144 MVS update

September 1998

In this issue

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

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

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

Editor
Jaime Kaminski

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

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

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

Printed in England.

Y2K and missing source

THE PROBLEM

For anyone working on upgrading existing applications to be Year 2000 capable, the worst problem that you can encounter is finding that some or all of the source code is missing. Given the number of applications being worked on around the world, this situation will undoubtedly affect many sites. Overcoming the problem generally involves either re-writing the code, replacing the application, or, in extreme cases, killing the application.

A SOLUTION

When the problem hit one of our applications recently another approach was suggested, which may be of interest to others. To describe this approach, let me start by defining the problem of missing source in another way. For the purposes of Y2K, missing source matters only if the application cannot function in Y2K. In other words, if testing identifies that the application is not Y2K capable, then some action is required. This may seem obvious, but there is a key point to make about that definition, and that is the fact that it takes testing to identify that the application will not work in the year 2000. This testing technology has the potential for solving the problem.

The typical technique used to test a Y2K environment uses a data ageing product to move data forward into the next century. This is followed by running the application under a date manipulation tool, and then comparing the results with those created by ageing the results of a current run. Assume now that the date is the year 2000. To run the application, all that is required is to use the ageing tool to put data back in time, run the application under the date tool, and re-age the data. Or even more simply, backdate the data now to a chosen date. The application always runs at an offset from the current date, and the only ageing process required is after the application has run.

There is also another potential use for this concept. If the testing technology works for an application, then that application does not necessarily have to be changed by the year 2000 (ie using this technique could help to soften the Y2K immovable deadline). I'll leave it open to discussion as to how widely this idea (data youthing as against data ageing) might be used, but it would be nice to think that all the products being developed for Y2K might have more benefits than just for one deadline.

STCK display under TSO

INTRODUCTION

Timestamps are often encountered in places like system traces, dumps, and logs. The standard form of the STCK is hard to interpret and convert to print format. This has, however, been made a lot easier with the availability of the STCKCONV macro available in MVS Version 5 and upwards.

This program is an application of that macro. It is called by means of a REXX routine. Many TSO users work from workstations with cut-and-paste buffers these days, and it was written so that you can simply cut-and-paste the address into the command. So, when looking at a dump or trace, you can simply enter the command, get the timestamp and continue looking at the trace. It will accept either of the following two formats:

```
TSO  STCK  XXXXXXXX  XXXXXXXX
```

or

```
TSO STCK  XXXXXXXXXXXXXXXX
```

with the parameters containing the print format of the current STORE CLOCK output. For example:

```
TSO  STCK 12345678 12345678
```


will deliver

```
Time=15:59:32.94266100, Date=1910.055.
```

To see when the system clock started and will end, enter

```
TSO  STCK 00000000 00000000
```

which gives

```
Time=00:00:00.00000000, Date=1900.001
```

and

```
TSO  STCK FFFFFFFF FFFFFFFF
```

which gives

```
Time=23:53:47.37049500, Date=2042.260.
```

(The designers of System/390 will then have to come up with some solution, something along the lines of the Y2K conversion currently going on worldwide.) Here is the REXX routine used to invoke the utility:

```
 /* REXX */
parse arg val1
if substr(val1, 9, 1) ¬= ' '  then
            val1 = substr(val1, 1, 8) || '  '|| substr(val1, 9, 8)
"CALL  'your.loadlib(STCKMOD)'     '"val1"' "
```

STCK

```
STCKPGM  CSECT
STCKPGM  AMODE 31
STCKPGM  RMODE 24
         BAKR  R14,0               .Save Caller's Status
         BALR  R12,0
         USING *,12
*********************************************************************
*        Main driver routine
*********************************************************************
Load     L     R4,0(R1)            .Ptr to passed parameter
         CLC   0(2,R4),=H'17'      .Input parms must be 17 bytes
         BE    Storage
         TPUT  ParmFmt,L'ParmFmt
         LA    R15,8
         PR                        .Quick way out
         LA    R3,GetMSize         .Our requirement
Storage  STORAGE OBTAIN,LENGTH=(3),LOC=BELOW
```

```
         LR    R2,R1               .Point to getmained area
         LA    R3,GetMSize
         XR    R9,R9
         MVCL  R2,R8               .Propagate binary zeros
         USING GetMArea,R1
         ST    R13,SaveArea+4      .Backchain
         DROP  R1
         LR    R13,R1
         USING GetMArea,R13        .Addressability to getmained area
         BAS   R14,ChckParm        .Go do further validations on parm
         LTR   R15,R15             .Parm OK?
         BNZ   Return              .No, get out
         BAS   R14,CnvParm         .Go convert passed parm to HEX
         BAS   R14,STCKCONV        .Go convert STCK to decimal format
         LTR   R15,R15             .Parm OK?
         BNZ   Return              .No, get out
         BAS   R14,STCKPRT         .Go convert STCK to print format
Return   EQU   *                   .Pick up return code
         L     R4,RetCode          .Pick up return code
         LA    R3,GetMSize         .Size of area to free
         LR    R2,R13              .Address of area to free
         STORAGE RELEASE,LENGTH=(R3),ADDR=(R2)
         LR    R15,R4              .Copy return code
ToCaller PR    ,                   .=>Caller
         DS    ØD                  .Align
         EJECT
*********************************************************************
*        This routine verifies the input parm's content
*********************************************************************
Chckparm BAKR  R14,Ø
         MVC   Address(8),2(R4)    .First 8 bytes of STCK value
         MVC   Address+8(8),11(R4) .Second 8 bytes of STCK value
         TRT   Address,TrtTab1     .Must be Ø-9 and A-F
         BNZ   ParmErr             .Contains invalid characters
         XR    R15,R15             .Clear return code
         B     ChckParX            .Get out
ParmErr  TPUT  InvlChar,L'InvlChar .Tell user garbage in input list
         LA    R15,8               .Set return code to 8
         ST    R15,Retcode         .Plug return code
ChckParX PR                       .Back to caller
*********************************************************************
*        This routine converts print format STCK parm to hexadecimal
*********************************************************************
CnvParm  BAKR  R14,Ø
         TR    Address,TrtTab2     .Convert the passed address
         LA    R1,8                .Number of loop iteration
         LA    R2,Address          .Where we are converting from
         LA    R3,HEXAddr          .Where we are converting to
ConvLoop EQU   *
         IC    R4,Ø(R2)            .Pick up the next character
         SLL   R4,4                .Move half byte to the left
```

```
         STC   R4,Ø(R3)            .Left half of the byte
         OC    Ø(1,R3),1(R2)       .Right half of the byte
         LA    R2,2(R2)            .Bump up from pointer
         LA    R3,1(R3)            .Bump up to pointer
         BCT   R1,ConvLoop         .Do for all 16 bytes
CnvParmX PR                        .Return to our caller
*********************************************************************
*        This routine converts HEX STCK value to understandable format
*********************************************************************
STCKCONV BAKR  R14,Ø
         STCKCONV STCKVAL=HEXAddr,CONVVAL=ADDRESS,TIMETYPE=DEC
         LTR   R15,R15             .Successful?
         BNZ   ConvErr             .No
         XR    R15,R15             .Set our return code to zero
         B     STCKCONX            .Get out
ConvErr  TPUT  NoCnvrt,L'NoCnvrt
         LA    R15,8               .Set the return code to 8
         ST    R15,RetCode         .Plug return code
STCKCONX PR
*********************************************************************
*        This routine converts decimal STCK value to print format
*********************************************************************
STCKPrt  BAKR  R14,Ø
         MVC   PrtAddr1,Address    .Print left half of each byte
         NC    PrtAddr1,=16X'FØ'   .Turn right halves off in each byte
         TR    PrtAddr1,LftHalve   .Make left halves printable
         MVC   PrtAddr2,Address    .Print right half of each byte
         NC    PrtAddr2,=16X'ØF'   .Turn left halves off in each byte
         TR    PrtAddr2,RgtHalve   .Make right halves printable
         LA    R1,16               .Number of bytes
         LA    R2,PrtAddr1         .Point to left half
         LA    R3,PrtAddr2         .Point to right half
         LA    R4,Result           .Where we want the result to be
MoveLoop MVC   Ø(1,R4),Ø(R2)       .Move left half
         MVC   1(1,R4),Ø(R3)       .Move the right half
         LA    R2,1(R2)            .Bump up left half pointer
         LA    R3,1(R3)            .Bump up right half pointer
         LA    R4,2(R4)            .Bump up result pointer
         BCT   R1,MoveLoop         .Do for each byte
         MVC   FinRslt+5(2),Result
         MVC   FinRslt+8(2),Result+2
         MVC   FinRslt+11(2),Result+4
         MVC   FinRslt+14(8),Result+6
         MVC   FinRslt+29(4),Result+17
         MVC   FinRslt+34(3),Result+21
         TPUT  FinRslt,L'FinRslt
STCKPrtX PR                        .Return to caller
*********************************************************************
*        Constants follow
*********************************************************************
TrtTab1  DC    193X'Ø1',6X'ØØ',41X'Ø1',1ØX'ØØ',6X'Ø1'
```

```
TrtTab2  DC    193X'FF',X'ØA',X'ØB',X'ØC',X'ØD',X'ØE',X'ØF',41X'FF'
         DC    X'ØØ',X'Ø1',X'Ø2',X'Ø3',X'Ø4',X'Ø5',X'Ø6',X'Ø7',X'Ø8'
         DC    X'Ø9'
TrtTab3  DC    X'ØØ',X'Ø1',X'Ø2',X'Ø3',X'Ø4',X'Ø5',X'Ø6',X'Ø7',X'Ø8'
         DC    X'Ø9'
LftHalve DS    ØCL24Ø
         DC    X'FØ',15X'ØØ',X'F1',15X'ØØ',X'F2',15X'ØØ',X'F3'
         DC    15X'ØØ',X'F4',15X'ØØ',X'F5',15X'ØØ',X'F6',15X'ØØ',X'F7'
         DC    15X'ØØ',X'F8',15X'ØØ',X'F9',15X'ØØ',X'C1',15X'ØØ',X'C2'
         DC    15X'ØØ',X'C3',15X'ØØ',X'C4',15X'ØØ',X'C5',15X'ØØ',X'C6'
RgtHalve DC    X'FØF1F2F3F4F5F6F7F8F9C1C2C3C4C5C6'
FinRslt  DC    C'Time=xx:xx:xx.99999999, Date=yyyy.ddd'
ParmFmt  DC C'Input paramaters must be of the format XXXXXXXX XXXXXXXX'
InvlChar DC    C'Passed address must only contain Ø-9 and A-F'
NoCnvrt  DC    C'Non-zero return code received from STCKCONV macro'
         LTORG
*********************************************************************
*        DSECTs follow
*********************************************************************
GetMArea DSECT
SaveArea DS    18F                .General savearea
Retcode  DS    F                  .Return code
Address  DS    CL16               .Passed value
PrtAddr1 DS    CL16               .Left halves
PrtAddr2 DS    CL16               .Right halves
Result   DS    CL5Ø               .Final result to be displayed
HEXAddr  DS    CL8                .Passed value in HEX
GetMSize EQU   *-GetMArea
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
```

