23',C'K '
                 DC    X'2E',C'E '
                 DC    X'26',C'-1'
                 DC    X'27',C'-2'
                 DC    X'24',C'-3'
                 DC    X'32',C'-9'
                 DC    X'FF',C'  '
Not_found_msg    DC    C'no matching volume(s) found'
Heading          DC    C'List of volume(s) matching ******'
Volser           DC    CL6' '
Volume_mask      DC    CL6' '
TPUT_line        DC    CL4Ø' '
Unit_no          DC    CL4' '
IPL_volume       DC    CL6' '
UCB_copy         DC    XL48'ØØ'
DCE_copy         DC    XL32'ØØ'
UCB_scanwork     DC    XL1ØØ'ØØ'
H_flag           DC    X'ØØ'
                 DS ØF
SSOB_area        DC (SSOBHSIZ)X'ØØ'

                 DS ØF
SSIB_area        DC (SSIBSIZE)X'ØØ'

                 DS ØF
SSSA_area        DC (SSSALN+SSSA1LN+32)X'ØØ'
         LTORG
         CVT   DSECT=YES
         IEFJESCT
         IEFSSOBH
         IEFJSSIB
         IEFSSSA
         IGDVLD
         IEFUCBOB
DCE      IECDDCE
         @REGS
         END
```

Automatic transmission of files

INTRODUCTION

When you have two or more MVS regions, the normal way of moving files between them is by using the classic transmit and receive commands. However, this is a manual procedure, that can only be applied to non-VSAM files.

To overcome this limitation, I developed two programs, one acting as a sender (RSPLIT) and the other as a receiver (RJOIN), that can perform these tasks automatically, simply by submitting a job on the sending side. Furthermore, these programs can also deal with VSAM files. The file being sent and the file at the destination need not be of the same kind: you can send a VSAM to a VSAM, a VSAM to a sequential, to a PDS member, or *vice versa*, in whatever combination you want. Non-VSAM files can have RECFM fixed, variable or undefined, with record lengths up to 32KB. Spanned records are not supported. VSAM files can be KSDS, ESDS, or RRDS.

The process behind this scheme is simple: program RSPLIT reads the input file and cuts it into 79 byte pieces. It also reads a collection of job cards that are meant to run at the receiving region. The split file is inserted in the appropriate location in those jobcards, as an inline file, and that job is submitted via an internal reader to the remote region. When it arrives there, the job executes program RJOIN, which reads the inline file pieces and rebuilds the original records, writing them to the destination dataset you specify.

Each record from the original file is preceded by two bytes containing the record length. The receiving program uses that information to correctly rebuild the records, and also uses it to write to the output file, in case of a VSAM file or non-VSAM with variable or undefined LRECL.

Each piece of the cut file is 79 bytes long, preceeded by a space. Since the file travels in the job stream, the initial space ensures that there is no risk of any accidental combination of characters being interpreted as job cards (for example, ‘/*’).

An example shown below will show more clearly how things work. The example below represents a job that will send a nonvsam sequential file to a VSAM ESDS.

You can see that there is a job within the job. The inside job is meant to run at the remote site. This job contains a /*XEQ card to the remote JES, as well as the user and password for the remote system. The cards are shifted to the right by two bytes, to avoid being processed by the local JES.

When you submit this job, program RSPLIT is executed. The first thing it does is to read the inline file with DDname JCLCARDS, on line 8 of the example. It reads the remote jobcards, removes the two spaces on the left of each, and writes them to the internal reader, which has the DDname INTERDR. This continues until it finds the SPLITFI DDname, on line 27 in the example. After reading and writing this line, the program starts reading INFILE, splits it as explained above, and sends it to the internal reader.

When INFILE is finished, the program continues to read from the jobcards at line 28. That line must be a '/*' to close SPLITFI. You can add other steps to the job to be executed remotely, if you want, either before or after the step that executes the RJOIN program. In this example, I have a step that deletes and recreates the destination file, but this is optional, of course. You might use an existing file to be overwritten or appended.

One important point, if your file is VSAM, the program needs to know about it. In the example, the receiving file is an ESDS, so you must pass a parameter to program RJOIN telling just that. If the sending file was a VSAM, then you should pass a parameter to RSPLIT. The parameter must be KSDS, ESDS, or RRDS. If the file is not VSAM, no parameter is needed.

Forgetting about this parameter will cause an abend. If either RSPLIT or RJOIN abends, the first thing to look for is whether the parameter (or lack of it) is correct. If RJOIN abends, the second thing to check is LRECL compatibility. You may be trying to write a record greater that the maximum record size of the output file.

Both jobs, local and remote, will inform you (on sysprint) of how many records were read and how many were written, so you can check it if you want. Program RJOIN will also test for return and reason codes when writing the output, in case it is VSAM. If the return code is not zero, the program terminates and sets the RC in register 15 for the job step. In this case, it also displays in sysprint the reason code.

For a complete list of return and reason codes, see *DFSMS/MVS Macro Instructions for Datasets*, SC26-4913. If the file is not VSAM, then a writing problem will cause the program to abend, with the apropriate system code.

Both programs can be used for purposes other than this automatic transmission scheme. Program RSPLIT does not need the JCLCARD file. You can declare it as DUMMY and direct INTERDR DDname to any 80 byte record file you want. In this case, you will have a split file that can be later restored by program RJOIN.

EXAMPLE JOB

```
000001 //JOB000 JOB REGION=2048K,MSGCLASS=X,MSGLEVEL=(1,1)
000002 //*
000003 //STEP0    EXEC PGM=RSPLIT
000004 //STEPLIB  DD DISP=SHR,DSN=my.local.loadlib
000005 //SYSPRINT DD SYSOUT=*
000006 //INFILE   DD DISP=SHR,DSN=my.local.input.file
000007 //INTERDR  DD SYSOUT=(B,INTRDR),DCB=(LRECL=80)
000008 //JCLCARDS DD  *
000009   //JOB001  JOB MSGLEVEL=(1,1),MSGCLASS=X,CLASS=X,
000010   //         REGION=2048K,USER=USER1,PASSWORD=USER1PW
000011   //*XEQ    JES2remote
000012   //STEPRM0  EXEC PGM=IDCAMS
000013   //SYSPRINT DD SYSOUT=*
000014   //SYSIN    DD *
000015       DELETE  my.remote.output.vsam CL PURGE
000016       DEFINE  CL(NAME(my.remote.output.vsam)  -
000017                  TRK(15 15)                   -
000018                  RECSZ(1000 2000)             -
000019                  NIXD                         -
000020                  VOL(VOL234))
000021   /*
000022   //*
000023   //STEPRM1  EXEC PGM=RJOIN,
000024   //         PARM='ESDS'
000025   //STEPLIB  DD DISP=SHR,DSN=my.remote.loadlib
```

```
000026   //SYSPRINT DD SYSOUT=*
000027   //OUTFILE  DD DISP=SHR,DSN=my.remote.output.vsam
000028   //SPLITFI  DD *
000029   /*
00003Ø   //*************************************
000031   //* Other steps for the remote partition
000032   //* can be inserted here
000033   //*************************************
000034 /*
```

RSPLIT

```
*=====================================================================*
*                                                                     *
* RSPLIT - Cuts a file in 79-byte pieces, with each record preceeded *
* by a halfword with its length. The output file has 8Ø-byte records *
* (one blank followed by 79 bytes of information).                    *
* Argument: input filetype (KSDS ESDS RRDS or blank for others).      *
* Returns R15 as RC in case of opening error.                         *
*                                                                     *
* Register use:                                                       *
*                                                                     *
* R2  - Address input record.                                         *
* R3  - Address input record (current location to be moved to output)*
* R4  - Length of input record (remaining to be moved).               *
* R5  - Address output buffer (current location to receive data).     *
* R6  - Length of output buffer (remaining to be filled).             *
* R7  - Argument for EX move. Equals the lower of R4 or R6 minus 1.   *
* R9  - Counts the number of read records.                            *
* R1Ø - Return address for BAL internal subroutines.                  *
* R11 - Address for IHADCB DSECT (DCB map).                           *
*                                                                     *
*=====================================================================*
&PROGRAM SETC  'RSPLIT'
&LEN79   SETC  '79'
LEN79    EQU   79
&PROGRAM AMODE 31
&PROGRAM RMODE 24
&PROGRAM CSECT
         SAVE  (14,12)
         LR    R12,R15
         USING &PROGRAM,R12
         USING IHADCB,R11             Address DCB for non vsam
         ST    R13,SAVEA+4
         LA    R11,SAVEA
         ST    R11,8(R13)
         LR    R13,R11
         B     FILETYPØ
         DC    CL16' &PROGRAM 1.2'
```

```
         DC    CL8'&SYSDATE'
*
*=====================================================================*
* Choose filetype from parm, set a flag and jump to correct open
*=====================================================================*
*
FILETYPØ DS    ØF
         LR    R2,R1
         L     R2,Ø(Ø,R2)          Get parameter
         OPEN  (SYSPRINT,OUTPUT)   Open for error messages
*
         CLC   =C'KSDS',2(R2)
         BNE   FILETYP1
         MVI   FILETYPE,C'K'
         B     OPENACB
*
FILETYP1 EQU   *
         CLC   =C'RRDS',2(R2)
         BNE   FILETYP2
         MVI   FILETYPE,C'K'
         B     OPENACB
*
FILETYP2 EQU   *
         CLC   =C'ESDS',2(R2)
         BNE   OPENDCB
         MVI   FILETYPE,C'E'
         MODCB RPL=INFILER,OPTCD=ADR  Only for ESDS
         B     OPENACB
*
*=====================================================================*
* Open input file (DCB or ACB) and then the remaining files
*=====================================================================*
*
OPENDCB  EQU   *                   Open sequential (non vsam)
         MVI   FILETYPE,C'S'       Set flag sequential type
         OPEN  (INFILE,INPUT)
         LTR   R15,R15
         BNZ   ERRO1
         LA    R11,INFILE          Address IHADCB of input file.
         TM    DCBRECFM,DCBBIT1    Is recfm V or U (B'x1xxxxxx)
         BNO   OPENFILS            No, jump ahead
         MVI   FILEVARI,C'U'       set recfm undefined
         TM    DCBRECFM,DCBBITØ    Is recfm U (B'11xxxxxx)
         BO    OPENFILS            Yes, jump ahead
         MVI   FILEVARI,C'V'       set recfm variable
         B     OPENFILS
*
OPENACB  EQU   *                   Open VSAM file
         OPEN  (INFILEA,INPUT)
         LTR   R15,R15
```

```
         BNZ   ERRO1
         B     OPENFILS
*
OPENFILS EQU   *                    Open other files
         OPEN  (INTERDR,OUTPUT)     Open internal rdr file
         LTR   R15,R15
         BNZ   ERRO2
         OPEN  (JCLCARDS,INPUT)     Open instream jcl
*
*====================================================================*
* Read and write JCL cards before instream file.
*====================================================================*
*
JCLBEF   EQU   *                     Read JCL cards before file and
         BAL   R1Ø,JCLINOU           write them to intrdr, until
         CLI   JCLENDFI,C'1'         EOF flag set?
         BE    JCLBEFX               yes, leave this loop.
         CLC   =C'//SPLITFI',JCL2    DDname for instream file found?
         BE    JCLBEFX               yes, leave loop.
         B     JCLBEF
*
JCLBEFX  EQU   *
         SR    R9,R9                 Read counter
         LA    R6,LEN79              Set length and output buffer.
         LA    R5,OUTBUF1
*
*====================================================================*
*        Mainloop to create instream file.
*====================================================================*
*
LEITURA  EQU   *                    Readloop
         BAL   R1Ø,READIN
         CH    R6,=H'1'              Check how many bytes remain in
         BH    R6TWO                 outbuf, before storing LRECL.
         BL    R6ZERO
*
R6ONE    EQU   *                    Just one byte remains,
         STCM  R4,B'ØØ1Ø',Ø(R5)     store first byte of LRECL,
         BAL   R1Ø,WRITEOUT         write outbuffer
         STCM  R4,B'ØØØ1',Ø(R5)     and store the second byte at
         LA    R5,1(Ø,R5)           the beginning of a new buffer,
         S     R6,=F'1'             and update pointer (R5)
         B     R6ZERO               and length (R6).
*
R6TWO    EQU   *                    Two or more bytes remain,
         STH   R4,Ø(Ø,R5)           store LRECL, update pointer and
         LA    R5,2(Ø,R5)           length.
         S     R6,=F'2'
         B     R6ZERO
*
```

```
R6ZERO   EQU   *                  Check if outbuf still has space
         LTR   R6,R6              available. If so, jump ahead to
         BNZ   MOVER              move more data, otherwise write
         BAL   R1Ø,WRITEOUT       outbuf first.
         B     MOVER
*
MOVER    EQU   *                  Which one is bigger, input data
         CR    R6,R4              left or output buffer available?
         BL    MOVER1             Choose the smaller to load R7.
         LR    R7,R4
         B     EXECMOVE
*
MOVER1   EQU   *
         LR    R7,R6
*
EXECMOVE EQU   *
         SH    R7,=H'1'           Prepare R7 for execute instr.
         EX    R7,EXMOVE
         LA    R7,1(Ø,R7)         And restore it afterwards.
         SR    R4,R7              Update pointers and lenghts.
         SR    R6,R7
         AR    R3,R7
         AR    R5,R7
         LTR   R6,R6              Output buffer ended?
         BNZ   MOVER2
         BAL   R1Ø,WRITEOUT       If yes, write outbuf.
         B     MOVER2
*
MOVER2   EQU   *
         LTR   R4,R4              Input record ended?
         BZ    LEITURA            Yes, go read another.
         B     MOVER              No, contunue to move data.
*
*=========================================================================*
*  Instream file ends here. Process what was left in the buffer,
*  if any, and write it. After that, write pos-instream file JCL
*  cards (like /* and other steps) and terminate program.
*=========================================================================*
*
ENDFILE  EQU   *                  End of input file. If R6 equals
         CH    R6,=H'&LEN79'      79, outbuf is empty, so just exit.
         BE    JCLAFT
         XR    R2,R2              If not, write a zero fullword to
         ST    R2,Ø(Ø,R5)         mark the end of file (or zero
         BAL   R1Ø,WRITEOUT       record length) for the RJOIN
*
JCLAFT   EQU   *                  End of input file.
         BAL   R1Ø,JCLINOU        Write JCL after instream file.
         CLI   JCLENDFI,C'1'      JCL file EOF flag set?
         BE    EXITØ              yes, leave.
```

```
         B     JCLAFT
*
EXITØ    EQU   *                  Close files and exit.
         BAL   R1Ø,REGXDISP       Prepare R9 for display
         PUT   SYSPRINT,ZMSG      Say how many records were read.
*
EXIT1    EQU   *                  Close files and exit.
         CLOSE INFILE
         CLOSE INFILER
         CLOSE INTERDR
         CLOSE SYSPRINT
         L     R15,RETCODE        Put return code in R15
         L     R13,SAVEA+4        and reload other regs.
         L     R14,12(R13)
         LM    RØ,R12,2Ø(R13)
         BR    R14
*
*=========================================================================*
*        Subroutines
*=========================================================================*
*
READIN   EQU   *                  Read input file subroutine
         CLI   FILETYPE,C'S'      Is it sequential (nonvsam?)
         BE    READSEQ            Yes, jump
*
READVSAM EQU   *
         GET   RPL=INFILER        Read VSAM file (locate method)
         LTR   R15,R15            End of file?
         BNZ   ENDFILE
         L     R3,VAREA           Get address of data in R3.
         SHOWCB RPL=INFILER,                                          X
               AREA=LRECL,                                            X
               LENGTH=4,                                              X
               FIELDS=RECLEN
         L     R4,LRECL           Get record length
         B     READEXIT           Branch to exit and return
*
READSEQ  EQU   *
         GET   INFILE             Read sequential (locate method)
         LR    R3,R1              copy address of data to R3.
         LH    R4,DCBLRECL        Load R4 with record length.
         CLI   FILEVARI,C'V'      Is recfm variable?
         BNZ   READEXIT           No, jump ahead.
         LA    R3,4(Ø,R3)         Yes, skip 4 bytes of RDW
         SH    R4,=H'4'           And reduce record length.
*
READEXIT EQU   *
         LA    R9,1(Ø,R9)         Increment record counter
         BR    R1Ø                Return
*
```

```
WRITEOUT EQU   *                   Write instream file routine
         PUT   INTERDR,OUTBUF      Write output buffer
         LA    R6,LEN79            Restore available length
         LA    R5,OUTBUF1          and pointer.
         BR    R1Ø                 Return
*
JCLINOU  EQU   *                   Read/write JCL cards routine
         GET   JCLCARDS,JCL1
         PUT   INTERDR,JCL2
         BR    R1Ø                 Return
JCLINOUX EQU   *                   Target EODAD for JCL cards
         MVI   JCLENDFI,C'1'       Set flag end file
         BR    R1Ø                 An return to whoever called IO.
*
REGXDISP EQU   *                   Routine to convert Rx to display
         CVD   R9,ZUNP             Convert to decimal
         UNPK  ZUNP2,ZUNP          Unpack from ZUnp to ZUnp2
         OI    ZUNP2A,X'FØ'        Remove signal
         MVC   ZDISP(8),ZUNP2      Move result to message area
         B     REGXDEND            Jump around storage
ZUNPØ    DS    ØD                  Double align for unpack
ZUNP     DS    CL8                 Decimal field
ZUNP2    DS    ØCL8                Unpacked field (8 bytes)
         DS    CL7
ZUNP2A   DS    C                   Sign byte will be ORed with FØ
ZMSG     DC    C'>>> Number of records read from input file: '
ZDISP    DC    CL36' '
REGXDEND DS    ØH                  Align
         BR    R1Ø                 Return
*
ERRO1    EQU   *
         ST    R15,RETCODE
         PUT   SYSPRINT,=CL8Ø'>>> Error opening input file'
         B     EXIT1
ERRO2    EQU   *
         ST    R15,RETCODE
         PUT   SYSPRINT,=CL8Ø'>>> Error opening internal rdr file'
         B     EXIT1
*
INFILEA  ACB   DDNAME=INFILE        VSAM ACB
INFILER  RPL   ACB=INFILEA,         VSAM RPL                        X
               OPTCD=LOC,           Locate method                   X
               AREA=VAREA,          Record buffer address           X
               ARG=CHAVE            Only needed for rrds
*
INFILE   DCB   DSORG=PS,MACRF=(GL), For sequential files            X
               EODAD=ENDFILE,                                       X
               DDNAME=INFILE
*
JCLCARDS DCB   DSORG=PS,MACRF=(GM), JCL input                       X
```

```
               EODAD=JCLINOUX,                                          X
               DDNAME=JCLCARDS
*
INTERDR  DCB   DSORG=PS,MACRF=(PM), Output for internal reader          X
               LRECL=8Ø,                                                X
               DDNAME=INTERDR
*
SYSPRINT DCB   DSORG=PS,MACRF=(PM),                                     X
               LRECL=8Ø,                                                X
               DDNAME=SYSPRINT
*
         LTORG
         DS    ØF
EXMOVE   MVC   Ø(Ø,R5),Ø(R3)
SAVEA    DS    18F
VAREA    DS    F                    Address of record buffer (VSAM)
CHAVE    DS    F                    Record key (rrds - VSAM)
LRECL    DS    F                    Record length (VSAM)
RETCODE  DC    F'Ø'                 Return code
*
OUTBUF   DC    C' '
OUTBUF1  DS    CL&LEN79
OUTBUFX  DS    CL4
FILETYPE DS    C                    Flag for Seq, ESDS, KSDS, RRDS
JCLENDFI DC    C'Ø'                 Flag set to 1 when JCL EOF.
FILEVARI DC    C'F'                 Flag preset to recfm F
*
JCL1     DS    ØC                   Read JCL cards area
         DS    CL2                  Offset by two positions
JCL2     DS    CL78                 Output JCL
         DC    C'  '                Remaining two bytes.
*
         DCBD  DSORG=PS
         YREGS
         END
```

RJOIN

```
*=================================================================*
*                                                                 *
* RJOIN - Restores files created by program RSPLIT.               *
* Argument: output filetype (KSDS ESDS RRDS or blank for others)  *
* Returns R15 as RC in case of opening error or VSAM write error. *
* VSAM write error also displays the reason code.                 *
*                                                                 *
* Register use is identical to program RSPLIT.                    *
*                                                                 *
*=================================================================*
&PROGRAM SETC  'RJOIN'
```

```
LEN79    EQU   79
&PROGRAM AMODE 31
&PROGRAM RMODE 24
&PROGRAM CSECT
         SAVE  (14,12)
         LR    R12,R15
         USING &PROGRAM,R12
         USING IHADCB,R11
         ST    R13,SAVEA+4
         LA    R11,SAVEA
         ST    R11,8(R13)
         LR    R13,R11
         B     FILETYPØ
         DC    CL16' &PROGRAM 1.2'
         DC    CL8'&SYSDATE'
*
*====================================================================*
* Choose filetype from parm, set a flag and jump to correct open
*====================================================================*
*
FILETYPØ DS    ØF
         LR    R2,R1
         L     R2,Ø(Ø,R2)          Get parameter
         OPEN  (SYSPRINT,OUTPUT)   Open for error messages
*
         CLC   =C'KSDS',2(R2)
         BNE   FILETYP1
         MVI   FILETYPE,C'K'
         B     OPENACB
*
FILETYP1 EQU   *
         CLC   =C'RRDS',2(R2)
         BNE   FILETYP2
         MVI   FILETYPE,C'K'
         B     OPENACB
*
FILETYP2 EQU   *
         CLC   =C'ESDS',2(R2)
         BNE   OPENDCB
         MVI   FILETYPE,C'E'
         MODCB RPL=OUTFILER,       Modify RPL, ESDS                X
               OPTCD=ADR
         B     OPENACB
*
*====================================================================*
* Open output file (DCB or ACB) and then the remaining files
*====================================================================*
*
OPENDCB  EQU   *                   Open non VSAM
         MVI   FILETYPE,C'S'       Set flag sequential type
```

```
         OPEN  (OUTFILE,OUTPUT)
         LTR   R15,R15
         BNZ   ERRO1
         LA    R11,OUTFILE         Address DCB
         TM    DCBRECFM,DCBBIT1    Is recfm V or U (B'x1xxxxxx)
         BNO   OPENFILS            No, jump ahead
         MVI   FILEVARI,C'U'       set recfm undefined
         TM    DCBRECFM,DCBBITØ    Is recfm U (B'11xxxxxx)
         BO    OPENFILS            Yes, jump ahead
         MVI   FILEVARI,C'V'       set recfm variable
         B     OPENFILS
*
OPENACB  EQU   *                   Open VSAM file
         OPEN  (OUTFILEA,OUTPUT)
         LTR   R15,R15
         BNZ   ERRO1
         B     OPENFILS
*
OPENFILS EQU   *
         OPEN  (SPLITFI,INPUT)     Open input file
         LTR   R15,R15
         BNZ   ERRO2
         LA    R2,RECORDF          R2: record area without RDW
         CLI   FILEVARI,C'V'       Is recfm variable?
         BNE   RESETPTR            No, jump
         LA    R2,RECORDV          Yes, R2: record area with RDW
*
RESETPTR EQU   *
         LA    R3,RECORDF          R3 points record data
         SR    R4,R4               Reset counters
         SR    R6,R6
         SR    R9,R9
*
*=====================================================================*
* Mainloop to rebuild file
*=====================================================================*
*
LEITURA  EQU   *
         BAL   R1Ø,READIN          Read input file
         LTR   R4,R4               Anything to move?
         BNZ   MOVER               Yes, jump
*
NEWOUT   EQU   *
         CH    R6,=H'1'            Input bytes remaining
         BL    R6ZERO              None, jump and read another
         BH    R6TWO
*
R6ONE    EQU   *                   One, insert it, read another
         ICM   R4,B'ØØ1Ø',Ø(R5)    record and insert its first
         BAL   R1Ø,READIN          byte to complete halfword
```

```
         ICM   R4,B'ØØØ1',Ø(R5)    with the length of the record
         LA    R5,1(Ø,R5)          being rebuilt.
         SH    R6,=H'1'
         B     R6ZERO
*
R6TWO    EQU   *                   Two bytes: record length.
         LH    R4,Ø(Ø,R5)          If length is zero, the outfile
         LTR   R4,R4               is completed, exit.
         BZ    EXITØ
         LA    R5,2(Ø,R5)          Advance halfword
         SH    R6,=H'2'
         ST    R4,LRECL            Store length for vsam RPL
         CLI   FILEVARI,C'F'       Is recfm fixed?
         BE    R6ZERO              Yes, jump
         STH   R4,DCBLRECL         Put leng in DCB (recfm V or U)
*
R6ZERO   EQU   *
         LTR   R6,R6               Any input bytes left?
         BZ    LEITURA             No, read more
         B     MOVER               Yes, move them.
*
MOVER    EQU   *
         CR    R6,R4               Move data from input to output
         BL    MOVER1              Choose shortest for move length
         LR    R7,R4
         B     EXECMOVE
*
MOVER1   EQU   *
         LR    R7,R6
*
EXECMOVE EQU   *
         SH    R7,=H'1'
         EX    R7,EXMOVE           Move data
         LA    R7,1(Ø,R7)
         SR    R4,R7               Reset pointers
         SR    R6,R7
         AR    R3,R7
         AR    R5,R7
         LTR   R4,R4
         BNZ   R6ZERO              Record rebuild complete?
         BAL   R1Ø,WRITEOUT        Yes, write it.
         B     NEWOUT
*
*=================================================================*
* End of input file. Write remaining bytes, write message with
* the number of records written, set RCand leave.
*=================================================================*
*
ENDFILE  EQU   *
         LTR   R4,R4               Any data remaining?
```

```
         BZ    EXITØ               No, exit.
         BAL   R1Ø,WRITEOUT        Yes, write it.
*
EXITØ    EQU   *
         CVD   R9,ZUNP             Convert to decimal
         BAL   R1Ø,REGDDISP        Prepare R9 for display
         MVC   ZMSG,=CL31'>>> Number of records written: '
         PUT   SYSPRINT,ZMSG       Say how many recs were written
*
EXIT1    EQU   *
         CLOSE SPLITFI
         CLOSE OUTFILE
         CLOSE OUTFILEA
         CLOSE SYSPRINT
         L     R15,RETCODE         Put return code in R15
         L     R13,SAVEA+4         and reload other regs.
         L     R14,12(R13)
         LM    RØ,R12,2Ø(R13)
         BR    R14
*
*=====================================================================*
*        Subroutines
*=====================================================================*
*
READIN   EQU   *                   Read split file
         GET   SPLITFI
         LR    R5,R1               Copy address
         LA    R5,1(Ø,R5)          Skip initial blank byte
         LA    R6,LEN79            Set number of bytes
         BR    R1Ø
*
WRITEOUT EQU   *                   Write output subroutine
         CLI   FILETYPE,C'S'       Is it sequential (nonvsam?)
         BE    WRITESEQ            Yes, jump
*
WRITEVSA EQU   *                   Write VSAM:
         L     R4,LRECL            Get length for RPL
         MODCB RPL=OUTFILER,       Modify record length in RPL        X
               RECLEN=(R4)
         PUT   RPL=OUTFILER        Write VSAM
         LTR   R15,R15
         BZ    WRITEXIT            No error, go to exit and return
*
         ST    R15,RETCODE         Error, keep return code
         SHOWCB RPL=OUTFILER,      Get reason code.                   X
               AREA=REASONCO,                                         X
               LENGTH=4,                                              X
               FIELDS=FDBK
         B     ERRO3
*
```

```
WRITESEQ EQU   *                  Write sequential:
         CLI   FILEVARI,C'V'      Is recfm variable ?
         BNE   WRITESE1           No, jump
         LH    R8,LRECL+2         Yes, get lrecl
         LA    R8,4(Ø,R8)         Add 4 since RDW length has itself.
         STH   R8,RDW1            Store length
*
WRITESE1 EQU   *
         PUT   OUTFILE,(R2)       Write sequential
*
WRITEXIT EQU   *
         LA    R9,1(Ø,R9)         Increment rec counter
         LA    R3,RECORDF         Reset pointer
         BR    R1Ø                Return
*
ERRO1    EQU   *
         ST    R15,RETCODE
         PUT   SYSPRINT,=CL8Ø'>>> Error opening output file '
         B     EXIT1
ERRO2    EQU   *
         ST    R15,RETCODE
         PUT   SYSPRINT,=CL8Ø'>>> Error opening input file  '
         B     EXIT1
ERRO3    EQU   *
         PUT   SYSPRINT,=CL8Ø'>>> Error writing output file '
         MVC   YHEX,RETCODE
         BAL   R1Ø,REGXDISP
         MVC   ZMSG,=CL31'>>>>>> VSAM return code is'
         PUT   SYSPRINT,ZMSG
         MVC   YHEX,REASONCO
         BAL   R1Ø,REGXDISP
         MVC   ZMSG,=CL31'>>>>>> VSAM reason code is'
         PUT   SYSPRINT,ZMSG
         B     EXITØ
*
REGDDISP EQU   *                       Routine to convert ZUNP to
         UNPK  ZUNP2,ZUNP              decimal display
         OI    ZUNP2A,X'FØ'
         MVC   ZDISP(8),ZUNP2          Move result to message area
         BR    R1Ø                     Return
*
REGXDISP EQU   *                       Routine to convert YUNP to
         UNPK  YUNP9(9),YHEX5(5)       hexadecimal display.
         NC    YUNP,=X'ØFØFØFØFØFØFØFØF'
         TR    YUNP,=C'Ø123456789ABCDEF'
         MVC   ZDISP(8),YUNP           Move result to message area
         BR    R1Ø                     Return
*=============================================================*
*        Working areas
*=============================================================*
*
OUTFILEA ACB   DDNAME=OUTFILE,MACRF=OUT
```