ISPF dataset tool

INTRODUCTION

Since ISPF Version 4.2, there has been an 'undocumented' tool called ISRDDN supplied with every ISPF. ISRDDN has been developed as an internal tool by IBM. Although it is not mentioned in any ISPF manual, it has been enthusiastically described in their *ISPF News* magazine – so it is supplied but not officially supported. ISRDDN is a tool for listing your current dataset allocations, and then offering many functions for those datasets. Many sites have developed their own tools to carry out similar functions, but this one is supplied by IBM so it is worth investigating.

ISRDDN is a program, so you can invoke it from almost any ISPF panel by the command TSO ISRDDN. It lists all the allocated datasets, with the ddnames sorted alphabetically. It shows the ddname and dataset name plus VOLSER, DISP, BLKSIZE, LRECL, RECFM, and DSORG. It has line commands for the following:

- BROWSE – a dataset or the first four datasets in a file concatenation.
- EDIT – a dataset or the first four datasets in a file concatenation.
- VIEW – a dataset or the first four datasets in a file concatenation.
- FREE – the DD allocation (if it is not open).
- COMPRESS – a PDS (even if it is in use by other users!).
- INFORMATION – about a dataset (the same as ISPF 3.2).
- ENQUEUE – display for a dataset.

It has primary commands for the following:

- FIND – a string in the display.
- LOCATE – the first ddname containing a given string.
- ONLY – display the ddnames containing a given string.
- EXCLUDE – from display all ddnames containing a given string.

- MEMBER – scan for a particular member in the displayed libraries.
- CLIST – created with allocate statements for all current allocations.
- COUNT – the number of members in all the displayed libraries.
- LPA and LINKLIST – libraries to be listed.

IBM is continuing to develop ISRDDN. In ISPF 4.5 (available with OS/390 Version 2.5) there are additional functions. These include:

- Viewing your own storage.
- Displaying certain ISPF control blocks and PTF levels.
- Chaining through control blocks.
- Listing any enqueues on the system.
- Showing enqueue contention.
- Showing information about load modules.

All these functions are detailed in the HELP panels for ISRDDN.

CONCLUSION

ISRDDN is a good tool for diagnosing TSO/ISPF problems, and it will continue to be developed, at no (extra) cost to users.

Ron Brown
Systems Programmer (Germany)

Building control card images quickly

THE PROBLEM

Frequently within CLISTs and REXX EXECs, utilities are invoked that require input via control cards. While the control cards can be built within CLISTs and EXECs, it is not a user friendly process. For that reason, I came up with the CNTLCARD program, which makes it much easier to build the control card input required by such utilities.

A SOLUTION

The CNTLCARD program assumes (and requires) that a pre-allocated virtual I/O (VIO) dataset be used to hold the control card that is to be generated. This was done as a security feature, since sensitive information, like passwords, could be put into the control card and only the job or user allocating the VIO dataset could access it. It is also the reason that only one control card can be written to the VIO dataset using the CNTLCARD program (you cannot specify a disposition of MOD for a VIO dataset). If you do not require the security, or if you want to be able to create multiple control cards, you can modify the source code at label DDOK to skip the VIO checking. If you do not perform this modification, CNTLCARD will set a return code of 4 if the SYSIN DD does not refer to a VIO dataset.

CNTLCARD will return two other return codes. A code of 8 will be set if the SYSIN DD is not allocated, and a return code of 12 will be set if no parameter is passed to it. If you attempt to pass a control card whose length is greater than 80 bytes for CNTLCARD to write, it will truncate the length to 80 bytes without issuing any notification. In order to handle special character situations, such as leading blanks, the string of data to be used as a control card can be enclosed in single quotes, which will be stripped off if found by the CNTLCARD program. Otherwise the string can be coded without the quotes even if it has embedded blanks or special characters.

As an example of its use, to generate a SORT control card, you would code:

```
SYSLOAD  ' SORT FIELDS=(3,9,CH,A),FILSZ=E100'
```

Below is an example CLIST that uses an unmodified CNTLCARD program to write a single control card to a VIO dataset:

```
FREE DD(SYSIN) DELETE
ALLOC DD(SYSIN) UNIT(VIO)   SP(1) NEW REUSE                +
      RECFM(F) LRECL(80) BLKSIZE(80) DSORG(PS)
SYSLOAD ' SORT FIELDS=(3,9,CH,A),FILSZ=E100'
```

Below is an example CLIST that uses a modified version of the CNTLCARD program (as described earlier) to write multiple control cards to a non-VIO dataset:

```
FREE DD(SYSIN) DELETE
ALLOC DD(SYSIN) UNIT(SYSDA) SP(1) MOD REUSE DA(TEMP)       +
      RECFM(F) LRECL(80) BLKSIZE(80) DSORG(PS)
```

```
 SYSLOAD ' SORT FIELDS=(3,9,CH,A),FIlSZ=E1ØØ'
 SYSLOAD ' RECORD TYPE=F'
```