```
OUTFILER RPL   ACB=OUTFILEA,            VSAM RPL                      X
               AREA=RECORDF,            Record area address           X
               AREALEN=32760,           its length                    X
               ARG=CHAVE                RRDS likes this thing
*
OUTFILE  DCB   DSORG=PS,MACRF=(PM),     For sequential files          X
               DDNAME=OUTFILE
*
SPLITFI  DCB   DSORG=PS,MACRF=(GL),     Input split file              X
               EODAD=ENDFILE,                                         X
               DDNAME=SPLITFI
*
SYSPRINT DCB   DSORG=PS,MACRF=(PM),                                   X
               LRECL=80,                                              X
               DDNAME=SYSPRINT
*
         LTORG                         Must have this (within R12)
         DS    0F
EXMOVE   MVC   0(0,R3),0(R5)           Executed instruction.
SAVEA    DS    18F
CHAVE    DS    F                       RRDS obligue.
LRECL    DS    F                       Record length
RETCODE  DC    F'0'                    Return code for write
REASONCO DC    F'0'                    Reason code for write
FILETYPE DS    C                       Flag for Seq ESDS KSDS RRDS
FILEVARI DC    C'F'                    Flag preset for recfm fixed.
YHEX5    DS    0CL5
YHEX     DS    CL4
         DS    CL1
YUNP9    DS    0CL9
YUNP     DS    CL8
         DS    CL1
ZUNP0    DS    0D
ZUNP     DS    CL8
ZUNP2    DS    0CL8
         DS    CL7
ZUNP2A   DS    C
ZMSG     DS    CL31
ZDISP    DS    CL49
*
RECORDV  DS    0F                      RDW for variable nonvsam
RDW1     DC    H'0'                    Record length
RDW2     DC    H'0'                    Zero
RECORDF  DS    CL32760                 Record data rebuild area
         DCBD  DSORG=PS
         YREGS
         END
```

Luis Paulo Riberio
Systems Engineer
Edinfor (Portugal)

Commands and output at the master console

INTRODUCTION

The following procedure may be useful for issuing commands at the master console and displaying their output, especially if TSO is not up (or will not come up). It calls a program to request the command from the console (via WTOR) and passes the output to IKJEFT01 to be processed. The output is then passed to the final program to be displayed at the console. You may want to add additional DDnames to the IKJEFT01 step, such as REXX libraries, etc, which will enable you to enter a CLIST/REXX name to be executed.

ISSUACMD

```
//CMD      PROC
//WRIT     EXEC PGM=WRITPARM
//STEPLIB   DD  DISP=SHR,DSN=SYSG.LINKLIB
//OUTPUT    DD  DSN=&&OUTPUT,DISP=(MOD,PASS),UNIT=SYSDA,SPACE=(TRK,1)
//*
//TSO      EXEC PGM=IKJEFTØ1,DYNAMNBR=5Ø
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  DSN=&&TSPRT,DISP=(MOD,PASS),UNIT=SYSDA,SPACE=(TRK,1),
//          RECFM=FBA,LRECL=133,BLKSIZE=133
//SYSLBC    DD  DISP=SHR,DSN=SYS1.BRODCAST
//SYSUADS   DD  DISP=SHR,DSN=SYS1.UADS
//SYSTSIN   DD  DISP=(OLD,PASS),DSN=&&OUTPUT
//*
//DISPL    EXEC PGM=SPLATOUT
//STEPLIB   DD  DISP=SHR,DSN=SYSG.LINKLIB
//SYSPRINT DD  SYSOUT=*
//INPUT     DD  DSN=&&TSPRT,DISP=(OLD,PASS)
```

WRITPARM

```
         PRINT NOGEN
*********************************************************************
*              W R I T P A R M    U T I L I T Y                     *
*                                                                   *
*        THIS IS A UTILITY TO GET A COMMAND FROM THE CONSOLE        *
*        (VIA WTOR), THEN WRITE IT TO A DATASET (THIS WILL          *
*        THEN BE USED AS INPUT TO 'IKJEFTØ1' IN A LATER STEP).      *
*                                                                   *
*        CAN BE USED WHEN ACCESS TO TSO IS NOT POSSIBLE.            *
```

```
*********************************************************************
WRITPARM CSECT
         BAKR  R14,Ø               SAVE CALLER DATA ON STACK
         LR    R12,R15             GET ENTRY POINT
         USING WRITPARM,R12        MODULE ADDRESSABILITY
*
         MVC   ECB,=F'Ø'           ZEROIZE ECB
         WTOR  MF=(E,WTOR)         ISSUE WTOR
         WAIT  1,ECB=ECB           WAIT FOR REPLY
         CLC   REPLY(4),=C'END '   END?
         BE    RETURN4             YES..FORGET THE REST
*
         OPEN  (OUTPUT,(OUTPUT))
*
         PUT   OUTPUT,REPLY        WRITE RECORD
*
CLOSE    DS    ØH
         CLOSE OUTPUT              CLOSE OUTPUT
RETURN   DS    ØH
         XR    R15,R15             SET RC=Ø
         PR    ,                   RESTORE CALLER DATA, RETURN
RETURN4  DS    ØH
         LA    R15,4
         PR    ,
*
*————————  D A T A      A R E A S  ————————
*
ECB      DC    F'Ø'
REPLY    DC    CL8Ø' '
WTOR     WTOR  '>>> ENTER COMMAND TO BE PROCESSED, "END" TO TERMINATE',X
               REPLY,8Ø,ECB,MF=L
*
OUTPUT   DCB   DDNAME=OUTPUT,MACRF=PM,DSORG=PS,LRECL=8Ø,RECFM=F
*
         YREGS
*
         LTORG
*
         END
```

SPLATOUT

```
         PRINT NOGEN
*********************************************************************
*                                                                   *
*              S P L A T O U T    U T I L I T Y                     *
*                                                                   *
*        THIS IS A UTILITY TO RETRIEVE OUTPUT FROM A BATCH          *
*        TSO COMMAND AND DISPLAY IT ON THE CONSOLE.                 *
*                                                                   *
```

```
*        CAN BE USED WHEN ACCESS TO TSO IS NOT POSSIBLE.             *
*                                                                    *
**********************************************************************
SPLATOUT CSECT
         BAKR  R14,Ø               SAVE CALLER DATA ON STACK
         LR    R12,R15             GET ENTRY POINT
         USING SPLATOUT,R12        MODULE ADDRESSABILITY
*
         OPEN  (INPUT,(INPUT))
*
         WTO   MF=(E,WTO2)
GETLOOP  DS    ØH
         MVI   INAREA,C' '
         MVC   INAREA+1(L'INAREA-1),INPUT
*
         GET   INPUT,INAREA        READ RECORD
*
         MVC   WTO+7(8Ø),INAREA+1  IGNORE C-C
         WTO   MF=(E,WTO)          ISSUE WTO
         B     GETLOOP             GET MORE
CLOSE    DS    ØH
         WTO   MF=(E,WTO2)
         CLOSE INPUT               CLOSE INPUT
RETURN   DS    ØH
         XR    R15,R15             SET RC=Ø
         PR    ,                   RESTORE CALLER DATA, RETURN
RETURN4  DS    ØH
         LA    R15,4
         PR    ,
*
*————————  D A T A      A R E A S  ————————
*
HDR      DC    CL14Ø' '
INAREA   DC    CL14Ø' '
WTO      WTO   '>> 123456789Ø123456789Ø123456789Ø123456789Ø123456789Ø12X
               3456789Ø123456789Ø1234567890',MF=L
WTO2     WTO   '>>++++++++++++++++++++++++++++++++++++++++++++++++++++X
               +++++++++++++++<<',MF=L
*
INPUT    DCB   DDNAME=INPUT,MACRF=GM,DSORG=PS,EODAD=CLOSE
*
         YREGS
*
         LTORG
*
         END
```

SAMPLE OUTPUT (JOB)

```
 SDSF OUTPUT DISPLAY ISSUACMD STC60336  DSID     2 LINE 2       COLUMNS 02- 81
 COMMAND INPUT ===>                                            SCROLL ===> PAGE

17.32.34 STC60336 -- MONDAY,    21 AUG 2000 --
17.32.34 STC60336  IEF695I START ISSUACMD WITH JOBNAME ISSUACMD IS ASSIGNED TO U
17.32.34 STC60336  £HASP373 ISSUACMD STARTED
17.32.34 STC60336  IEF403I ISSUACMD - STARTED - TIME=17.32.34
17.32.35 STC60336 @14 >>> ENTER COMMAND TO BE PROCESSED, "END" TO TERMINATE
17.32.51 STC60336  R 14,SUPPRESSED
17.32.51 STC60336  -                                        -TIMINGS (MINS.)-
17.32.51 STC60336  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    TCB    SRB  CLOCK
17.32.51 STC60336  -ISSUACMD WRIT                 00      9    .00    .00     .2
17.32.52 STC60336  -ISSUACMD TSO                  00     40    .00    .00     .0
17.32.52 STC60336  +>>++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
17.32.52 STC60336  +>> IKJ56644I NO VALID TSO USERID, DEFAULT USER ATTRIBUTES US
17.32.52 STC60336  +>> READY
17.32.52 STC60336  +>> LISTC ALL ENT(SYS1.LINKLIB)
17.32.52 STC60336  +>> NONVSAM --- SYS1.LINKLIB
17.32.52 STC60336  +>>      IN-CAT - CATALOG.MASTPRI.SYSA
17.32.52 STC60336  +>>      HISTORY
17.32.52 STC60336  +>>        DATASET-OWNER--(NULL)     CREATION----1999.
17.32.52 STC60336  +>>        RELEASE---------2     EXPIRATION---0000.
17.32.52 STC60336  +>>      VOLUMES
17.32.52 STC60336  +>>        VOLSER------&RVOL2     DEVTYPE---X'000000
17.32.52 STC60336  +>>      ASSOCIATIONS----(NULL)
17.32.52 STC60336  +>>      ATTRIBUTES
17.32.52 STC60336  +>> READY
17.32.52 STC60336  +>> END
17.32.52 STC60336  +>>++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
17.32.52 STC60336  -ISSUACMD DISPL                 00     22    .00    .00     .
17.32.52 STC60336  IEF404I ISSUACMD - ENDED - TIME=17.32.52
17.32.52 STC60336  -ISSUACMD ENDED.  NAME-                    TOTAL TCB CPU TI
17.32.52 STC60336  £HASP395 ISSUACMD ENDED
```

SAMPLE OUTPUT (SYSLOG)

```
 SDSF SYSLOG  9397.330 CARG CARG 08/21/2000 LINE  27,155     COLUMNS  46 125
 COMMAND INPUT ===>                                           SCROLL ===> CSR
 00000094  IEF403I ISSUACMD - STARTED - TIME=17.32.34
 00000090 @14 >>> ENTER COMMAND TO BE PROCESSED, "END" TO TERMINATE
 00000094  ISF015I SDSF COMMAND EXECUTED  'REPLY 14   TEXT OF REPLY IS SUPPRESSE
             ' BQIBI26  £LPROC2  A31TUB01
 00000294  R 14 SUPPRESSED
 00000094  IEE600I REPLY TO 14 IS;SUPPRESSED
 00000294  -                                       -TIMINGS (MINS.)-
 00000294  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    TCB    SRB  CLOCK   SERV
 00000290  -ISSUACMD WRIT                 00      9    .00    .00     .2    600
```

```
00000290  -ISSUACMD TSO                  00     40    .00    .00     .0   2937
00000090  +>>+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
00000090  +>> IKJ56644I NO VALID TSO USERID, DEFAULT USER ATTRIBUTES USED
00000090  +>> READY
00000090  +>> LISTC ALL ENT(SYS1.LINKLIB)
00000090  +>> NONVSAM ─── SYS1.LINKLIB
00000090  +>>      IN-CAT — CATALOG.MASTPRI.CARG
00000090  +>>      HISTORY
00000090  +>>        DATASET-OWNER──(NULL)     CREATION────1999.298
00000090  +>>        RELEASE─────────2     EXPIRATION───0000.000
00000090  +>>      VOLUMES
00000090  +>>        VOLSER────────&RVOL2     DEVTYPE───X'00000000'
00000090  +>>      ASSOCIATIONS─────(NULL)
00000090  +>>      ATTRIBUTES
00000090  +>> READY
00000090  +>> END
00000090  +>>+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
00000290  -ISSUACMD DISPL                00     22    .00    .00     .0   1112
               0      0      0     0
00000005  IEF404I ISSUACMD - ENDED - TIME=17.32.52
00000290  -ISSUACMD ENDED.  NAME-                     TOTAL TCB CPU TIME=-    .
```

Grant Carson
Systems Programmer (UK)

A utility to update sequential and partitioned datasets

THE PROBLEM

We sometimes require general update in plenty of our jobs, source codes, or data. System programmers have to change all members of the parmlib library during the process of migration or cloning of the system. Production jobs during hardware and software migration generally must be updated (STEPLIB libraries, for example).

Application programmers can make new test data by tailoring fields from the existing dataset. They can also change all main programs to refer to a new subroutine name.

Changing the contents of multiple datasets using an editor is complex and can result in incomplete results.

A SOLUTION

We wrote a REXX procedure called UPDATE that solves our mass update problem. All you need to do is specify the parameters that are described at the beginning of the procedure. The REXX procedure UPDATE gives the following results:

- The original dataset is updated in place (with NOTEST).
- A detailed report about updated strings produced in the SYSPRINT file.

When you enlarge a length of the record by the update instructions, the procedure ends with a return code of 8, writes warnings in the SYSPRINT file, and does not update this record.

Note: we assume that you take a back-up copy of all datasets before updating their content in a standard way for your installation.

SOURCE

```
/******************************** REXX ***********************************/
/*                                                                       */
/* %UPDATE   NOTEST DsName Volume                                        */
/*           TEST                                                        */
/*                                                                       */
/*           NOTEST - Update will be performed and report will be        */
/*                    generated                                          */
/*           TEST   - Report will be generated with the record           */
/*                    content after updating, but update will not be     */
/*                    performed                                          */
/*                                                                       */
/*           DsName - Name of the sequential dataset or library          */
/*                                                                       */
/*           Volume - name of the dasd volume on which dataset           */
/*                    resides, if it is not in catalog                   */
/*                                                                       */
/*  You can specify following statements in the PARMLIB file:           */
/*                                                                       */
/*  C p 'string1' 'string2' - change string1 with string2               */
/*  D p 'string1'           - delete string1 from dataset               */
/*  R p 'string1' 'string2' - remove content bounded by string1 and     */
/*                            string2                                    */
/*                                                                       */
/*     p - position can be:                                              */
/*        nnnnn - number from 1 to length(record)-length(string1)        */
/*     ALL - change all occurrences of string1 into string2              */
/*                                                                       */
/*  *  comment                                                           */
```

```
/*********************************************************************/
/*  Trace ?R */

 ARG TEST DsName Volume
 userid=SYSVAR(SYSUID)
 prefix=SYSVAR(SYSPREF)
 "PROFILE NOPREFIX"
 If SYSDSN(DsName) <> 'OK'
 Then Do
      Say '>>> Missing dataset name !!!'
      rrc=12
      End
 Else Do
      action.Ø=Ø
      position.Ø=Ø
      str1.Ø=Ø
      str2.Ø=Ø

      call Get_parmlib

      rrc=Ø
      t=OUTTRAP('dsnc.',,NOCONCAT)
      "LISTDS "DsName
      t=OUTTRAP('OFF')

      PARSE UPPER VAR dsnc.3 recfm lrecl blksize dsorg

      If dsorg = 'PS' OR (dsorg = 'PO' AND Index(DsName,'(') > Ø)
      Then Do
           rcu=Update_dataset(TEST, DsName, Volume)
           rrc=MAX(rrc,rcu)
           End
      Else
      If dsorg = 'PO'
      Then Do;
           t=OUTTRAP('dsnc.',,NOCONCAT)
           "LISTDS "DsName" members "
           t=OUTTRAP('OFF')

           Do i=1 To dsnc.Ø
              If INDEX(dsnc.i,'MEMBERS') > Ø
              Then Leave
           End
           Do i=i+1 To dsnc.Ø
              Parse Var Dsnc.i Member Rest
              Ds_Name=DsName||'('||Member||')'
              rcu=Update_dataset(TEST, Ds_Name, Volume)
              rrc=MAX(rrc,rcu)
           End
           End
      Else Do
           Say 'This Dsorg' dsorg ' is not supported !!!'
           rrc=16
```

```
           End
      End
 If prefix <> ''
 Then "PROFILE PREFIX("prefix")"

Return rrc

/*-------------------------------------------------------------------*/
/* Update dataset                                                    */
/*-------------------------------------------------------------------*/
Update_dataset:Procedure Expose action. position. str1. str2.
Arg TEST, Ds_Name, Volume

Say COPIES('=',8Ø)
Say CENTER('>' Ds_Name '<',8Ø,'*')

call alloc_Ds 'INOUT' Ds_Name Volume
rrc=Ø

"EXECIO Ø DISKRU inout (OPEN)"
If RC <> Ø
Then Do
     Say '>>> Dataset' DS_name ' CANNOT BE OPENED !!!'
     Return 4
     End

"EXECIO 1 DISKRU inout (STEM Records.)"
NoRec=1
Do While(RC < 2)
   Rec_length_source=Length(Records.1)
   Recordt.1=Update_Record(Records.1,NoRec)
   Recordt.1=STRIP(Recordt.1,'T')
   Rec_length_target=Length(Recordt.1)

   If Rec_length_source >= Rec_length_target
   Then Do
        If TEST = 'NOTEST'
        Then "EXECIO 1 DISKW  inout (STEM Recordt.)"
        End
   Else Do
        Say '*** WARNING  Length of Record after UPDATE ',
            Rec_length_target ' is GREATER THAN source length',
            Rec_length_source
        Say '*** WARNING  Record 'NoRec' is not UPDATED !'
        rrc=8
        End
   "EXECIO 1 DISKRU inout (STEM Records.)"
   NoRec=NoRec+1
End
"EXECIO Ø DISKRU inout (FINIS)"
Return rrc
```

```
/*-------------------------------------------------------------------*/
/* Get parmlib statements                                            */
/*-------------------------------------------------------------------*/
Get_parmlib: Procedure Expose action. position. str1. str2.
"EXECIO * DISKR parmlib (STEM parmlib. FINIS)"
k=Ø
Do i = 1 To parmlib.Ø
   PARSE UPPER VAR parmlib.i  Act Pos  ''''S1''''  ''''S2''''
   If Act <> '*'           /* if parameter is not comment */
   Then Do
        k=k+1
        action.k=Act
        position.k=Pos
        str1.k=S1
        str2.k=S2
        Say '>>> Action='action.k 'Position='position.k,
            'String1='str1.k   'String2='str2.k
        End
End
action.Ø  =k
position.Ø=k
str1.Ø    =k
str2.Ø    =k
Return
/*-------------------------------------------------------------------*/
/* Update Record                                                     */
/*-------------------------------------------------------------------*/
Update_Record:Procedure Expose action. position. str1. str2.
Arg Rec, NoRec

Rec_source=Rec
change='N'
Do i=1 to action.Ø
   l=Length(str1.i)
   Select
     When action.i = 'C'
       Then Do
            If Position.i = 'ALL'
            Then Do Until (J=Ø)
                 j=Index(Rec,str1.i)
                 If J > Ø
                 Then Do
                      Rec=Left(Rec,j-1)||str2.i||Substr(Rec,j+l)
                      change='Y'
                      End
                 End
            Else Do
                 j=position.i
                 If Substr(Rec,j,l) = str1.i
                 Then Do
                      Rec=Left(Rec,j-1)||str2.i||Substr(Rec,j+l)
                      change='Y'
                      End
                 End
```

```
       End
When action.i = 'D'
  Then Do
       If Position.i = 'ALL'
       Then Do Until (J=Ø)
            j=Index(Rec,str1.i)
            If J > Ø
            Then Do
                 Rec=Delstr(Rec,j,l)
                 change='Y'
                 End
            End
       Else Do
            j=position.i
            If Substr(Rec,j,l) = str1.i
            Then Do
                 Rec=Delstr(Rec,j,l)
                 change='Y'
                 End
            End
       End
When action.i = 'R'
  Then Do
       If Position.i = 'ALL'
       Then Do Until (J=Ø)
            j=Index(Rec,str1.i)
            If J > Ø
            Then Do
                 k=Index(Substr(Rec,j+l),str2.i)
                 If k > Ø
                 Then Do
                      l2=Length(str2.i)
                      Rec=Left(Rec,j-1)||Substr(Rec,j+l+k+l2-1)
                      change='Y'
                      End
                 End
            End
       Else Do
            j=position.i
            If Substr(Rec,j,l) = str1.i
            Then Do
                 k=Index(Substr(Rec,j+l),str2.i)
                 If k > Ø
                 Then Do
                      l2=Length(str2.i)
                      Rec=Left(Rec,j-1)||Substr(Rec,j+l+k+l2-1)
                      change='Y'
                      End
                 End
            End
       End
Otherwise
  Do
```

```
       Say '>>> Invalid action !!!' action.i
       Exec 16
       End
   End /* select */
   If change='Y'
   Then Do
        Say COPIES('-',4Ø)
        Say '>>> Action' Action.i Position.i str1.i str2.i
        Say COPIES('-',12)
        Say NoRec '>>> Source' Rec_source
        Say Norec '>>> Target' Rec
        Rec_source=Rec
        change='N'
        End
 End
 Return Rec

/*-------------------------------------------------------------------*/
/* Alloc Dataset                                                     */
/*-------------------------------------------------------------------*/
 Alloc_DS: Procedure
 Arg DD_Name Ds_Name Volume

 msgstat=MSG("OFF")    /* Inhibit the display of TSO/E informational */
                       /* messages */
 "FREE F("DD_Name")"
 t=MSG(msgstat)        /* Return the previous status of message */.

 If Volume = ''
 Then "ALLOC F("DD_Name") DA("''''Ds_Name''''") OLD REUSE"
 Else "ALLOC F("DD_Name") DA("''''Ds_Name''''") OLD REUSE",
              " VOLUME("Volume") UNIT(SYSDA)"
 Return
```

JOB EXAMPLE FOR SUBMITTING PROCEEDURE IN BATCH

```
//useridU  JOB CLASS=A,MSGCLASS=X,MSGLEVEL=(Ø,Ø),NOTIFY=&SYSUID
//UPDATE    EXEC PGM=IKJEFTØ1,DYNAMNBR=5Ø,REGION=4M
//SYSPROC   DD  DSN=userid.PROD.CLIST,DISP=SHR
//SYSTSPRT  DD  SYSOUT=*
//SYSPRINT  DD  SYSOUT=*
//PARMLIB   DD  *
* update DASD volume names in parmlib for the clone system
   C  ALL  'MVSRS'  'CLORS'
   C  ALL  'MVSCAT'  'CLOCAT'
/*
//SYSTSIN   DD  *
     %UPDATE  NOTEST SYS1.PARMLIB CLORS1
/*
```

Tape and DASD UCB display

Several years ago, I submitted this program to *MVS Update*. It displays the details of tape and DASD UCBs under TSO. Since then it has been modified, mainly to cater for changes in MVS hardware configuration. Changes include:

- UCBs may now be above or below the 16MB line; some products do not allow UCBs above the line, eg IMS (databases) and CA1 (TMC), and we have found that there is a restriction on STEPCAT or JOBCAT DDs in that the device they refer to must be below the line. This information can be very important in diagnosing such problems.
- The addition of 4-digit addresses.
- Dynamic changes to the configuration via HCD (and the provision of the 'UCBLOOK' and 'UCBSCAN' macros to examine UCBs).

The original QUERYCMD program I sent in has now been modified to use the new macros, provide information on UCB location and handle 3 or 4-digit addresses as input, as well as now displaying the controller type to which the DASD is connected (via a user-updated table). The code works with no problems on OS/390 R1.3. It must be authorized to use the UCBLOOK macro. It can be invoked using a piece of REXX (called 'Q'):

```
/* "Q" Exec */
Address "TSO"
Parse Upper Arg parms
"CALL 'TEST.LOADLIB(QUERYCMD)'" "'"parms"'"
```

A list of possible parameters is displayed if a '?' is passed to the program. Sample commands and displays are:

```
 Display DASD addresses 123 to 126

Q DA 123-126

-CUU-TYPE-STATUS-VOLSER--SIZE-ATTRIBUTE-UCB--SHAREABLE--DEVICE--
| 123 3390 ONLINE  TEST01   ( 3,339)  PRIVATE  A   NON-SHAREABLE   3990-A    |
| 124 3390 ONLINE  TEST02   ( 3,339)  PRIVATE  B   NON-SHAREABLE   3990-A    |
| 125 3380 ONLINE  TEST03   ( 1,770)  STORAGE  A   SHAREABLE       3990-A    |
| 126 3380 OFFLINE                                 SHAREABLE       3990-A    |
                                      -
```

Display DASD address 1CBD:

```
Q DA 1CBD

-CUU-TYPE-STATUS-VOLSER--SIZE-ATTRIBUTE-UCB--SHAREABLE--DEVICE--
|1CBD 3390 ONLINE  PROD10   ( 3,339)  PRIVATE   A   NON-SHAREABLE   RAMAC-1  |
-------------------------------------------------------------------
```

Display DASD volume WORK01:

```
Q DA WORK01

-CUU-TYPE-STATUS-VOLSER--SIZE-ATTRIBUTE-UCB--SHAREABLE--DEVICE--
| 8F1 3390 ONLINE  WORK01   ( 3,339)  PRIVATE   A   NON-SHAREABLE  ICEBERG1  |
-------------------------------------------------------------------
```

Display DASD address 1CBD:

```
Q DA 1CBD

-CUU-TYPE-STATUS-VOLSER--SIZE-ATTRIBUTE-UCB--SHAREABLE--DEVICE--
|1CBD 3390 ONLINE  PROD10   ( 3,339)  PRIVATE   A   NON-SHAREABLE   3990-A   |
-------------------------------------------------------------------
```

Display all on-line tape UCBs:

```
Q TA ONLINE

-CUU-TYPE-STATUS-VOLSER-------------------------------
| 180 3420 ONLINE  123456                              |
| 280 3480 ONLINE  500001                              |
| 281 3480 ONLINE                                      |
| 380 3490 ONLINE                                      |
| 381 3490 ONLINE  500002                              |
----------------------------------------------------
```

QUERYCMD

```
         TITLE 'QUERYCMD - QUERY DASD/TAPE DEVICES'
*********************************************************************
* QUERYCMD: SCAN ALL UCBS, CHECKING FOR EITHER DASD OR TAPES, AND    *
*           DISPLAY INFO ON TSO.                                     *
*                                                                    *
* PARMS:    PASSED VIA 'PARM=...'.                                   *
*                                                                    *
*           POSSIBLE FORMS ARE DESCRIBED IN THE 'HELPMSG' MESSAGES   *
*           AT THE END OF THIS PROGRAM (DISPLAYED BY ENTERING '?'    *
*           AS THE ONLY PARM PASSED TO THE PROGRAM).                 *
*                                                                    *
* OUTPUT:   DISPLAY VIA 'TPUT' AND RETURN CODE -                     *
*                                                                    *
```

```
*           Ø  -  OK                                                  *
*           4  -  REQUESTED CUU OR VOLID NOT FOUND                    *
*           8  -  PARMS NOT PASSED, INCORRECT LENGTH OR INVALID       *
*                                                                     *
***********************************************************************
         PRINT NOGEN
***********************************************************************
* HOUSEKEEPING...                                                     *
***********************************************************************
QUERYCMD CSECT
QUERYCMD AMODE 31
QUERYCMD RMODE 24
         BAKR  R14,Ø                    SAVE CALLER DATA ON STACK
         LR    R12,R15                  GET ENTRY POINT
         LA    R11,2Ø48(R12)            LOAD SECOND BASE
         LA    R11,2Ø48(R11)            LOAD SECOND BASE
         USING QUERYCMD,R12,R11         ADDRESSABILITY
         L     R2,Ø(R1)                 GET ADDR OF PARM
         STLINENO LINE=1                ENSURE DISPLAY STARTS @ LINE1
***********************************************************************
* SCAN THE PARMS PASSED. FIRST CHECK IS TO SEE IF TOO MUCH/TOO LITTLE *
* HAS BEEN PASSED.                                                    *
***********************************************************************
         CLC   Ø(2,R2),=H'1'            NONE AT ALL?
         BE    GIVEHELP                 YES..NEED A HAND
         BL    BADPARM                  LESS..CAN'T BE GOOD
         CLC   Ø(2,R2),=H'12'           TOO MUCH?
         BH    BADPARM                  YES..NOT RIGHT EITHER
         LR    R3,R2                    LOAD PARM ADDRESS
         LA    R3,2(R3)                 JUMP PAST LENGTH
         LH    R2,Ø(R2)                 GET LENGTH
         XR    R4,R4                    CLEAR COUNT
         LA    R5,PRM1                  POINT TO FIRST PARM WORKAREA
***********************************************************************
* CHECK THAT FIRST PARM IS 'DA' OR 'TA'...                            *
***********************************************************************
PRMLOOP1 DS    ØH
         CLI   Ø(R3),C' '               END OF PARM?
         BE    PARM1                    YES..CHECK IT OUT
         LA    R4,1(R4)                 BUMP COUNT
         CH    R4,=H'2'                 TOO LONG?
         BH    BADPARM                  YES..
         MVC   Ø(1,R5),Ø(R3)            NO...SAVE CHARACTER
         LA    R3,1(R3)                 BUMP SCAN FIELD
         LA    R5,1(R5)                 BUMP OUTPUT FIELD
         BCT   R2,PRMLOOP1              CONTINUE SCAN...
         OI    SWITCH,ALLUCBS           MUST WANT ALL DASD OR TAPE
         MVC   MSG6(MSG6L),MSG6A        CHANGE HEADING LINE
PARM1    DS    ØH
         CLC   PRM1,DA                  DASD?
         BE    PRM1DA
         CLC   PRM1,TA                  TAPE?
```