CNTLCARD ASSEMBLE

```
         TITLE 'CNTLCARD-COMMAND TO LOAD SYSIN FROM COMMAND PARM.'
CNTLCARD CSECT
         YREGS
         STM   R14,R12,12(R13)
         LR    R12,R15
         USING CNTLCARD,R12
         LA    R2,SAVE
         ST    R2,8(,R13)
         ST    R13,4(,R2)
         LR    R13,R2
         LR    R2,R1               SAVE CPPL POINTER.
         RDJFCB SYSIN              LOAD JFCB.
         LTR   R15,R15             DDNAME EXIST?
         BZ    DDOK
         LA    R15,8               LOAD ERROR CODE OF 8.
         B     EXIT                EXIT.
DDOK     TM    JFCBAREA+JFCFLGS1-JFCB,JFCVRDS VIO DATASET?
         BO    VIOOK
         LA    R15,4               LOAD ERROR CODE OF 4.
         B     EXIT                EXIT.
VIOOK    OPEN  (SYSIN,OUTPUT),TYPE=J   OPEN THE VIO DATASET.
         L     R3,Ø(,R2)           POINT TO PARM.
         LH    R5,Ø(,R3)           LOAD LENGTH.
         LH    R4,2(,R3)           LOAD OFFSET.
         LA    R4,4(,R4)           CREATE ...
         SR    R5,R4               ... LENGTH.
         LTR   R5,R5               ANYTHING THERE?
         BP    PARMOK              B IF YES.
         LA    R15,12              LOAD ERROR CODE.
         B     EXIT                EXIT.
PARMOK   CH    R5,=H'8Ø'           GREATER THAN 8Ø?
         BNH   PARMOKØ             B IF NOT.
         LA    R5,8Ø               TRUNCATE.
PARMOKØ  LA    R3,Ø(R3,R4)         POINT TO PARM.
         CLI   Ø(R3),C''''         FIRST CHAR APOST?
         BNE   MPARM               B IF NOT.
         LA    R3,1(,R3)           BUMP POINTER
         SH    R5,=H'2'            ASSUME TRAILING APOST.
         BP    MPARM               B IF NOT MERELY ONE APOST.
         LA    R15,12              SET RC OF 12.
         B     EXIT                EXIT.
MPARM    BCTR  R5,*-*              GEN SS LEN.
         EX    R5,MOVEPARM         MOVE PARM.
         PUT   SYSIN,OUT           OUTPUT PARM.
         CLOSE (SYSIN)             CLOSE DATASET.
         SR    R15,R15             SET RC OF Ø.
```

```
EXIT     L     R13,SAVE+4
         L     R14,12(,R13)
         LM    R0,R12,20(R13)
         BR    R14
MOVEPARM MVC   OUT(*-*),0(R3)      EXECUTED MOVE.
SAVE     DS    18F
EXLST    DC    X'87',AL3(JFCBAREA)
JFCBAREA DS    XL176
OUT      DC    CL80' '
         LTORG
         PRINT NOGEN
SYSIN    DCB   DDNAME=SYSIN,DSORG=PS,LRECL=80,BLKSIZE=(80),            *
               RECFM=F,BUFNO=1,NCP=1,MACRF=(PM),EXLST=EXLST
JFCB     DSECT
         IEFJFCBN LIST=NO
         END
```


Displaying the active EDT

THE PROBLEM

Usually, there is little need to dynamically reference the Eligible Device Table (EDT) and even less need to alter Unit Control Blocks (UCBs) for specific devices. However, when the need arises, it is extremely useful to have this data available in an easily decipherable format.

THE SOLUTION

It was for reasons described above that the programs covered by this article were created. EDTISPF and BLDEDT are two programs that work together to provide an ISPF view into an active EDT environment on an MVS system. EDTISPF provides the ISPF interfaces required to manage the panel displays that occur while reading the EDT information. BLDEDT is invoked from EDTISPF to build the re-usable control block structure of an MVS systems active EDT. Some of the capabilities provided by this tool include the ability to:

- View all esoteric names associated with a specific device number.
- View the UCB of a specific device.

- Determine all devices associated with a specific esoteric device name.
- View all the esoteric names defined in the EDT.

```
———————————  ELIGIBLE DEVICE TABLE UTILITY  ———————————-
COMMAND ===>

UNIT NAME     ===>          (Leave blank for unit address or volser request.)
                            (Enter ALLNAMES to get unitname list.)

UNIT ADDRESS  ===>          (Leave blank for volser request.)
                            (Enter ALL to get entire device list.)
VOLSER        ===>
```

Figure 1: ISPF panel

```
———————————————  EDT UNITNAME LIST  ——  ROW 1 TO 17 OF 45
COMMAND ===> —                          SCROLL ===> HALF

LIST OF ALL AVAILABLE SYSTEM UNIT NAMES

     UNITNAME

     339Ø
     338Ø
     349Ø
     AFP1
     SCTC
     PUBLIC
     RS6KØ
     RS6K1
     RS6K2
     SYSDA
     SYSRQ
     SYSTSO
     SYSVIO
     TAPE
     VIO
     WORK
     348Ø
```

Figure 2: Panel after entering ALLNAMES

- Determine all devices defined in the existing I/O configuration.
- Change the active contents of the UCB for a specific device.

When first entering the ISPF environment for this process you are presented with the ISPF panel shown in Figure 1.

If you enter ALLNAMES for a unit name, you would get an ISPF panel displayed that looks similar to the one shown in Figure 2.

If you have a specific unit name you want to query, the ISPF panel that gets displayed is a list of UCBs that are included in that unit name definition. For example, if you were to enter 3490 for a unit name you would get a display similar to the one shown in Figure 3.

```
————————————  EDT UNIT ADDRESS LIST  ——  ROW 1 TO 1Ø OF 1Ø
COMMAND ===>                                        SCROLL ===> HALF

UNIT ADDRESS LIST FOR UNITNAME  349Ø

     UNIT                              UCB
CMD ADDRESS  VOLSER   DEVICE CODE   ADDRESS    STATUS   PATHS DEFINED

...  Ø3AØ             78048Ø81     ØØF39EC8   ONLINE   D9
...  Ø3A1             78048Ø81     ØØF3A4BØ   ONLINE   DD
...  Ø3A2             78048Ø81     ØØF39FAØ   ONLINE   E1
...  Ø3A3             78048Ø81     ØØF3A15Ø   ONLINE   82
...  Ø3A4             78048Ø81     ØØF3A228   ONLINE   86
...  Ø3A5             78048Ø81     ØØF3A3D8   ONLINE   9E
...  Ø3A6             78048Ø81     ØØF3AØ78   ONLINE   A9
...  Ø3A7             78048Ø81     ØØF3A3ØØ   ONLINE   AA
...  Ø3A8             78048Ø81     ØØF3A588   ONLINE   B2
...  Ø3A9             78048Ø81     ØØF3A66Ø   ONLINE   DA
*************************** BOTTOM OF DATA ****************************
```

Figure 3: Display after entering 3490

As you can see, the above table contains information about the individual devices such as the hex device code, the UCB address, whether the device is on-line or off-line (and in the case of on-line DASD, its volser, and whether the device is mounted PRIVATE, PUBLIC, or STORAGE), and the first four paths that have been defined for each device.