```
         BNE   BADPARM
PRM1TA   DS    ØH
         MVI   UCBCHECK+1,UCB3TAPE      SET UP TO LOOK FOR TAPE
         TM    SWITCH,ALLUCBS           WANT ALL TAPES?
         BO    GETPRM2                  YES..
         MVC   MSG6(MSG6L),MSG6B        NO...CHANGE HEADING LINE
         B     GETPRM2
PRM1DA   DS    ØH
         MVI   UCBCHECK+1,UCB3DACC      SET UP TO LOOK FOR DASD
         OI    SWITCH,GETDASD           SHOW WE'RE LOOKING FOR DASD
*********************************************************************
* NOW GET SECOND PARM (IF ANY) AND CHECK ITS FORMAT. THIS COULD BE:   *
*     - CUU        SINGLE CUU (3 DIGIT ADDRESS)                       *
*     - CCUU       SINGLE CUU (4 DIGIT ADDRESS)                       *
*     - VOLSER     SINGLE VOLSER                                      *
*     - VOL*       GENERIC VOLSER                                     *
*     - CUU-CUU    RANGE OF CUU VALUES (3 DIGIT ADDRESSES)            *
*     - CCUU-CCUU  RANGE OF CUU VALUES (4 DIGIT ADDRESSES)            *
*     - ONLINE     ALL ONLINE UNITS                                   *
* IF NO SECOND PARM IS PASSED, THE DEFAULT IS TO LIST ALL UCBS FOR    *
* DASD OR TAPE (AS SPECIFIED IN THE FIRST PARM).                      *
*********************************************************************
GETPRM2  DS    ØH
         TM    SWITCH,ALLUCBS           ANY MORE PARMS?
         BO    SCANUCBS                 NO...DISPLAY THE LOT
         BCTR  R2,Ø                     TAKE 1 OFF LENGTH FOR SPACE
         XR    R4,R4                    CLEAR COUNT
         LA    R5,PRM2                  POINT TO SECOND PARM WORKAREA
         LA    R3,1(R3)                 BUMP PAST SPACE
PRMLOOP2 DS    ØH
         CLI   Ø(R3),C' '               END OF PARM?
         BE    TOOMANY                  YES..MUST BE TOO MANY PARMS
         LA    R4,1(R4)                 BUMP COUNT
         CH    R4,=H'9'                 TOO LONG?
         BH    BADPARM                  YES...
         MVC   Ø(1,R5),Ø(R3)            NO...SAVE CHARACTER
         LA    R3,1(R3)                 BUMP SCAN FIELD
         LA    R5,1(R5)                 BUMP OUTPUT FIELD
         BCT   R2,PRMLOOP2              CONTINUE SCAN...
*********************************************************************
* NOW TRY AND WORK OUT THE FORMAT OF THE PARM (CUU, ONLINE, ETC)...   *
*********************************************************************
PARM2    DS    ØH
         CH    R4,=H'7'                 WAS IT 'CUU-CUU'?
         BE    CHKRNGE1                 ...MAYBE...
         CH    R4,=H'9'                 WAS IT 'CCUU-CCUU'?
         BE    CHKRNGE2                 ...MAYBE...
         CH    R4,=H'3'                 WAS IT CUU?
         BE    CHKCUU                   ...MAYBE...
         CH    R4,=H'4'                 WAS IT CCUU?
         BE    CHKCUU                   ...MAYBE...
*********************************************************************
```

```
* WE MUST ASSUME THAT ANYTHING ELSE IS EITHER A REQUEST FOR ON-LINE   *
* DEVICES OR A VOLID SPECIFICATION (FULLY-QUALIFIED OR GENERIC)...    *
************************************************************************
         CLC   PRM2(6),ONLINE          ON-LINE REQUEST?
         BNE   CHKVOLID                NO...MUST BE A VOLID
         OI    SWITCH,ONLIN            SHOW WE WANT ALL ON-LINE UNITS
         B     SCANUCBS                GO AND DISPLAY THEM
CHKVOLID DS    ØH
         BCTR  R5,Ø                    BACK UP 1 TO LAST CHAR IN PRM2
         CLI   Ø(R5),C'*'              GENERIC VOLID SPECIFIED?
         BE    GENVOLID                YES..
         OI    SWITCH,ONEVOLID         NO...SHOW JUST 1 VOLID REQUIRED
         MVC   VOLID,PRM2              SAVE VOLID TO LOOK FOR
         B     SCANUCBS                GO AND DISPLAY IT
GENVOLID DS    ØH
         OI    SWITCH,GENERIC          SHOW ITS A GENERIC REQUEST
         BCTR  R4,Ø                    TAKE 1 OFF LEN FOR '*'
         BCTR  R4,Ø                    TAKE 1 OFF LEN FOR 'EX' INSTR
         ST    R4,EXLENGTH             SAVE LENGTH FOR 'EX'
         MVC   VOLID,PRM2              SAVE VOLID FOR 'EX'
         B     SCANUCBS                GO AND DISPLAY THEM
************************************************************************
* SEE IF A VALID CUU WAS SPECIFIED. NOTE THAT IF IT ENDS IN '*' THEN  *
* IT MUST BE A GENERIC VOLID...                                       *
************************************************************************
CHKCUU   DS    ØH
         BCTR  R3,Ø                    BACK UP 1 TO LAST CHARACTER
         CLI   Ø(R3),C'*'              ENDS WITH '*'?
         BE    GENVOLID                YES..
         CH    R4,=H'4'                ALREADY 4 DIGIT?
         BE    CHKCUU2                 YES..JUST GO AND DO CHECK
         ICM   R1,15,PRM2              NO...GET CUU
         SRL   R1,8                    MOVE RIGHT 1 BYTE
         STCM  R1,15,PRM2              SAVE AS A 4 DIGIT ADDRESS
         OI    PRM2,X'FØ'              ENSURE VALID HEX IN 1ST BYTE
CHKCUU2  DS    ØH
         TRT   PRM2(4),TRTAB           VALID HEX?     (EG X'FØF2F3F4')
         BNZ   BADCUU                  NO...
SET1CUU  DS    ØH
         OI    SWITCH,ONECUU           SHOW WE JUST WANT THE ONE
         MVC   TRTAB+C'A'(6),LETTERS   MOVE X'ØA->ØF' INTO XLATE TABLE
         MVC   TRTAB+C'Ø'(1Ø),NUMBERS  MOVE X'ØØ->Ø9' INTO XLATE TABLE
         MVC   FWORD(4),PRM2           MOVE CCUU TO WORK FIELD
         BAL   R9,CONVCUU              CONVERT TO BINARY FOR COMPARES
         MVC   CUU,FWORD               SAVE BINARY VALUE OF CCUU
         B     SCANUCBS                GO AND DISPLAY IT
************************************************************************
* SEE IF A VALID CUU-CUU RANGE WAS ENTERED (3 DIGIT ADDRESSES)...     *
************************************************************************
CHKRNGE1 DS    ØH
         CLI   PRM2+3,C'-'             CORRECT FORMAT?
         BNE   BADPARM                 NO...
```

```
         TRT   PRM2(3),TRTAB           ENSURE VALID HEX (1ST CUU)
         BNZ   BADRANGE
         TRT   PRM2+4(3),TRTAB         ENSURE VALID HEX (2ND CUU)
         BNZ   BADRANGE
         MVC   TRTAB+C'A'(6),LETTERS   MOVE X'ØA->ØF' INTO XLATE TABLE
         MVC   TRTAB+C'Ø'(1Ø),NUMBERS  MOVE X'ØØ->Ø9' INTO XLATE TABLE
         MVC   FWORD+1(3),PRM2         SAVE 1ST CUU
         OI    FWORD,X'FØ'             SET LEADING ZERO
         BAL   R9,CONVCUU              CONVERT TO BINARY FOR COMPARES
         MVC   BINCUU1,FWORD           SAVE BINARY VALUE OF 1ST CUU
         MVC   FWORD+1(3),PRM2+4       SAVE 2ND CUU
         OI    FWORD,X'FØ'             SET LEADING ZERO
         BAL   R9,CONVCUU              CONVERT TO BINARY FOR COMPARES
         MVC   BINCUU2,FWORD           SAVE BINARY VALUE OF 2ND CUU
         B     CHKRNGE3                GO DO COMMON BIT
*********************************************************************
* SEE IF A VALID CCUU-CCUU RANGE WAS ENTERED (4 DIGIT ADDRESSES)...   *
*********************************************************************
CHKRNGE2 DS    ØH
         CLI   PRM2+4,C'-'             CORRECT FORMAT?
         BNE   BADPARM                 NO...
         TRT   PRM2(4),TRTAB           ENSURE VALID HEX (1ST CCUU)
         BNZ   BADRANGE
         TRT   PRM2+5(4),TRTAB         ENSURE VALID HEX (2ND CCUU)
         BNZ   BADRANGE
         MVC   TRTAB+C'A'(6),LETTERS   MOVE X'ØA->ØF' INTO XLATE TABLE
         MVC   TRTAB+C'Ø'(1Ø),NUMBERS  MOVE X'ØØ->Ø9' INTO XLATE TABLE
         MVC   FWORD(4),PRM2           SAVE 1ST CCUU
         BAL   R9,CONVCUU              CONVERT TO BINARY FOR COMPARES
         MVC   BINCUU1,FWORD           SAVE BINARY VALUE OF 1ST CCUU
         MVC   FWORD(4),PRM2+5         SAVE 2ND CCUU
         BAL   R9,CONVCUU              CONVERT TO BINARY FOR COMPARES
         MVC   BINCUU2,FWORD           SAVE BINARY VALUE OF 2ND CCUU
*********************************************************************
* SEE IF A VALID CUU-CUU RANGE WAS ENTERED (COMMON BIT)...            *
*********************************************************************
CHKRNGE3 DS    ØH
         L     R8,BINCUU1              GET 1ST CUU
         L     R9,BINCUU2              GET 2ND CUU
         CR    R8,R9                   SEE IF WE'VE GOT A VALID RANGE
         BH    BADRANGE                ...NO
         BE    THESAME                 THE SAME! TREAT AS ONLY 1 CUU
         OI    SWITCH,RANGE            SHOW WE WANT A RANGE
         B     SCANUCBS                GO AND SCAN THEM...
THESAME  DS    ØH
         OI    SWITCH,ONECUU           SHOW WE JUST WANT THE ONE
         MVC   CUU,BINCUU1             SAVE BINARY VALUE OF CCUU
         B     SCANUCBS                GO AND DISPLAY IT
*********************************************************************
* SCAN THROUGH THE UCBS, LOOKING FOR THE TYPE WE WANT...              *
*********************************************************************
SCANUCBS DS    ØH
```

```
         LA    R7,99                    FORCE HEADINGS
         USING UCBOB,R4                 ADDRESSABILITY TO UCB
         LA    R4,UCBAREA               +POINT TO UCB STORAGE AREA
         LA    R3,MSG3CUU1              POINT TO FIRST MSG FIELD
         XR    R5,R5                    SET COUNT OF ITEMS IN LINE
         XC    UCBWORK,UCBWORK          +INITIALIZE UCBSCAN WORKAREA
UCBLOOP  DS    ØH
*
         UCBSCAN COPY,                                                 X
               WORKAREA=UCBWORK,                                       X
               UCBAREA=UCBAREA,                                        X
               DCEAREA=NONE,                                           X
               DCELEN=Ø,                                               X
               VOLSER=NONE,             DON'T SELECT BY VOLSER         X
               DEVN=Ø,                  START WITH FIRST UCB           X
               DYNAMIC=YES,             INCLUDE DYNAMICALLY ADDED UCBS X
               RANGE=ALL,               4 AND 3-DIGIT UCBS             X
               NONBASE=NO,              NOT SURE WHAT THIS DOES        X
               DEVCLASS=ALL,            ALL DEVICE CLASSES             X
               DEVCID=Ø,                DON'T SELECT BY DEVICE CHAR.   X
               IOCTOKEN=NONE,           NO IODEVICE TABLE TOKEN        X
               LINKAGE=SYSTEM,          USE PC CALL                    X
               PLISTVER=MAX
*
         LTR   R15,R15                  GOT UCB OK?
         BZ    UCBCHECK                 YES..CHECK IT
         C     R15,=F'4'                END OF UCBS?
         BE    ENDUCBS                  YES..CLEAN UP, ETC
         B     BADCALL                  NO...SHOW RETURN/REASON CODES
*********************************************************************
* THE NEXT INSTRUCTION IS UPDATED TO INSERT THE CORRECT BYTE FOR DASD *
* OR TAPE, AS REQUESTED (AT LABEL 'PRM1TA' OR 'PRM1DA')...           *
*********************************************************************
UCBCHECK DS    ØH
         TM    UCBTBYT3,X'ØØ'           IS IT UCB TYPE WE WANT?
         BNO   UCBLOOP                  NO...GET NEXT ONE
*********************************************************************
* NOW WE'VE GOT A UCB OF THE TYPE WE WANT - IF WE WANT ALL THEN GO    *
* AND GET OUR INFO, OTHERWISE SEE IF IT IS THE CUU OR VOLID REQUIRED, *
* OR AN ONLINE ONE IF THAT IS WHAT WE ARE LOOKING FOR...              *
*********************************************************************
GOTUCB   DS    ØH
         TM    SWITCH,ALLUCBS           WANT ALL UNITS?
         BO    GETINFO                  YES..
         TM    SWITCH,ONLIN             WANT ONLY ON-LINE UNITS?
         BNO   CHK1VOL                  NO...MUST BE CUU/VOL OR GENERIC
         TM    UCBSTAT,UCBONLI          IS THIS ONE ON-LINE?
         BO    GETINFO                  YES..
         B     UCBLOOP                  NO...IGNORE IT
CHK1VOL  DS    ØH
         TM    SWITCH,ONEVOLID          WANT JUST ONE VOLID?
         BO    COMPVOL                  YES..
```

```
         TM    SWITCH,GENERIC           WANT GENERIC VOLIDS?
         BNO   COMPCUU                  NO...
         L     R9,EXLENGTH              YES..GET LENGTH FOR COMPARE
         EX    R9,EXCLC                 ONE OF THE VOLIDS WE WANT?
         BNE   UCBLOOP                  NO...
         B     GETINFO                  YES..
COMPCUU  DS    ØH
         TM    SWITCH,RANGE             RANGE REQUESTED?
         BNO   CLCCUU                   NO...SEE IT WE WANT THIS ONE
         LH    R8,UCBCHAN               GET VALUE OF UCB CCUU
         L     R9,BINCUU1               GET VALUE OF LOW CCUU IN RANGE
         CR    R8,R9                    LOWER?
         BL    UCBLOOP                  YES..FORGET IT
         L     R9,BINCUU2               GET VALUE OF HIGH CCUU IN RANGE
         CR    R8,R9                    HIGHER?
         BH    UCBLOOP                  YES..FORGET IT
         B     GETINFO                  MUST BE IN OUR RANGE
CLCCUU   DS    ØH
         CLC   CUU+2(2),UCBCHAN         IS IT THE CUU WE WANT?
         BNE   UCBLOOP                  NO...
         B     GETINFO                  YES..
COMPVOL  DS    ØH
         CLC   UCBVOLI(6),VOLID         IS IT THE VOLID WE WANT?
         BNE   UCBLOOP                  NO...
*********************************************************************
* GET THE DEVICE CHARACTERISTICS FOR DISPLAY. WE WON'T GO OVERBOARD,  *
* JUST A FEW SIMPLE CHECKS...                                        *
* NEW CODE ADDED Ø1/Ø2/94 TO GET THE NUMBER OF CYLINDERS ON THE       *
* DEVICE, BUT *ONLY* IF WE'RE GETTING ON-LINE DASD OR JUST ONE        *
* DASD VOLID (AS THOSE ARE THE ONLY DISPLAYS WITH ENOUGH SPACE        *
* TO FIT THIS EXTRA INFORMATION IN).                                 *
* NB AS WE GET THE FORMAT4 FROM THE VTOC THERE IS SOMETIMES AN        *
* EXTRA (CE?) CYLINDER ADDED ON - WE WILL TAKE THIS OFF IF WE CAN     *
* RECOGNIZE THAT IT IS THERE (EG 886 ON A 338Ø 'D').                 *
*********************************************************************
GETINFO  DS    ØH
         TM    SWITCH,ALLUCBS           LOOKING FOR ALL VOLUMES?
         BO    CARRYON                  YES..SKIP THIS BIT
         TM    SWITCH,GETDASD           LOOKING FOR DASD?
         BNO   CARRYON                  NO...
         TM    UCBSTAT,UCBONLI          IS THIS ONE ON-LINE?
         BNO   CARRYON                  NO...
         MVC   27(8,R3),=C'(?????) '    YES..SET UP DEFAULT SIZE
*
         LSPACE UCB=(R4),               GET THE FORMAT4 DSCB...        X
               F4DSCB=F4DSCB,                                          X
               MSG=F4ERRMSG             PLACE ANY ERROR MSG IN HERE
*
         LTR   R15,R15                  LSPACE WORKED OK?
         BZ    CHKCYLS                  YES..
         ST    R15,FWORD                NO...SAVE RETURN CODE
         TM    SWITCH,MSGSENT           ALREADY DISPLAYED ERROR TEXT?
```