A table similar to the one in Figure 3 would also be displayed if you entered ALL in the unit address field, the difference being that the table would include all UCBs defined within the system. You can drill down to the actual UCB display by entering any character in the 'CMD' field for a specific table entry. For example, if you entered 's' in the CMD field for unit address 03A0 from the panel above you would see a UCB similar to the one shown in Figure 4.

```
──────────────────  UCB MAP  ──────────────────
COMMAND ===>

UCB MAP FOR DEVICE  Ø3AØ

UCB ADDRESS  ØØF39EC8       COMMAND ===>        ('UPDATE' TO CHANGE UCB)

 COMMON AREA                    DEVICE DEPENDENT AREA

 (ØØ) UCBJBNR  - ØØ             (18) UCBFSCT  - ØØØØ
 (Ø1) UCBFL5   - Ø8             (1A) UCBFSEQ  - ØØØØ
 (Ø2) UCBID    - FF             (1C) UCBVOLI  -
 (Ø3) UCBSTAT  - 8Ø             (22) UCBSTAB  - ØØ
 (Ø4) UCBCHAN  - Ø3AØ           (23) UCBDMCT  - ØØ
 (Ø6) UCBFLA   - 4Ø             (24) UCBFSER  - ØØØØØØØØØØØØ
 (Ø7) UCBFLB   - 8Ø             (2A) UCBRDEV  - EØ
 (Ø8) UCBNXUCB - ØØØØØØØØ       (2B) UCBTFL1  - ØØ
 (ØC) UCBWGT   - ØØ             (2C) UCBVOPT  - ØØ
 (ØD) UCBNAME  - 3AØ            (2D) UCBXTNB  - F39EF8
 (1Ø) UCBTYP   - 78048Ø81
 (14) UCBFLC   - Ø4
 (15) UCBEXTP  - F39E78
```

Figure 4: Example UCB

A useful feature of this panel is the ability to alter the contents of the UCB. By over-typing the contents of any of the fields (excluding UCBSTAT, UCBNAME, and UCBVOLI) and entering update in the COMMAND field, you can dynamically alter the contents of the currently displayed UCB. In order to provide security around this process, you must have RACF write access to a dataset name of UCB.UPDATE. A message is issued to your ISPF session as well as to the operator console when a successful update to the UCB has occurred. This will facilitate an audit trail for the UCB update process. Entering a specific unit address or volser from the panel displayed in Figure 1 will lead you to a UCB display similar to Figure 4.

OPERATIONAL ENVIRONMENT

This utility has been tested in an MVS/ESA 4.3 and OS/390 1.2 environment. The code has not been written to support four-digit device numbers. To install this utility the following must be accomplished:

- Include the following statements in an existing ISPF selection menu:

```
%   E +EDT UTILITY  - EDT/UCB DISPLAY/UPDATE UTILITY
          .
          .
          .
                    E,'PGM(EDTISPF)'
```

- BLDEDT must be included as one of the program names of the AUTHTSF parameter of the IKJTSO00 member of SYS1.PARMLIB.
- make EDTISPF available in a load library accessible by ISPF/PDF and link BLDEDT into an authorized (most likely LNKLST) library. You can use the following linkedit options:

```
INCLUDE MODIN(EDTISPF)
ENTRY   EDTISPF
NAME    EDTISPF(R)
INCLUDE MODIN(BLDEDT)
SETCODE AC(1)
NAME    BLDEDT(R)
```

Include the following panel members in an ISPF/PDF panel library.

DEVPNL00 PANEL MEMBER

```
%————————————  ELIGIBLE DEVICE TABLE UTILITY  ————————————-
%
%
%                                HIT 'ENTER' AND THEN
%              PLEASE WAIT FOR INTERNAL TABLE FORMATTING TO COMPLETE
```

DEVPNL01 PANEL MEMBER

```
%————————————  ELIGIBLE DEVICE TABLE UTILITY  ————————————-
%COMMAND ===>_ZCMD
+
%
%
%UNIT NAME    +===>_Z     +  (Leave blank for unit address or volser request.)
```

```
%                        +  (Enter%ALLNAMES+to get unitname list.)
%
%UNIT ADDRESS +===>_Z    +  (Leave blank for volser request.)
%                        +  (Enter%ALL+to get entire device list.)
%
%VOLSER       +===>_Z    +
%
)INIT
  .ZVARS = '(UNITNAME CHARUADR VOLSER)'
  .CURSOR = UNITNAME
  .HELP = DEVHELP1
)PROC
 VER(&CHARUADR,PICT,'CCC')
)END
```

DEVPNL02 PANEL MEMBER

```
)ATTR
  @ TYPE(OUTPUT) INTENS(LOW)
  # TYPE(OUTPUT) INTENS(HIGH)
)BODY
%———————————  EDT UNIT ADDRESS LIST  ———————————-
%COMMAND ===>_ZCMD
%SCROLL===>_VAMT+
%
+UNIT ADDRESS LIST FOR UNITNAME #UNITNAME
%
+     UNIT                              UCB
+CMD ADDRESS  VOLSER   DEVICE CODE   ADDRESS    STATUS   PATHS DEFINED
+
)MODEL
_Z   @Z      @Z        @Z           @Z         @Z       @Z
)INIT
  &VAMT='HALF'
  &ZCSKEY='VAMT'
  .ZVARS='(CMD UNITADDR VOLSER DEVCODE UCBADDR STATUS DEFPATHS)'
)END
```

DEVPNL03 PANEL MEMBER

```
)ATTR
  @ TYPE(OUTPUT) INTENS(LOW)
  # TYPE(OUTPUT) INTENS(HIGH)
)BODY
%———————————————  EDT UNITNAME LIST  ———————————-
%COMMAND ===>_ZCMD
%SCROLL===>_VAMT+
%
+UNITNAME LIST FOR DEVICE #UNITADDR
+   HIT%'ENTER'+TO GET UCB DISPLAY
%
```

```
%     UNITNAME
+
)MODEL
     @Z
)INIT
  &VAMT='HALF'
  &ZCSKEY='VAMT'
  .ZVARS='(UNITNAME)'
)PROC
 VER(&UNITADDR,HEX)
)END
```

DEVPNL04 PANEL MEMBER

```
)ATTR
  @ TYPE(OUTPUT) INTENS(LOW)
  # TYPE(OUTPUT) INTENS(HIGH)
)BODY
%———————————————  UCB MAP  ———————————————
%COMMAND ===>_ZCMD
+
%
+UCB MAP FOR DEVICE #UNITADDR
%
+UCB ADDRESS @Z        +     COMMAND ===> _UCBCMD+  ('UPDATE' TO CHANGE
UCB)
+
+ COMMON AREA
+
+ (ØØ) UCBJBNR  -_Z +
+ (Ø1) UCBFL5   -_Z +
+ (Ø2) UCBID    -_Z +
+ (Ø3) UCBSTAT  -_Z +
+ (Ø4) UCBCHAN  -@Z +
+ (Ø6) UCBFLA   -_Z +
+ (Ø7) UCBFLB   -_Z +
+ (Ø8) UCBNXUCB -_Z +
+ (ØC) UCBWGT   -_Z +
+ (ØD) UCBNAME  -@Z +
+ (1Ø) UCBTYP   -_Z +
+ (14) UCBFLC   -_Z +
+ (15) UCBEXTP  -_Z     +
+
)INIT
  .ZVARS='(UCBADDR $CBJBNR $CBFL5 $CBID $CBSTAT $CBCHAN $CBFLA +
           $CBFLB $CBNXUCB $CBWGT $CBNAM $CBTYP $CBFLC $CBEXTP)'
  .CURSOR = ZCMD
)PROC
  VER(&UCBCMD,LIST,ALTER,UPDATE)
  VER(&$CBJBNR,HEX)
  VER(&$CBFL5,HEX)
  VER(&$CBID,HEX)
```

```
  VER(&$CBSTAT,HEX)
  VER(&$CBFLA,HEX)
  VER(&$CBFLB,HEX)
  VER(&$CBNXUCB,HEX)
  VER(&$CBWGT,HEX)
  VER(&$CBTYP,HEX)
  VER(&$CBFLC,HEX)
  VER(&$CBEXTP,HEX)
)END
```