```
         BO    SHOWRC                   YES..
         OI    SWITCH,MSGSENT           NO...SET SO WE DON'T REPEAT IT
         TPUT  F4ERRMSG,3Ø              DISPLAY ERROR TEXT
SHOWRC   DS    ØH
         UNPK  DWORD(3),FWORD+3(2)      UNPACK RETURN CODE + 1 BYTE
         TR    DWORD(2),HEXTAB-24Ø      XLATE TO PRINTABLE HEX
         MVC   26(5,R3),=C'RC=X"'       SET UP CONSTANT
         MVC   31(2,R3),DWORD           MOVE IN RC
         MVI   33(R3),C'"'
         B     CARRYON                  IGNORE REST OF THIS BIT
CHKCYLS  DS    ØH
         MVC   27(7,R3),=X'4Ø2Ø2Ø6B2Ø212Ø'  YES..MOVE IN EDIT PATTERN
         LH    R1,F4DSCB+18             GET NUMBER OF CYLINDERS
         CH    R1,=H'886'               886 CYLS (338Ø 'D')?
         BE    TAKE1OFF                 YES..
         CH    R1,=H'1771'              1771 CYLS (338Ø 'E')?
         BE    TAKE1OFF                 YES..
         CH    R1,=H'2656'              2656 CYLS (338Ø 'K')?
         BE    TAKE1OFF                 YES..
         CH    R1,=H'1114'              1114 CYLS (339Ø M1)?
         BE    TAKE1OFF                 YES..
         CH    R1,=H'2227'              2227 CYLS (339Ø M2)?
         BE    TAKE1OFF                 YES..
         CH    R1,=H'334Ø'              3340 CYLS (339Ø M3)?
         BE    TAKE1OFF                 YES..
         CH    R1,=H'1ØØ18'             1ØØ18 CYLS (339Ø M9)?
         BE    TAKE1OFF                 YES..
         B     GETCYLS                  NO...
TAKE1OFF DS    ØH
         BCTR  R1,Ø                     THERE IT GOES...
GETCYLS  DS    ØH
         CVD   R1,DWORD                 CONVERT TO DECIMAL
         ED    27(7,R3),DWORD+5         EDIT IN NUMBER OF CYLINDERS
         MVI   27(R3),C'('              MAKE IT LOOK PRETTY
         MVI   34(R3),C')'
CARRYON  DS    ØH
         TM    UCBFL1,UCBBOX            BOXED?
         BO    ITSBOXED                 YES..
         TM    UCBSTAT,UCBALOC          ALLOCATED?
         BO    ITSALLOC                 YES..
         TM    UCBSTAT,UCBONLI          ON-LINE?
         BO    ITSONLIN                 YES..
         B     ITSOFLIN                 NO...LET'S CALL IT OFF-LINE
ITSONLIN DS    ØH
         MVC   1Ø(7,R3),ONLINE          SET UP MSG
         B     GETVOLID                 GO AND GET VOLID
ITSOFLIN DS    ØH
         MVC   1Ø(7,R3),OFFLINE         SET UP MSG
         B     GETVOLID                 GO AND GET VOLID
ITSBOXED DS    ØH
         MVC   1Ø(7,R3),BOXED           SET UP MSG
         B     GETDEVTP                 GO AND GET DEVICE TYPE
```

```
ITSALLOC DS    ØH
         MVC   1Ø(7,R3),ALLOC          SET UP MSG
         B     GETVOLID                GO AND GET THE VOLID
GETVOLID DS    ØH
         MVC   18(6,R3),UCBVOLI        MOVE VOLID TO MSG LINE
*********************************************************************
* ADD CODE HERE FOR ANY NEW DEICE TYPES...                          *
*********************************************************************
GETDEVTP DS    ØH
         MVC   5(4,R3),QUERIES         SET UP UNKNOWN DEVTYPE
         CLI   UCBTBYT4,X'ØE'          338Ø? DASD
         BE    SET338Ø                 YES..
         CLI   UCBTBYT4,X'ØF'          339Ø? DASD
         BE    SET339Ø                 YES..
         CLI   UCBTBYT4,UCB34ØØ        342Ø? TAPE
         BE    SET342Ø                 YES..
         CLI   UCBTBYT4,UCB348Ø        348Ø? CART
         BE    SET348Ø                 YES..
         CLI   UCBTBYT4,UCB349Ø        349Ø? CART
         BE    SET349Ø                 YES..
         B     DISPLAY                 NO...LEAVE AS '????'
SET338Ø  DS    ØH
         MVC   5(4,R3),=C'338Ø'
         B     CHKUCBS                 YES..
SET339Ø  DS    ØH
         MVC   5(4,R3),=C'339Ø'
CHKUCBS  DS    ØH                      YES..
         TM    SWITCH,ALLUCBS          DOING ALL UCBS?
         BO    DISPLAY                 YES..SKIP EXTRA INFO BIT
         MVC   52(13,R3),NONSHARE      DEFAULT TO NON-SHAREABLE
         TM    UCBTBYT2,UCBRR          SHAREABLE?
         BNO   CHECKPRI                NO...
         MVC   52(13,R3),SHARE         YES..SET TO SHAREABLE
CHECKPRI DS    ØH
         TM    UCBSTAB,UCBBPRV         PRIVATE?
         BNO   CHECKPUB                NO...
         MVC   37(7,R3),PRIVATE        YES..SHOW THAT IN MSG
         BAL   R9,LOCUCB               SEE WHERE UCB IS...
         B     CHECKCTL
CHECKPUB DS    ØH
         TM    UCBSTAB,UCBBPUB         PUBLIC?
         BNO   CHECKSTR                NO...
         MVC   37(7,R3),PUBLIC         YES..SHOW THAT IN MSG
         BAL   R9,LOCUCB               SEE WHERE UCB IS...
         B     CHECKCTL
CHECKSTR DS    ØH
         TM    UCBSTAB,UCBBSTR         STORAGE?
         BNO   CHECKCTL
         MVC   37(7,R3),STORAGE        YES..SHOW THAT IN MSG
         BAL   R9,LOCUCB               SEE WHERE UCB IS...
CHECKCTL DS    ØH
         XC    FWORD2,FWORD2           CLEAR WORK REG
```

```
         MVC   FWORD2+2(2),UCBCHAN      DEVICE ADDRESS TO LOOK FOR
         MVC   CTLUNIT,CTLNFND          SET DEFAULT CTLUNIT
         BAL   R9,FINDCTL               GO AND FIND CTLUNIT
         MVC   67(8,R3),CTLUNIT         SET CTLUNIT
         B     DISPLAY
SET342Ø  DS    ØH
         MVC   4(4,R3),=C'342Ø'
         B     DISPLAY
SET348Ø  DS    ØH
         MVC   5(4,R3),=C'348Ø'
         B     DISPLAY
SET349Ø  DS    ØH
         MVC   5(4,R3),=C'349Ø'
*********************************************************************
* DISPLAY THE LINE (IF ITS FULL)...                                 *
*********************************************************************
DISPLAY  DS    ØH
         UNPK  UNPKFLD(5),UCBCHAN(3)    UNPACK HEX CUU + 1 CHAR
         TR    UNPKFLD(4),TRTAB2-24Ø    MAKE PRINTABLE HEX
         CLI   UNPKFLD,C'Ø'             LEADING ZERO?
         BNE   DISPLAY2                 NO...
         MVI   UNPKFLD,C' '             YES..BLANK OUT
DISPLAY2 DS    ØH
         MVC   Ø(4,R3),UNPKFLD          MOVE CUU TO MSG LINE
         TM    SWITCH,ALLUCBS           DISPLAYING ALL UCBS?
         BNO   DISPLAY3                 NO...
         LA    R5,1(R5)                 BUMP COUNTER
         LA    R3,26(R3)                BUMP TO NEXT DISPLAY SLOT
         C     R5,=F'3'                 FULL LINE YET?
         BNE   UCBLOOP                  NO...GET NEXT UCB
         B     DISPLAY5                 YES..DISPLAY THE LINE
DISPLAY3 DS    ØH
         MVI   MSG3LIN2,C' '            CLEAR OUT '|'S FOR SINGLE UNIT
         MVI   MSG3LIN3,C' '
DISPLAY5 DS    ØH
         BAL   R9,FIRSTLIN              SEE IF WE WANT A HEADING
         OI    SWITCH,FOUND1            SHOW WE FOUND AT LEAST ONE
         TPUT  MSG3,MSG3L               DISPLAY INFO
         TM    SWITCH,ONECUU+ONEVOLID   DOING ONE VOLI/ONE CUU?
         BNZ   DOTRAIL                  NO...THAT'S IT THEN...
         MVI   MSG3,C' '                YES..CLEAR OUT LINE
         MVC   MSG3+1(MSG3L-1),MSG3
         MVI   MSG3LIN1,C'|'            PUT BACK THE '|'S
         MVI   MSG3LIN2,C'|'
         MVI   MSG3LIN3,C'|'
         MVI   MSG3LIN4,C'|'
         XR    R5,R5                    RESET COUNTER
         LA    R3,MSG3CUU1              AND POINTER
         B     UCBLOOP                  AND GET NEXT UCB
*********************************************************************
* END OF UCBS - DISPLAY LAST LINE IF NECESSARY...                   *
```

```
*********************************************************************
ENDUCBS  DS    ØH
         TM    SWITCH,ONEVOLID          LOOKING FOR ONE VOLID?
         BO    NOVOLID                  YES..CAN'T HAVE FOUND IT THEN
         TM    SWITCH,ONECUU            LOOKING FOR JUST ONE CUU?
         BO    NOCUU                    YES..CAN'T HAVE FOUND IT THEN
         TM    SWITCH,FOUND1            FOUND ANYTHING FOR OTHER OPTS?
         BO    LASTONE                  YES..SKIP NEXT BIT
         TM    SWITCH,RANGE             LOOKING FOR CUU RANGE?
         BO    NOCUURNG                 YES..CAN'T HAVE FOUND ANY
         TM    SWITCH,GENERIC           LOOKING FOR GENERIC VOLIDS?
         BO    NOGENFND                 YES..CAN'T HAVE FOUND ANY
LASTONE  DS    ØH
         CLC   MSG3CUU1(3),=C'   '      ANYTHING TO DISPLAY?
         BE    DOTRAIL                  NO...SKIP FINAL DISPLAY LINE
         TPUT  MSG3,MSG3L               YES..DISPLAY WHAT'S LEFT
DOTRAIL  DS    ØH
         MVI   MSG6,C'-'
         MVC   MSG6+1(MSG6L-1),MSG6
         TPUT  MSG6,MSG6L               DISPLAY TRAILER LINE
*********************************************************************
* RETURN TO CALLER WITH RELEVANT RETURN CODE...                     *
*********************************************************************
RETURN   DS    ØH
         L     R15,RETC                 LOAD RETURN CODE
         PR    ,                        RESTORE CALLER DATA, RETURN
*********************************************************************
* INVALID OR NO PARMS PASSED...                                     *
*********************************************************************
BADPARM  DS    ØH
         TPUT  MSG1,MSG1L               TELL USER PARM IS BAD/MISSING
         B     SETRC8
*********************************************************************
* CUU SPECIFIED WAS NOT VALID HEX...                                *
*********************************************************************
BADCUU   DS    ØH
         MVC   MSG2TEXT,PRM2            SET CUU IN MSG
         TPUT  MSG2,MSG2L               TELL USER CUU IS BAD
         B     SETRC8
*********************************************************************
* VOLID SPECIFIED WAS NOT FOUND...                                  *
*********************************************************************
NOVOLID  DS    ØH
         MVC   MSG5TEXT,VOLID           SET VOLID IN MSG
         TPUT  MSG5,MSG5L               NO MATCHING VOLID FOUND
         MVC   RETC,=F'4'               SET RC=4
         B     RETURN
*********************************************************************
* CUU SPECIFIED WAS NOT FOUND...                                    *
*********************************************************************
NOCUU    DS    ØH
         MVC   MSG4TEXT,PRM2            SET CUU IN MSG
```

```
         TPUT  MSG4,MSG4L               NO MATCHING UCB FOUND
         MVC   RETC,=F'4'               SET RC=4
         B     RETURN
*************************************************************************
* TOO MANY PARMS PASSED...                                              *
*************************************************************************
TOOMANY  DS    ØH
         TPUT  MSG7,MSG7L               TOO MANY
         B     SETRC8
*************************************************************************
* INVALID CUU RANGE PASSED...                                           *
*************************************************************************
BADRANGE DS    ØH
         MVC   MSG8TEXT,PRM2            SET CUU-CUU IN MSG
         TPUT  MSG8,MSG8L               BAD RANGE
         B     SETRC8
*************************************************************************
* NO MATCHES FOUND FOR A CUU-CUU RANGE...                               *
*************************************************************************
NOCUURNG DS    ØH
         MVC   MSG9TEXT,PRM2            SET CUU-CUU IN MSG
         TPUT  MSG9,MSG9L               NO RANGE FOUND
         B     SETRC8
*************************************************************************
* NO MATCHES FOUND FOR A GENERIC VOLID...                               *
*************************************************************************
NOGENFND DS    ØH
         MVC   MSGATEXT,VOLID           SET GENERIC VOLID IN MSG
         TPUT  MSGA,MSGAL               NO VOLID(S) FOUND
         B     SETRC8
*************************************************************************
* BAD RETURN CODE FROM CALL TO 'UCBSCAN'...                             *
*************************************************************************
BADCALL  DS    ØH
         ST    R15,RETCD                SAVE RETURN CODE FROM UCBSCAN
         ST    RØ,REASN                 SAVE REASON CODE FROM UCBSCAN
         UNPK  UNPKFLD(3),RETCD+3(2)    UNPK RETURN CODE + 1 BYTE
         TR    UNPKFLD(2),TRTAB2-24Ø    XLATE TO PRINTABLE HEX
         MVC   MSGBTXT1,UNPKFLD         MOVE RETURN CODE TO MSG AREA
         UNPK  UNPKFLD(3),REASN+3(2)    UNPK REASON CODE + 1 BYTE
         TR    UNPKFLD(2),TRTAB2-24Ø    XLATE TO PRINTABLE HEX
         MVC   MSGBTXT2,UNPKFLD         MOVE REASON CODE TO MSG AREA
         TPUT  MSGB,MSGBL               SHOW CODES...
         B     SETRC8
*************************************************************************
* SET RC=8...                                                           *
*************************************************************************
SETRC8   DS    ØH
         MVC   RETC,=F'8'               SET RC=8
         B     RETURN
*************************************************************************
* DISPLAY HELP INFO (IF PARM WAS A '?')...                              *
```