DEVPNL05 PANEL MEMBER

```
)ATTR
  @ TYPE(OUTPUT) INTENS(LOW)
  # TYPE(OUTPUT) INTENS(HIGH)
)BODY
%————————————————  UCB MAP  ————————————————
%COMMAND ===>_ZCMD
+
%
+UCB MAP FOR DEVICE õUNITADDR
%
+UCB ADDRESS @Z        +     COMMAND ===> _UCBCMD+  ('UPDATE' TO CHANGE UCB)
+
+ COMMON AREA                    DEVICE DEPENDENT AREA
+
+ (ØØ) UCBJBNR  -_Z +            (18) UCBVTOC  -_Z       +
+ (Ø1) UCBFL5   -_Z +            (1C) UCBVOLI  -@Z     +
+ (Ø2) UCBID    -_Z +            (22) UCBSTAB  -_Z +
+ (Ø3) UCBSTAT  -_Z +            (23) UCBDMCT  -_Z +
+ (Ø4) UCBCHAN  -@Z   +          (24) UCBSQC   -_Z +
+ (Ø6) UCBFLA   -_Z +            (25) UCBFL4   -_Z +
+ (Ø7) UCBFLB   -_Z +            (26) UCBUSER  -_Z   +
+ (Ø8) UCBNXUCB -_Z       +      (28) UCBBASE  -_Z       +
+ (ØC) UCBWGT   -_Z +            (2C) UCBNEXP  -_Z       +
+ (ØD) UCBNAME  -@Z  +
+ (1Ø) UCBTYP   -_Z       +
+ (14) UCBFLC   -_Z +
+ (15) UCBEXTP  -_Z     +
+
)INIT
  .ZVARS='(UCBADDR $CBJBNR   $CBVTOC  $CBFL5  $CBVOLI  $CBID   +
           $CBSTAB  $CBSTAT  $CBDMCT  $CBCHAN  $CBSQC  $CBFLA  +
           $CBFL4  $CBFLB  $CBUSER  $CBNXUCB  $CBBASE  $CBWGT  +
           $CBNEXP  $CBNAM  $CBTYP  $CBFLC  $CBEXTP)'
  .CURSOR = ZCMD
)PROC
  VER(&UCBCMD,LIST,ALTER,UPDATE)
  VER(&$CBJBNR,HEX)
  VER(&$CBFL5,HEX)
  VER(&$CBID,HEX)
  VER(&$CBSTAT,HEX)
  VER(&$CBFLA,HEX)
```

```
  VER(&$CBFLB,HEX)
  VER(&$CBNXUCB,HEX)
  VER(&$CBWGT,HEX)
  VER(&$CBTYP,HEX)
  VER(&$CBFLC,HEX)
  VER(&$CBEXTP,HEX)
  VER(&$CBVTOC,HEX)
  VER(&$CBSTAB,HEX)
  VER(&$CBDMCT,HEX)
  VER(&$CBSQC,HEX)
  VER(&$CBFL4,HEX)
  VER(&$CBUSER,HEX)
  VER(&$CBBASE,HEX)
  VER(&$CBNEXP,HEX)
)END
```

DEVPNL06 PANEL MEMBER

```
)ATTR
  @ TYPE(OUTPUT) INTENS(LOW)
  # TYPE(OUTPUT) INTENS(HIGH)
)BODY
%———————————————  UCB MAP  ———————————————————
%COMMAND ===>_ZCMD
+
%
+UCB MAP FOR DEVICE õUNITADDR
%
+UCB ADDRESS @Z        +     COMMAND ===> _UCBCMD+  ('UPDATE' TO CHANGE UCB)
+
+ COMMON AREA                    DEVICE DEPENDENT AREA
+
+ (ØØ) UCBJBNR  -_Z +            (18) UCBFSCT  -_Z +
+ (Ø1) UCBFL5   -_Z +            (1A) UCBFSEQ  -_Z +
+ (Ø2) UCBID    -_Z +            (1C) UCBVOLI  -@Z +
+ (Ø3) UCBSTAT  -_Z +            (22) UCBSTAB  -_Z +
+ (Ø4) UCBCHAN  -@Z +            (23) UCBDMCT  -_Z +
+ (Ø6) UCBFLA   -_Z +            (24) UCBFSER  -_Z +
+ (Ø7) UCBFLB   -_Z +            (2A) UCBRDEV  -_Z +
+ (Ø8) UCBNXUCB -_Z +            (2B) UCBTFL1  -_Z +
+ (ØC) UCBWGT   -_Z +            (2C) UCBVOPT  -_Z +
+ (ØD) UCBNAME  -@Z +            (2D) UCBXTNB  -_Z +
+ (1Ø) UCBTYP   -_Z +
+ (14) UCBFLC   -_Z +
+ (15) UCBEXTP  -_Z +
+
)INIT
  .ZVARS='(UCBADDR $CBJBNR   $CBFSCT  $CBFL5  $CBFSEQ  $CBID   +
           $CBVOLI  $CBSTAT  $CBSTAB  $CBCHAN  $CBDMCT $CBFLA  +
           $CBFSER $CBFLB  $CBRDEV  $CBNXUCB  $CBTFL1  $CBWGT  +
           $CBVOPT  $CBNAM  $CBXTNB  $CBTYP  $CBFLC  $CBEXTP)'
  .CURSOR = ZCMD
)PROC
  VER(&UCBCMD,LIST,ALTER,UPDATE)
```

```
  VER(&$CBJBNR,HEX)
  VER(&$CBFL5,HEX)
  VER(&$CBID,HEX)
  VER(&$CBSTAT,HEX)
  VER(&$CBFLA,HEX)
  VER(&$CBFLB,HEX)
  VER(&$CBNXUCB,HEX)
  VER(&$CBWGT,HEX)
  VER(&$CBTYP,HEX)
  VER(&$CBFLC,HEX)
  VER(&$CBEXTP,HEX)
  VER(&$CBFSCT,HEX)
  VER(&$CBFSEQ,HEX)
  VER(&$CBSTAB,HEX)
  VER(&$CBDMCT,HEX)
  VER(&$CBFSER,HEX)
  VER(&$CBRDEV,HEX)
  VER(&$CBTFL1,HEX)
  VER(&$CBVOPT,HEX)
  VER(&$CBXTNB,HEX)
)END
```

DEVPNL07 PANEL MEMBER

```
)ATTR
  @ TYPE(OUTPUT) INTENS(LOW)
)BODY
%————————————————  EDT UNITNAME LIST  ———————————————-
%COMMAND ===>_ZCMD
%SCROLL===>_VAMT+
%
+LIST OF ALL AVAILABLE SYSTEM UNIT NAMES
%
%     UNITNAME
+
)MODEL
     @Z
)INIT
  &VAMT='HALF'
  &ZCSKEY='VAMT'
  .ZVARS='(UNITNAME)'
)END
```

EDTUM00 MESSAGE MEMBER

This message should be included in an ISPF/PDF message library.

```
EDTUMØØØ   'UCB HAS BEEN UPDATED'  .ALARM=NO
'THE REQUESTED UPDATES FOR THE CURRENT UCB HAVE BEEN MADE'

EDTUMØØ1   'ALTER REQUEST ACCEPTED'  .ALARM=NO
'ALTERED UCB DISPLAYED.  ENTER ''UPDATE'' TO CHANGE THE ACTIVE UCB.'
```

```
EDTUMØØ2   'DDR IN PROGRESS'  .ALARM=YES
'DDR IS ACTIVE FOR THIS DEVICE.  UCB INFORMATION MAY NOT BE ACCURATE.'

EDTUMØØ3   'INVALID UNITNAME'  .ALARM=YES
'THE UNITNAME REQUESTED IS NOT DEFINED TO THE SYSTEM'

EDTUMØØ4   'INVALID UNIT'  .ALARM=YES
'THE UNIT ADDRESS REQUESTED IS NOT DEFINED TO THE SYSTEM'

EDTUMØØ5   'TABLE BUILD ERROR'  .ALARM=YES
'AN ERROR OCCURRED WHILE BUILDING THE EDT/UCB TABLES'

EDTUMØØ7   'UNAUTHORIZED REQUEST'  .ALARM=YES
'USER IS UNAUTHORIZED TO PERFORMED REQUESTED FUNCTION'

EDTUMØØ8   'LOCK UNAVAILABLE'  .ALARM=YES
'THE UCBLOCK IS UNAVAILABLE.  WAIT MOMENTARILY AND RETRY THE REQUEST.'

EDTUMØØ9   'VOLSER NOT FOUND'  .ALARM=YES
'THE REQUESTED VOLSER IS NOT CURRENTLY MOUNTED.'
```

EDTISPF PROGRAM

```
         MACRO
         VDEFINE &FLDNAME=,&VARNAME=,&TYPE=,&LEN=
         L     R15,LINKADDR
         CALL  (15),(VDEFINE,                                          X
               &FLDNAME,                                               X
               &VARNAME,                                               X
               &TYPE,                                                  X
               &LEN),                                                  X
               VL,MF=(E,ISPLINK)
         MEND
EDTISPF  CSECT
         PRINT NOGEN
         STM   R14,R12,12(R13)           SAVE INCOMING ENVIRONMENT
         LR    R11,R15                   SAVE MODULE BASE ADDRESS
         USING EDTISPF,R11,R12
         LA    R12,4Ø95(,R11)            SET SECOND BASE ...
         LA    R12,1(,R12)                  REGISTER VALUE
         USING UCBOB,R7                  SET UCB BASE REG
         ST    R13,SAVEAREA+4            SAVE SAVEAREA ADDRESS
         LA    R13,SAVEAREA              LOAD R13 WITH NEW SAVEAREA
         GBLC  &SYSSPLV
         SPLEVEL SET=1
*   SET UP ISPF ENVIRONMENT
         LOAD  EP=ISPLINK                GET ISPLINK ADDRESS
         ST    RØ,LINKADDR               SAVE ADDRESS
         VDEFINE FLDNAME=XL2VARS,VARNAME=UNITADDR,TYPE=HEX,LEN=L2
         VDEFINE FLDNAME=CL3VARS,VARNAME=CMD,TYPE=CHAR,LEN=L3
         VDEFINE FLDNAME=XL4VARS,VARNAME=DEVCODE,TYPE=HEX,LEN=L4
         VDEFINE FLDNAME=CL6VARS,VARNAME=VOLSER,TYPE=CHAR,LEN=L6
```

```
         VDEFINE FLDNAME=CL7VARS,VARNAME=STATUS,TYPE=CHAR,LEN=L7
         VDEFINE FLDNAME=CL8VARS,VARNAME=UNITNAME,TYPE=CHAR,LEN=L8
         VDEFINE FLDNAME=CL23VARS,VARNAME=DEFPATHS,TYPE=CHAR,LEN=L23
         VDEFINE FLDNAME=CL8ØVARS,VARNAME=ZCMD,TYPE=CHAR,LEN=L8Ø
         VDEFINE FLDNAME=UCBVARØ1,VARNAME=$CBJBNR,TYPE=HEX,LEN=L1
         VDEFINE FLDNAME=UCBVARØ2,VARNAME=$CBFL5,TYPE=HEX,LEN=L1
         VDEFINE FLDNAME=UCBVARØ3,VARNAME=$CBID,TYPE=HEX,LEN=L1
         VDEFINE FLDNAME=UCBVARØ4,VARNAME=$CBSTAT,TYPE=HEX,LEN=L1
         VDEFINE FLDNAME=UCBVARØ5,VARNAME=$CBCHAN,TYPE=HEX,LEN=L2
         VDEFINE FLDNAME=UCBVARØ6,VARNAME=$CBFLA,TYPE=HEX,LEN=L1
         VDEFINE FLDNAME=UCBVARØ7,VARNAME=$CBFLB,TYPE=HEX,LEN=L1
         VDEFINE FLDNAME=UCBVARØ8,VARNAME=$CBNXUCB,TYPE=HEX,LEN=L4
         VDEFINE FLDNAME=UCBVARØ9,VARNAME=$CBWGT,TYPE=HEX,LEN=L1
         VDEFINE FLDNAME=UCBVAR1Ø,VARNAME=$CBNAME,TYPE=CHAR,LEN=L3
         VDEFINE FLDNAME=UCBVAR11,VARNAME=$CBTYP,TYPE=HEX,LEN=L4
         VDEFINE FLDNAME=UCBVAR12,VARNAME=$CBFLC,TYPE=HEX,LEN=L1
         VDEFINE FLDNAME=UCBVAR13,VARNAME=$CBEXTP,TYPE=HEX,LEN=L3
         VDEFINE FLDNAME=UCBVAR14,VARNAME=$CBVTOC,TYPE=HEX,LEN=L4
         VDEFINE FLDNAME=UCBVAR15,VARNAME=$CBVOLI,TYPE=CHAR,LEN=L6
         VDEFINE FLDNAME=UCBVAR16,VARNAME=$CBSTAB,TYPE=HEX,LEN=L1
         VDEFINE FLDNAME=UCBVAR17,VARNAME=$CBDMCT,TYPE=HEX,LEN=L1
         VDEFINE FLDNAME=UCBVAR18,VARNAME=$CBSQC,TYPE=HEX,LEN=L1
         VDEFINE FLDNAME=UCBVAR19,VARNAME=$CBFL4,TYPE=HEX,LEN=L1
         VDEFINE FLDNAME=UCBVAR2Ø,VARNAME=$CBUSER,TYPE=HEX,LEN=L2
         VDEFINE FLDNAME=UCBVAR21,VARNAME=$CBBASE,TYPE=HEX,LEN=L4
         VDEFINE FLDNAME=UCBVAR22,VARNAME=$CBNEXP,TYPE=HEX,LEN=L4
         VDEFINE FLDNAME=UCBVAR23,VARNAME=$CBFSCT,TYPE=HEX,LEN=L2
         VDEFINE FLDNAME=UCBVAR24,VARNAME=$CBFSEQ,TYPE=HEX,LEN=L2
         VDEFINE FLDNAME=UCBVAR25,VARNAME=$CBFSER,TYPE=HEX,LEN=L6
         VDEFINE FLDNAME=UCBVAR26,VARNAME=$CBRDEV,TYPE=HEX,LEN=L1
         VDEFINE FLDNAME=UCBVAR27,VARNAME=$CBTFL1,TYPE=HEX,LEN=L1
         VDEFINE FLDNAME=UCBVAR28,VARNAME=$CBVOPT,TYPE=HEX,LEN=L1
         VDEFINE FLDNAME=UCBVAR29,VARNAME=$CBXTNB,TYPE=HEX,LEN=L3
         LA    R1,PARMAREA                GET PARM ADDR FOR TSF
         STCM  R1,15,PGMPARM1+2           SAVE IN PARAMETER LIST
         L     R15,16                     GET CVT ADDRESS
         L     R15,CVTTVT-CVT(,R15)       GET TSO VECTOR TABLE ADDRESS
         L     R15,TSVTASF-TSVT(,R15)     GET TSO SERVICE ROUTINE ADDR
         MVI   TSFFLAG,TBLBUILD           SET THE 'BUILD TABLE' FLAG
         CALL  (15),(TSFFLAG1,PGMNAME1,BUFLEN1,RETCODE1,RSNCODE2,      X
               ABNDCOD1,PARMLST1),VL
         CLC   RETCODE1(4),=F'Ø'          DID WE BUILD A TABLE?
         BNE   BUILDERR                   NO - ISSUE AN ERROR
         B     PASTPARM
TSFFLAG1 DS    ØF
         DC    X'ØØ'
         DC    X'Ø1'
         DC    X'Ø1'
         DC    X'Ø2'
PGMNAME1 DC    C'BLDEDT'
BUFLEN1  DC    F'6'
RETCODE1 DS    F
RSNCODE2 DS    F
```

```
ABNDCOD1 DS    F
PGMPARM1 DC    H'4'
         DS    XL4
PARMLST1 CALL  ,(PGMPARM1),VL,MF=L
PASTPARM EQU   *
         CLC   RETCODE1(4),=F'Ø'          DID WE BUILD A TABLE?
         BE    SETPANEL                   NO - ISSUE AN ERROR
DISPLAYØ MVC   PRIMEPNL(8),DEVPNLØØ       MOVE IN PANEL NAME
DISPLAY1 MVC   ZCMD(8Ø),=8ØC' '           CLEAR ZCMD
         MVC   UNITNAME(8),=8C' '         CLEAR UNITNAME
         MVC   CHARUADR(3),=8C' '         CLEAR UNIT ADDRESS
         MVC   VOLSER(6),=8C' '           CLEAR VOLSER
         L     R15,LINKADDR               LOAD ISPLINK ADDRESS
         L     R3,MSGADDR                 LOAD MESSAGE ADDRESS
         CALL  (15),(DISPLAY,PRIMEPNL,(R3)),VL,MF=(E,ISPLINK)
         LTR   R15,R15                    DISPLAY OK?
         BZ    GETINFO                    YES - GET INFORMATION
         B     RETURN                     NO - QUIT
GETINFO  EQU   *
         TM    FLAG,BLDERR                TABLE BUILD ERROR?
         BO    DISPLAYØ                   ISSUE ERROR
SETPANEL LA    R3,BLANKMSG                GET A MESSAGE ADDRESS
         ST    R3,MSGADDR                 SAVE IT
         TM    FLAG,GOTEDT                DONE THIS ALREADY?
         BO    NOBUILD                    YES - DON'T WAIT
         OI    FLAG,GOTEDT                NO - SET THE SUCCESS FLAG
         XR    R4,R4                      CLEAR R4
         L     R5,NUMDEVT                GET NUMBER OF UNIQUE UNITNAMES