```
*************************************************************************
GIVEHELP DS    ØH
         CLI   2(R2),C'?'              WAS HELP REQUESTED?
         BNE   BADPARM                 NO...
         STLINENO LINE=1               ENSURE DISPLAY STARTS @ LINE1
         TPUT  HELPMSG1,5Ø             DISPLAY HELP INFO
         TPUT  HELPMSG2,5Ø
         TPUT  HELPMSG3,1
         TPUT  HELPMSG4,5Ø
         TPUT  HELPMSG3,1
         TPUT  HELPMSG5,5Ø
         TPUT  HELPMSG6,5Ø
         TPUT  HELPMSG7,5Ø
         TPUT  HELPMSG8,5Ø
         TPUT  HELPMSG9,5Ø
         TPUT  HELPMSGA,5Ø
         TPUT  HELPMSGB,5Ø
         TPUT  HELPMSGC,5Ø
         TPUT  HELPMSGD,5Ø
         TPUT  HELPMSGE,5Ø
         TPUT  HELPMSG3,1
         TPUT  HELPMSGF,5Ø
         TPUT  HELPMSGG,5Ø
         B     RETURN
*************************************************************************
*                   + + S U B R O U T I N E + + +                       *
* CONVERT CHARACTER CUU (EG 'Ø94F') INTO ITS BINARY EQUIVALENT. THIS    *
* IS SO THAT VALID RANGE COMPARISONS CAN BE MADE IF A RANGE OF CUUS     *
* HAS BEEN REQUESTED. THE ROUTINE USES 4-DIGIT ADDRESSES, PADDED WITH   *
* A LEADING 'Ø', IF REQUIRED.                                           *
*************************************************************************
CONVCUU  DS    ØH
         TR    FWORD(4),TRTAB          CONV. C'A->F' INTO X'A->F'
         XC    DWORD,DWORD             CLEAR OUT WORKAREA
         PACK  DWORD+4(4),FWORD(5)     REMOVE ZONES
         L     R8,DWORD+4              LOAD 'ØØCCUUØØ'
         SRL   R8,8                    SHIFT OUT TRAILING 'ØØ'
         ST    R8,FWORD                SAVE BINARY CUU VALUE
         BR    R9                      RETURN FROM SUBROUTINE
*************************************************************************
*                   + + S U B R O U T I N E + + +                       *
* SEE IF WE NEED HEADINGS... WE WILL USE 21 LINES AS A SCREENFUL.       *
*************************************************************************
FIRSTLIN DS    ØH
         CH    R7,=H'21'               TIME FOR HEADINGS YET?
         BH    SETLINE                 YES..
         LA    R7,1(R7)                NO...BUMP LINE COUNT
         BR    R9                      RETURN FROM ROUTINE
SETLINE  DS    ØH
         TPUT  MSG6,MSG6L              DISPLAY HEADING LINE
         LA    R7,1                    RESET LINE COUNT
         BR    R9                      RETURN FROM ROUTINE
```

```
*********************************************************************
*                    + + S U B R O U T I N E + + +                  *
* SEE IF UCB IS ABOVE ('A') OR BELOW ('B') THE 16MEG LINE. NOTE THAT  *
* THERE IS ONLY AN EXTENSION FOR UCBS IF THEY ARE 'BELOW THE LINE',   *
* SO THAT ANY UCB WITHOUT THE EXTENSION IS DEEMED TO BE ABOVE.        *
*********************************************************************
LOCUCB   DS    ØH
         MVI   48(R3),C'?'              DEFAULT IS "DON'T KNOW"
         MODESET MF=(E,SUPMODE)         ENTER SUPERVISOR MODE
         UCBLOOK DEVN=UCBCHAN,          LOOK BY DEVICE ADDRESS         X
               UCBPTR=FWORD,            TO HOLD A(UCB COMMON SEGMENT)  X
               DYNAMIC=YES,             INCLUDE DYNAMIC UCBS           X
               RANGE=ALL,               3 AND 4 DIGIT UCBS             X
               NOPIN,                   DON'T PIN UCB                  X
               LOC=ANY                  ABOVE AND BELOW THE LINE
         LTR   R15,R15                  SUCCESSFUL?
         BNZ   RESET                    NO...LEAVE AS DEFAULT
         MODESET MF=(E,PROBMODE)        RETURN TO PROBLEM MODE
         MVI   48(R3),C'A'              DEFAULT IS 'A'BOVE
         L     R1,FWORD                 GET UCB ADDRESS
         C     R1,=F'16777216'          ABOVE 16M?
         BHR   R9                       YES..RETURN
         MVI   48(R3),C'B'              NO...MAKE IT 'B'ELOW
         BR    R9                       RETURN FROM ROUTINE
RESET    DS    ØH
         MODESET MF=(E,PROBMODE)        RETURN TO PROBLEM MODE
         BR    R9                       RETURN FROM ROUTINE
*********************************************************************
*                    + + S U B R O U T I N E + + +                  *
* FIND WHICH CONTROL UNIT THE ADDRESS IS ON: ICEBERG1, RAMAC1, EMC-1, *
* 399Ø-A, ETC...                                                      *
*********************************************************************
FINDCTL  DS    ØH
         LA    R1,CTLRTAB               LOCATE CTLUNIT TABLE
         LA    R2,CTLENTS               NUMBER OF ENTRIES
         L     R1Ø,FWORD2               GET BINARY CUU VALUE
FINDLOOP DS    ØH
         L     R6,Ø(R1)                 GET LOW RANGE ADDRESS
         CR    R1Ø,R6                   CUU EQUAL?
         BL    FINDBUMP                 LOW - NOT IN RANGE, TRY NEXT
         L     R6,4(R1)                 GET HIGH RANGE ADDRESS
         CR    R1Ø,R6                   CUU EQUAL?
         BH    FINDBUMP                 HIGH - NOT IN RANGE, TRY NEXT
         MVC   CTLUNIT,8(R1)            IN RANGE - SAVE CTLUNIT NAME
         BR    R9                       RETURN FROM ROUTINE
FINDBUMP DS    ØH
         LA    R1,16(R1)                BUMP TO NEXT ENTRY
         BCT   R2,FINDLOOP              KEEP LOOKING
         BR    R9                       RETURN FROM ROUTINE
         EJECT
*———————————————————————————————*
*
```

```
         LTORG                           LITERAL POOL
OFFLINE  DC    CL7'OFFLINE'
ONLINE   DC    CL7'ONLINE '
BOXED    DC    CL7'BOXED  '
ALLOC    DC    CL7'ALLOC  '
PRIVATE  DC    CL7'PRIVATE'
PUBLIC   DC    CL7'PUBLIC '
STORAGE  DC    CL7'STORAGE'
SHARE    DC    CL13'SHAREABLE'
NONSHARE DC    CL13'NON-SHAREABLE'
HEXTAB   DC    C'0123456789ABCDEF'
LETTERS  DC    X'0A0B0C0D0E0F'
NUMBERS  DC    X'000102030405060708090'
BINCUU1  DS    F
BINCUU2  DS    F
FWORD    DS    F
FWORD2   DS    F
EXLENGTH DS    F
DWORD    DS    D
EXCLC    CLC   UCBVOLI(0),VOLID
F4DSCB   DS    CL96
F4ERRMSG DS    CL30
QUERIES  DC    CL4'????'
DA       DC    CL3'DA '
TA       DC    CL3'TA '
CUU      DS    CL4
VOLID    DS    CL6
PRM1     DC    CL2' '
PRM2     DC    CL9' '
SWITCH   DC    X'00'                   SWITCH FIELD
FOUND1   EQU   X'01'
GENERIC  EQU   X'02'
RANGE    EQU   X'04'
GETDASD  EQU   X'08'
MSGSENT  EQU   X'04'
ALLUCBS  EQU   X'10'
ONECUU   EQU   X'20'
ONEVOLID EQU   X'40'
ONLIN    EQU   X'80'
RETC     DS    F
UCBAREA  DS    XL48                    HOLDS UCB COMMON & DEV SEGS
UCBWORK  DS    XL100                   UCBSCAN WORKAREA
UNPKFLD  DS    CL5
RETCD    DS    F
REASN    DS    F
SUPMODE  MODESET KEY=ZERO,MODE=SUP,MF=L
PROBMODE MODESET KEY=NZERO,MODE=PROB,MF=L
*
*               0 1 2 3 4 5 6 7 8 9 A B C D E F
TRTAB    DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 0
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 1
```

```
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 2
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 3
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 4
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 5
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 6
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 7
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 8
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 9
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' A
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' B
         DC    X'FF000000000000FFFFFFFFFFFFFFFFFF' C  (ABCDEF)
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' D
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' E
         DC    X'00000000000000000000FFFFFFFFFFFF' F  (0123456789)
TRTAB2   DC    CL16'0123456789ABCDEF'
*************************************************************************
*                                                                       *
* THIS TABLE HOLDS THE RANGE OF ADDRESSES FOR EACH CONTROL UNIT (IN     *
* ITS DECIMAL EQUIVALENT). THE 'USER DESCRIPTION' IS WHAT NAME IS       *
* GIVEN LOCALLY TO THE CONTROLLER (TO DIFFERENTIATE IT FROM OTHERS).    *
*                                                                       *
* THIS CAN BE USEFUL TO EASILY IDENTIFY WHICH CONTROLLER A PARTICULAR   *
* DISK IS ATTACHED TO, ESPECIALLY IN AN ENVIRONMENT WITH A MIXTURE OF   *
* DIFFERENT CONTROLLERS, FROM DIFFERENT SUPPLIERS.                      *
*                                                                       *
* AS SOMETIMES THE ADDRESSES CAN BE DIFFERENT ON DIFFERENT MVS IMAGES   *
* THEY ARE ADDED IN BOTH THEIR 3- AND 4-DIGIT REPRESENTATIONS.          *
*                                                                       *
*************************************************************************
*               LOW       HIGH     USER      HEX RANGE
*              RANGE     RANGE  DESCRIPTION
CTLRTAB  DS    0F
         DC    F'02560',F'02687',C'ICEBERG1'  0A00-0A7F ICEBERG1 3DIGIT
         DC    F'06656',F'06783',C'ICEBERG1'  1A00-1A7F ICEBERG1 4DIGIT
         DC    F'02752',F'02799',C' RAMAC1 '  0AC0-0AEF RAMAC1   3DIGIT
         DC    F'06848',F'06895',C' RAMAC1 '  1AC0-1AEF RAMAC1   4DIGIT
         DC    F'02112',F'02191',C' 3990-A '  0840-088F 3990-A   3DIGIT
         DC    F'06208',F'06287',C' 3990-A '  1840-188F 3990-A   4DIGIT
         DC    F'02816',F'03071',C' HDS-1  '  0B00-0BFF HDS-1    3DIGIT
         DC    F'06912',F'07167',C' HDS-1  '  1B00-1BFF HDS-1    4DIGIT
         DC    F'03072',F'03327',C' 3990-B '  0C00-0CFF 3990-B   3DIGIT
         DC    F'07168',F'07423',C' 3990-B '  1C00-1CFF 3990-B   4DIGIT
         DC    F'01024',F'01279',C' H.D.S. '  0400-04FF HDS      3DIGIT
         DC    F'00256',F'00263',C'  4305  '  0100-0107 4305     3DIGIT
CTLENTS  EQU   (*-CTLRTAB)/12                  NUMBER OF TABLE ENTRIES
CTLNFND  DC    CL8'????????'
CTLUNIT  DC    CL8' '
*
MSG1     DC    C'>>> PARM IS MISSING/INVALID - ENTER "?" FOR FORMAT...'
MSG1L    EQU   *-MSG1
*
MSG2     DC    C'>>> INVALID CUU SPECIFIED...(CCUU).'
```

```
MSG2TEXT EQU   MSG2+29,4
MSG2L    EQU   *-MSG2
MSG3     EQU   *
MSG3LIN1 DC    CL1'|'
MSG3CUU1 DC    CL5' '
MSG3TYP1 DC    CL5' '
MSG3STA1 DC    CL8' '
MSG3VOL1 DC    CL7' '
MSG3LIN2 DC    CL1'|'
MSG3CUU2 DC    CL5' '
MSG3TYP2 DC    CL5' '
MSG3STA2 DC    CL8' '
MSG3VOL2 DC    CL7' '
MSG3LIN3 DC    CL1'|'
MSG3CUU3 DC    CL5' '
MSG3TYP3 DC    CL5' '
MSG3STA3 DC    CL8' '
MSG3VOL3 DC    CL7' '
MSG3LIN4 DC    CL1'|'
MSG3L    EQU   *-MSG3
MSG4     DC    C'>>> NO MATCHING UCB FOUND...(CCUU).'
MSG4TEXT EQU   MSG4+29,4
MSG4L    EQU   *-MSG4
*
MSG5     DC    C'>>> NO MATCHING VOLID FOUND...(VOLSER).'
MSG5TEXT EQU   MSG5+31,6
MSG5L    EQU   *-MSG5
*
MSG6     DC    C'—CUU-TYPE-STATUS—VOLSER——SIZE—ATTRIBUTE—UCB'
         DC    C'——SHAREABLE——DEVICE——'
MSG6L    EQU   *-MSG6
*
MSG6A    DC    C'—CUU-TYPE-STATUS—VOLSER—CUU-TYPE-STATUS—VOLSER'
         DC    C'—CUU-TYPE-STATUS—VOLSER—'
*
MSG6B    DC    C'—CUU-TYPE-STATUS—VOLSER——————————————'
         DC    C'————————————————'
*
MSG7     DC    C'>>> TOO MANY PARMS PASSED...'
MSG7L    EQU   *-MSG7
*
MSG8     DC    C'>>> INVALID CUU RANGE SPECIFIED...(AAAA-BBBB).'
MSG8TEXT EQU   MSG8+35,9
MSG8L    EQU   *-MSG8
*
MSG9     DC    C'>>> NO MATCHING CUU''S FOUND...(AAA-BBB  ).'
MSG9TEXT EQU   MSG9+31,9
MSG9L    EQU   *-MSG9
*
MSGA     DC    C'>>> NO MATCHING VOLID''S FOUND...(VOLSER).'
MSGATEXT EQU   MSGA+33,6
MSGAL    EQU   *-MSGA
```