         M     R4,=F'12'                  GET TABLE LENGTH
         ST    R5,TBLLEN                  SAVE IT
         MVC   PRIMEPNL(8),DEVPNLØ1       MOVE IN PANEL NAME
         B     DISPLAY1                   GO DISPLAY NEW PANEL
NOBUILD  CLC   UNITNAME(8),=C'ALLNAMES'   ALL UNITNAMES?
         BE    SETTBLØ3                   YES - BUILD ALLNAMES TABLE
         CLC   UNITNAME(8),=8C' '         BLANKS?
         BNE   SETTBLØ1                   NO - SET TO TABLE 1
         CLC   CHARUADR(3),=C'ALL'        ALL DEVICES?
         BNE   CHKNXTUA                   NO - CHECK FOR UNIT ADDRESS
         MVC   UNITNAME(7),=C'ALLDEVS'    SET UNITNAME TO ALLDEVS
         B     SETTBLØ1                   SET TABLE ID
CHKNXTUA CLC   CHARUADR(3),=8C' '         BLANKS?
         BNE   UADDR                      NO - DO UNIT ADDRESS
         CLC   VOLSER(6),=8C' '           A VOLSER?
         BE    DISPLAY1                   NO - GO REDISPLAY
         XC    SCANWORK(1ØØ),SCANWORK     CLEAR THE WORK AREA
         UCBSCAN COPY,WORKAREA=SCANWORK,UCBAREA=AREA4UCB,VOLSER=VOLSER,X
               DEVCLASS=ALL,DYNAMIC=YES
         LA    R7,AREA4UCB                GET UCB AREA ADDRESS
         LTR   R15,R15                    DID WE FIND A UCB?
         BNZ   VOLERR                     NO - ISSUE AN ERROR
         B     SAVEUADR                   GO ON
SAVEUADR MVC   CHARUADR(3),UCBNAME        SAVE THE UNIT ADDRESS
UADDR    MVI   DBL1,C'Ø'                  PAD FIRST BYTE
         MVC   DBL1+1(3),CHARUADR         MOVE IN DEVICE NUMBER
```

```
         TR    DBL1(4),TRTABLE           TRANSLATE
         L     R15,=F'3'                 SET LOOP COUNTER
         LA    R14,DBL1+1                GET STARTING ADDRESS
DEVNUMLP CLI   Ø(R14),X'8Ø'              A BAD DIGIT?
         BE    UADDRERR                  YES - ISSUE AN ERROR
         LA    R14,1(,R14)               SET TO NEXT BYTE
         BCT   R15,DEVNUMLP              CHECK OUT NEXT BYTE
         PACK  DBL2(8),DBL1(4)           PACK
         L     R15,DBL2+4                GET PACKED VALUE
         SRL   R15,4                     GET RID OF SIGN
         STCM  R15,B'ØØ11',UNITADDR      SAVE BINARY UNIT ADDRESS
         L     R3,UCBCHAIN               GET UCB CHAIN ANCHOR ADDRESS
CHKUADDR EQU   *
         LTR   R3,R3                     END OF CHAIN?
         BZ    UADDRERR                  YES - UADDR DOESN'T EXIST
         CLC   UNITADDR(2),2(R3)         UCB ADDRESS MATCH?
         BE    OURUADDR                  YES - GO PROCESS
         BL    UADDRERR                  LOW - UNIT ADDR DOESN'T EXIST
         L     R3,4(,R3)                 POINT TO NEXT ENTRY
         B     CHKUADDR                  GO CHECK IT OUT
OURUADDR EQU   *
         L     R5,8(,R3)                 GET DEVICE ENTRY ADDRESS
         L     R7,12(,R5)                GET UCB ADDRESS
         ST    R7,UCBADDR                SAVE UCB ADDRESS
         LA    R3,12(,R3)                POINT TO FIRST ENTRY
SETTBLØ2 MVC   TBLNAME(8),TBLNAME2       MOVE IN TABLE 2 NAME
         LA    R5,TBLISTØ2               GET TABLE LIST ADDRESS
         B     TBLCR                     CREATE TABLE
SETTBLØ3 MVC   TBLNAME(8),TBLNAME3       MOVE IN TABLE 3 NAME
         LA    R5,TBLISTØ2               GET TABLE LIST ADDRESS
         B     TBLCR                     CREATE TABLE
SETTBLØ1 MVC   TBLNAME(8),TBLNAME1       MOVE IN TABLE 1 NAME
         LA    R5,TBLISTØ1               GET TABLE LIST ADDRESS
TBLCR    L     R15,LINKADDR              LOAD ISPLINK ADDRESS
         CALL  (15),(TBCREATE,TBLNAME,NOKEY,(R5),NOWRITE,REPLACE),     X
               VL,MF=(E,ISPLINK)
         C     R15,=F'4'                 SUCCESSFUL CREATE?
         BH    RETURN                    NO - CLEAN UP AND PACK IT IN
         CLC   TBLNAME(8),TBLNAME1       TABLE 1?
         BE    BLDTBL1                   YES - BUILD TABLE 1
         CLC   TBLNAME(8),TBLNAME2       TABLE 2?
         BE    BLDTBL2                   YES - BUILD TABLE 2
BLDTBL3  L     R8,MAINTBL                GET MAIN TABLE ADDRESS
         L     R5,NUMDEVT                GET MAX LOOP COUNT
TBL3LOOP MVC   UNITNAME(8),Ø(R8)         MOVE IN UNITNAME
         LA    R6,TBLISTØ2               GET TABLE LIST ADDRESS
         L     R15,LINKADDR              GET ISPLINK ADDRESS
         CALL  (15),(TBADD,TBLNAME3,(R6)),VL,MF=(E,ISPLINK)
         LA    R8,12(,R8)                SET TO NEXT MAIN TBL ENTRY
         BCT   R5,TBL3LOOP               GO BACK IF MORE
         B     DISPLAY2                  GO DISPLAY THE TABLE
BLDTBL2  EQU   *
         LTR   R3,R3                     AT THE END?
```

```
         BZ    DISPLAY2                  YES - GO DISPLAY THE TABLE
         L     R4,Ø(,R3)                 GET ADDRESS OF UNITNAME
         MVC   UNITNAME(8),Ø(R4)         MOVE IT IN
         LA    R5,TBLISTØ2               GET TABLE LIST ADDRESS
         L     R15,LINKADDR              GET ISPLINK ADDRESS
         CALL  (15),(TBADD,TBLNAME2,(R5)),VL,MF=(E,ISPLINK)
         L     R3,4(,R3)                 SET POINTER TO NEXT ENTRY
         B     BLDTBL2                   BUILD NEXT ENTRY
BLDTBL1  EQU   *
         L     R8,MAINTBL                GET MAIN TABLE ADDRESS
         L     R5,NUMDEVT                GET MAX LOOP COUNT
         MVC   CMD(3),=C'...'            MOVE IN COMMAND FIELD
         CLC   UNITNAME(7),=C'ALLDEVS'   ALL DEVICES?
         BE    ALLDEVS                   YES - GET ALL DEVICES
FINDUNIT EQU   *
         CLC   Ø(8,R8),UNITNAME          OUR UNIT?
         BE    OURUNIT                   YES - GO GET INFO
         LA    R8,12(,R8)                POINT TO NEXT TABLE ENTRY
         BCT   R5,FINDUNIT               LOOP BACK
         B     UNITERR                   ISSUE ERROR
OURUNIT  L     R2,8(,R8)                 GET ANCHOR ADDRESS
ADDCHECK LTR   R2,R2                     ALL DONE?
         BZ    DISPLAY2                  YES - GO DISPLAY
ADDDEV   MVC   UNITADDR(2),1Ø(R2)        MOVE IN UNIT ADDRESS