```
*
MSGB     DC    C'>>> ERROR IN CALL TO "UCBSCAN"...RC=X''..'', RS=X''..'X
               '.'
MSGBTXT1 EQU   MSGB+38,2
MSGBTXT2 EQU   MSGB+48,2
MSGBL    EQU   *-MSGB
*
HELPMSG1 DC    CL50'                  Q U E R Y C M D'
HELPMSG2 DC    CL50'                  ==============='
HELPMSG3 DC    C' '
HELPMSG4 DC    CL50'    DISPLAY TAPE OR DASD INFO ON TSO. PARMS ARE:'
HELPMSG5 DC    CL50'         ??           DISPLAY ALL UNITS'
HELPMSG6 DC    CL50'         ?? CUU       DISPLAY ONLY UNIT "CUU"'
HELPMSG7 DC    CL50'         ?? CCUU      DISPLAY ONLY UNIT "CCUU"'
HELPMSG8 DC    CL50'         ?? VOLSER    DISPLAY ONLY UNIT "VOLSER"'
HELPMSG9 DC    CL50'         ?? ONLINE    DISPLAY ONLY ONLINE UNITS'
HELPMSGA DC    CL50'         ?? XXX*      DISPLAY ONLY ONLINE UNITS'
HELPMSGB DC    CL50'                      WITH VOLIDS STARTING "XXX"'
HELPMSGC DC    CL50'         ?? AAAA-BBBB OR'
HELPMSGD DC    CL50'         ?? AAA-BBB   DISPLAY ONLY UNITS WITHIN'
HELPMSGE DC    CL50'                      CUU RANGE OF "AAA" TO "BBB"'
HELPMSGF DC    CL50'         WHERE "??" IS:- DA FOR DASD'
HELPMSGG DC    CL50'                         TA FOR TAPE'
*-------------------------------------*
* REGISTERS EQUATES, ETC...                                          *
*-------------------------------------*
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         PRINT ON,GEN
UCBDEF   DSECT
         IEFUCBOB
         PRINT NOGEN
         CVT   DSECT=YES
         END                    , END OF PROGRAM
```

PDSE utilities

INTRODUCTION

PDSEs (extended partitioned datasets) offer a number of advantages over normal partitioned datasets.These include:

- They are self-reorganizing (IEBCOPY does not need to be used to reclaim space).
- Directory blocks do not need to be preassigned.
- They offer improved performance.
- Extended aliases can be used (a PDSE alias can be 1024-bytes long and contain mixed-case characters).

Unfortunately, only limited program support is provided for this last item, restricted mainly to the BINDER (Linkage Editor replacement) that can be used to assign long and mixed-case alias names.

The two utility programs described in this article provide the necessary support at the application level to use extended aliases.

EXTENDED ALIASES

Increasingly programming languages are leaving the historical restriction that limited program names to just eight characters. Before the advent of PDSEs, such 'long' names had to be realized using such techniques as name mangling as used by C, and C++.

Interpreted languages like REXX have long been able to use extended names for internal routines, namely routines contained in the same physical source. However, because of the lack of standard support for such long names, it has not been practicable to store routines under their extended names in libraries (which counters the efforts of making code available for reuse). The SETPDSE program described in this article allows the use of extended names.

A second problem arises in listing the contents of directories with extended names. ISPF utilities ignore extended names.

For example, using PDF browse to list a PDSE source with extended aliases produced the following directory list:

```
sqrt.rex
CLIUTIL
MYREXX
SquareRt
SQRT
```

Whereas the LISTPDSE program described in this article produced the following (correct) list:

```
NAME     MYREXX
ALIAS    GetSquareRoot                  SQRT
NAME     SQRT
ALIAS    sqrt.rex                       SQRT
ALIAS    SQUAREROOT                     SQRT
ALIAS    SquareRt                       SQRT
ALIAS    cliutil.h                      CLIUTIL
NAME     CLIUTIL
```

As can be seen, the PDF utility ignores the three extended aliases: GetSquareRoot, SQUAREROOT, and cliutil.h. Although IEHLIST does handle extended aliases correctly, it is awkward to use and the listing impractical for large directories (no direct assignment of extended aliases to the base module or alias). The LISTPDSE program described in this article solves this problem.

SETPDSEA

SETPDSEA sets the PDSE alias and requires as input the extended alias and the name of the entry to which the extended alias is to be added.

Sample call

```
//S1 EXEC PGM=SETPDSE,PARM='SQRT     GetSquareRoot'
//*                        pppppppp a.......a
//* pppppppp = base program name (or alias)
//* a.......a = extended alias
//SYSUT2 DD DSN=TUSERØ1.PDSE.SOURCE,DISP=OLD
```

EXEC parameter

pppppppp a.......a

pppppppp Name of the base entry (8 characters, case-sensitive, left-justified)

a.....a Name of the extended alias (case-sensitive, left-justified)

File

SYSUT2 DD statement for the PDSE.

Messages

Any error messages are written to the programmer log (WTO ROUTCDE=11) and the return code set appropriately. The error messages are self-explanatory.

SOURCE

```
         TITLE 'Set PDSE Alias'
         PRINT NOGEN
**
* SETPDSE: Set PDSE alias
**
* Call: EXEC PGM=SETPDSE,PARM='pppppppp aaa...'
*  pppppppp = Program (entry) name (8 characters fixed, uppercase)
*  aaa... = Alias name (variable length, mixed case)
* Note: The initial part of the EXEC parameter has a fixed format
*       (8-character left-justified name entry plus separating blank).
**
* DD: SYSUT2. The PDSE for which the alias is to be set.
**
SETPDSE  CSECT
SETPDSE  AMODE 31
SETPDSE  RMODE 24
         BAKR  R14,Ø              save registers and return address
         BASR  R12,Ø              set base register
         USING *,R12
         SPACE
         MVC   MSG,MSG-1          clear message line
         SPACE
         L     R1,Ø(R1)           pointer to parameter
         LH    R3,Ø(R1)           PARM-length
         LA    R4,2(R1)           PARM data address
         USING PARM,R4
         OPEN  (SYSUT2,(OUTPUT))
         LTR   R15,R15
         BZ    OPENOK
         MVC   MSGTYPE,=CL16'OPEN ERROR'
```

```
         B     PUTMSG
         SPACE
OPENOK   LA    R8,OALIAS
OA       USING DESN,R8            old alias
         LA    R9,NALIAS
NA       USING DESN,R9            new alias
         SPACE
         MVC   OA.DESN_LEN,=H'8'
         MVC   OA.DESN_VAL(8),PARMOLD
         SH    R3,=H'9'           remaining length (=alias length)
         STH   R3,NA.DESN_LEN
         LA    R2,PARMNEW
         LA    RØ,NA.DESN_VAL
         LA    R1,L'NALIAS-L'DESN_LEN
         MVCL  RØ,R2
         USING DESL,INLIST
         ST    R8,DESL_OLD_NAME_PTR
         ST    R9,DESL_NEW_NAME_PTR
* position at member
         MVC   BLDLNAME,PARMOLD
         BLDL  SYSUT2,BLDLLIST
         LTR   R15,R15
         ST    R15,RC
         BZ    BLDLOK
         MVC   MSGTYPE,=CL16'FIND ERROR'
         MVC   MSGNAME,BLDLNAME
         B     PUTMSG
         SPACE
BLDLOK   DS    ØH
* add temporary alias
         MVC   BLDLNAME,=CL8' TEMP'
         MVI   BLDLFLAG,B'1ØØØØØØØ'
         STOW  SYSUT2,BLDLNAME
         LTR   R15,R15
         ST    R15,RC
         BZ    STOWOK
         MVC   MSGTYPE,=CL16'STOW ERROR'
         MVC   MSGNAME,BLDLNAME
         B     PUTMSG
STOWOK   DS    ØH
         MVC   BLDLLEN,=H'8'
         LA    RØ,BLDLLEN
         ST    RØ,DESL_OLD_NAME_PTR
         DESERV FUNC=RENAME,DCB=SYSUT2,RETCODE=RC,RSNCODE=RSC,         X
               NAME_LIST=(INLIST,1)
         ICM   R15,15,RC
         BZ    RENAMEOK
         MVC   MSGTYPE,=CL16'RENAME ERROR'
         B     PUTMSG
         SPACE
RENAMEOK LA    R15,Ø                normal end
```

```
         MVC   MSGTYPE,=CL16'RENAME OK'
         B     PUTMSG
         SPACE
EXIT     ST    R15,RC
         CLOSE (SYSUT2)
         L     R15,RC
         PR    ,                    program return
         SPACE 2
PUTMSG   DS    ØH                   output message to log
         WTO   TEXT=MSGLINE,ROUTCDE=(11)
         L     R15,RC               load return code
         B     EXIT
         SPACE 2
MSGLINE  DC    AL2(MSGEND-MSGFILL)
MSGFILL  DC    C' '  fill character
MSG      DS    CL(MSGEND-MSGSTART)
         ORG   MSG
MSGSTART EQU   *
MSGTYPE  DS    CL16,C
MSGNAME  DS    CL8
MSGEND   EQU   *
         SPACE
BLDLLIST DC    AL2(1)
BLDLLEN  DC    AL2(12)
BLDLNAME DS    CL8
BLDLTTR  DS    XL3
BLDLFLAG DS    X
         SPACE
INLISTCT DC    F'1'
INLIST   DS    CL16
OALIAS   DS    CL32
NALIAS   DS    CL32
PTR      DC    A(Ø)
RC       DS    F
RSC      DS    F
         SPACE
SYSUT2   DCB   DDNAME=SYSUT2,DSORG=PO,MACRF=W
         SPACE
PARM     DSECT
PARMOLD  DS   CL8
         DS   C
PARMNEW  DS   CL8Ø
* symbolic register equates
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
```

```
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END
```

LISTPDSE

LISTPDSE lists the directory contents of a PDSE. The output is written directly to the programmer log (WTO ROUTCDE=11). Each line contains two (or three) items:

- etype – name [bname]
- etype – entry type (NAME, ALIAS)
- name – entry name
- pname – base entry name (for alias) - this entry appears only for an alias.

Sample call

```
//S2 EXEC PGM=LISTPDSE
//SYSUT1 DD DSN=TUSER01.PDSE.SOURCE,DISP=SHR
```

Sample output

```
NAME     MYREXX
ALIAS    GetSquareRoot                  SQRT
NAME     SQRT
ALIAS    sqrt.rex                       SQRT
ALIAS    SQUAREROOT                     SQRT
ALIAS    cliutil.h                      CLIUTIL
NAME     CLIUTIL
```

File

SYSUT1 – DD statement for the PDSE.

Messages

Any error messages are written to the programmer log and the return

code set appropriately. The error messages are self-explanatory.

SOURCE

```
         TITLE 'LISTPDSE: List PDSE'
**
* LISTPDSE: List PDSE(PDS) directory entries
* The entry names and alias names are written to the programmer log
* (WTO with ROUTCDE=11).
* Each alias name also lists the primary name.
**
* Invocation:
*  // EXEC PGM=LISTPDSE
*  //SYSUT1 DD DSN=pdsedsname,DISP=SHR
**
* Output:
*  etype    name    <pname>
**
*  etype: Entry type (NAME, ALIAS)
*  name:  Entry name
*  pname: Primary entry name (for alias)
**
* DD: SYSUT1. The directory to be listed.
**
         PRINT NOGEN
LISTPDSE CSECT
LISTPDSE AMODE 31
LISTPDSE RMODE 24
         BAKR  R14,Ø
         BASR  R12,Ø
         USING *,R12
         OPEN  (SYSUT1,(INPUT))
         LTR   R15,R15
         LA    R15,2Ø              OPEN error
         BNZ   EXIT
         DESERV FUNC=GET_ALL,AREAPTR=PTR,DCB=SYSUT1
         LTR   R15,R15
         BNZ   EXIT                error
         SPACE
         L     R8,PTR              pointer to DESB
         USING DESB,R8
         L     R9,DESB_COUNT       no. of DESB entries
         LA    R7,DESB_DATA        first SMDE
         USING SMDE,R7
SMDELOOP MVC   MSG,MSG-1           clear message line
         LH    R6,SMDE_NAME_OFF    NAME offset
         LA    R5,Ø(R6,R7)         address of NAME entry
         USING SMDE_NAME,R5
         LA    RØ,SMDE_NAME_VAL    address of NAME data
         LH    R1,SMDE_NAME_LEN    length of NAME data
```

```
         O     R1,=X'40000000'
         LA    R14,NAME
         LA    R15,L'NAME
         MVCL  R14,RØ              move to output line
         MVC   MSGNAME,=CL8'NAME'  set line type
         TM    SMDE_FLAG,SMDE_FLAG_ALIAS
         BZ    NOALIAS
         MVC   MSGNAME,=CL8'ALIAS' set line type
         LH    R6,SMDE_PNAME_OFF   PNAME (Primary Name) offset
         LA    R5,Ø(R6,R7)
         LA    RØ,SMDE_NAME_VAL    address of PNAME data
         LH    R1,SMDE_NAME_LEN    length of PNAME data
         O     R1,=X'40000000'
         LA    R14,PNAME
         LA    R15,L'PNAME
         MVCL  R14,RØ              move to output line
* output line
NOALIAS  WTO   TEXT=MSGLINE,ROUTCDE=(11)
* get next SMDE entry
         A     R7,SMDE_LEN
         BCT   R9,SMDELOOP
* last entry
         LA    R15,Ø               set normal end
EXIT     STH   R15,RC
         CLOSE (SYSUT1)
         LH    R15,RC
         PR    ,                   program return
         SPACE
SYSUT1   DCB   DDNAME=SYSUT1,DSORG=PO,MACRF=R
         SPACE
RC       DS    H
PTR      DS    A
* message line
MSGLINE  DC    AL2(MSGEND-MSGFILL)
MSGFILL  DC    C' '  fill character
MSG      DS    CL(MSGEND-MSGSTART)
         ORG   MSG
MSGSTART EQU   *
MSGNAME  DS    CL8,C
NAME     DS    CL32,C
PNAME    DS    CL32
MSGEND   EQU   *
         SPACE
         IGWSMDE
* symbolic register equates
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
```

```
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END
```

Example

The following example shows a REXX program that uses external routines with extended names (SquareRoot, written in two ways).

Note: if the previously listed PDSE is used, the call to SquareRt would fail. REXX folds names to uppercase (even when they are written within quotes). This means the call to SquareRt would be converted to call SQUARERT, which, however, is not in the directory.

```
/* REXX */
arg = 5
val = 'SquareRoot'(arg)
SAY arg val val*val

arg = 6
val = SquareRoot(arg)
SAY arg val val*val

arg = 3
val = 'SquareRt'(arg)
SAY arg val val*val
```


MVS Update code available from the Web

As a free service to subscribers and to remove the need to rekey the scripts, code in individual articles can be accessed on our Web site. Subscribers need the user-id printed on the envelope containing their *Update* issue. Once they have registered, any code requested will be e-mailed to them.

Landmark Systems is to expand its TMON product line to include new performance management products for TCP/IP, IMS, and Unix System Services, addressing network management and application services as well as expanded coverage for OS/390 performance management.

The TCP/IP tool will enable organizations to monitor and manage the availability of network devices and overall network performance. It will provide insight into all critical resources impacting TCP/IP network performance by extending beyond the OS/390 to monitor and manage non-OS/390 TCP/IP stacks and Cisco routers, while also providing visibility into S/390 subsystems, applications, and resources that impact the OS/390 TCP/IP stack.

The IMS tool provides broad problem solving and resolution capabilities, as well as task analysis, exception analysis, and resource management.

The system for managing the performance of Unix System Services will provide configuration management, performance monitoring, and diagnosis and tuning for USS.

Products for TCP/IP and IMS will ship in December, with the USS tool shipping in the first half of 2001. Prices weren't announced.

For further information, contact:
Landmark Corporation, 8000 Towers Crescent Drive, Vienna, VA 22182, USA.

http://www.landmark.com.

* * *

BMC has begun shipping its new Instant Snapshot for the recovery of OS/390 databases.

Instant Snapshot for OS/390, available for DB2 and IMS databases utilizing EMC Symmetrix storage systems with EMC TimeFinder software, is said to provide integration with EMC storage systems for back-ups. Customers can use the Symmetrix systems to obtain hardware-based copies of datasets made at a specific point-in-time. Instant Snapshot uses the hardware copies to provide recovery.

The bi-directional implementation of BMC's snapshot technology utilizes the dataset snapshot capabilities of the storage devices. During the Instant Copy process, the datasets are copied to the same or another device via the hardware snapshot process.

For further information contact:
BMC Software, 2101 City West Boulevard, Houston, TX 77042-2827, USA.

http://www.bmc.com

* * *

Xephon will be providing a full analysis of z/OS – its functionality, pricing, and future directions – at our forthcoming *Mainframe Futures 2000* conference on 21-22 November in London. Book your place now to avoid disappointment.

http://www.xephon.com/zevent.html

* * *