         L     R7,12(,R2)                GET UCB ADDRESS
         ST    R7,UCBADDR                SAVE UCB ADDRESS
         L     R4,=F'8'                  SET LOOP MAXIMUM
         LA    R5,16(,R2)                GET SOURCE AREA ADDRESS
         MVC   DEFPATHS(23),=8ØC' '      CLEAR OUT THE AREA
         LA    R3,DEFPATHS               GET AREA ADDRESS
OURDEV   EQU   *
         CLC   Ø(2,R5),=2C' '            BLANKS?
         BE    LASTCHN                   YES - PATH LIST IS DONE
         MVC   Ø(2,R3),Ø(R5)             MOVE IN PATH ID
         LA    R5,2(,R5)                 POINT TO NEXT POSSIBLE ONE
         LA    R3,3(,R3)                 POINT TO NEXT TARGET AREA
         BCT   R4,OURDEV                 CHECK IT OUT
LASTCHN  EQU   *
         MVC   VOLSER(6),=8C' '          CLEAR OUT VOLSER AREA
         TM    UCBSTAT,UCBONLI           ON-LINE?
         BNO   OFFLINE                   NO - SET OFF-LINE STATUS
         TM    UCBTBYT3,UCB3DACC         DASD?
         BO    CHKSTAT                   YES - CHECK STATUS
         B     ONLINE                    NO - SET ON-LINE INDICATOR
CHKSTAT  MVC   VOLSER(6),UCBVOLI         MOVE IN VOLSER
         TM    UCBSTAB,UCBBPRV           PRIVATE?
         BO    PRIVATE                   YES - SET PRIVATE STATUS
         TM    UCBSTAB,UCBBPUB           PUBLIC?
         BO    PUBLIC                    YES - SET PUBLIC STATUS
         TM    UCBSTAB,UCBBSTR           STORAGE?
         BO    STORAGE                   YES - SET STORAGE STATUS
         B     GETDEVT                   GET DEVICE TYPE
PRIVATE  MVC   STATUS(7),=C'PRIVATE'     MOVE IN MOUNT ATTRIBUTE
         B     GETDEVT                   GET DEVICE TYPE
```

```
PUBLIC   MVC   STATUS(7),=C'PUBLIC '      MOVE IN MOUNT ATTRIBUTE
         B     GETDEVT                   GET DEVICE TYPE
STORAGE  MVC   STATUS(7),=C'STORAGE'      MOVE IN MOUNT ATTRIBUTE
         B     GETDEVT                   GET DEVICE TYPE
OFFLINE  MVC   STATUS(7),=C'OFFLINE'      MOVE IN MOUNT ATTRIBUTE
         B     GETDEVT                   GET DEVICE TYPE
ONLINE   MVC   STATUS(7),=C'ONLINE '      MOVE IN MOUNT ATTRIBUTE
         B     GETDEVT                   GET DEVICE TYPE
GETDEVT  EQU   *
         MVC   DEVCODE(4),UCBTYP          MOVE IN DEVICE TYPE
         LA    R5,TBLISTØ1                GET TABLE LIST ADDRESS
         L     R15,LINKADDR               GET ISPLINK ADDRESS
         CALL  (15),(TBADD,TBLNAME1,(R5)),VL,MF=(E,ISPLINK)
         CLC   UNITNAME(7),=C'ALLDEVS'    ALL DEVICES?
         BE    ALLDEVS1                   YES - PROCESS DEVICE LIST
         L     R2,Ø(,R2)                  GET NEXT ADDRESS
         B     ADDCHECK                   CHECK IF ANOTHER UNIT
ALLDEVS  L     R6,UCBCHAIN                GET STARTING ENTRY
         B     ALLDEVS2                   GET GOING
ALLDEVS1 L     R6,4(,R6)                  GET NEXT ENTRY ADDRESS
ALLDEVS2 LTR   R6,R6                      ANY MORE?
         BZ    DISPLAY2                   NO - GO DISPLAY PANEL
         L     R2,8(,R6)                  GET DEVICE ENTRY ADDRESS
         B     ADDDEV                     GO ADD TO TABLE
DISPLAY2 EQU   *
         L     R15,LINKADDR               LOAD ISPLINK ADDRESS
         CALL  (15),(TBTOP,TBLNAME),VL,MF=(E,ISPLINK)
         LA    R3,BLANKMSG                LOAD MESSAGE ADDRESS
         XC    CRP(4),CRP                 CLEAR ROW POINTER
         CLC   TBLNAME(8),TBLNAME1        TABLE 1?
         BE    NAME1                      YES - SET TABLE 1 PANEL NAME
         CLC   TBLNAME(8),TBLNAME2        TABLE 2?
         BE    NAME2                      YES - SET TABLE 2 PANEL NAME
         MVC   TBLPNLNM(8),DEVPNLØ7       MOVE IN PANEL NAME
         B     TBDISP                     GO DISPLAY
NAME1    MVC   TBLPNLNM(8),DEVPNLØ2       SET TABLE 1 PANEL NAME
         B     TBDISP                     GO DISPLAY
NAME2    MVC   TBLPNLNM(8),DEVPNLØ3       SET TABLE 2 PANEL NAME
TBDISP   L     R15,LINKADDR               LOAD ISPLINK ADDRESS
         CALL  (15),(TBDISPL,TBLNAME,TBLPNLNM,(R3),BLANKMSG,CRP),      X
               VL,MF=(E,ISPLINK)
         C     R15,=F'8'                  FINISHED?
         BNE   CHKDISP                    NO - CHECK FOR DISPLAY
         MVC   VERBAREA(8),=8C' '         CLEAR OUT AREA
         MVC   VL8(4),L8                  MOVE IN VERB LENGTH
         L     R15,LINKADDR               LOAD ISPLINK ADDRESS
         CALL  (15),(VCOPY,ZVERB,VL8,VERBAREA,MOVE),VL,MF=(E,ISPLINK)
         CLC   VERBAREA(6),=C'RETURN'     VERB = 'RETURN'?
         BE    RETURN                     YES - CLEAN UP & QUIT
         B     CLOSETBL                   NO - CLOSE TABLE
CHKDISP  EQU   *
         ST    R15,RTNCODE                SAVE RETURN CODE
         L     R15,LINKADDR               LOAD ISPLINK ADDRESS
```

```
         CALL  (15),(CONTROL,DISPLAY,SAVE),VL,MF=(E,ISPLINK)
CHKDISP1 L     R7,UCBADDR                 GET UCB ADDRESS
         MVC   $CBJBNR($CBLEN),Ø(R7)      MOVE VALUES OVER
         TM    UCBTBYT3,X'2Ø'             DASD?
         BNO   TAPECHK                    NO - CHECK FOR TAPE
         MVC   $CBVTOC(DASDDDSL),$CBLEN(R7) MOVE IN DASD DDS INFO
         MVC   PNLNAME(8),DEVPNLØ5        MOVE IN PANEL NAME
         XR    R14,R14                    CLEAR R14
         LA    R14,$CBLEN+DASDDDSL        GET LENGTH
         ST    R14,SVUCBLEN               SAVE THE LENGTH
         B     UCBPNLD                    GO DISPLAY UCB PANEL
TAPECHK  TM    UCBTBYT3,X'8Ø'             TAPE?
         BNO   OTHERUCB                   NO - MUST BE ANOTHER TYPE
         MVC   $CBFSCT(TAPEDDSL),$CBLEN(R7) MOVE IN TAPE DDS INFO
         MVC   PNLNAME(8),DEVPNLØ6        MOVE IN PANEL NAME
         XR    R14,R14                    CLEAR R14
         LA    R14,$CBLEN+TAPEDDSL        GET LENGTH
         ST    R14,SVUCBLEN               SAVE THE LENGTH
         B     UCBPNLD                    GO DISPLAY UCB PANEL
OTHERUCB MVC   PNLNAME(8),DEVPNLØ4        MOVE IN PANEL NAME
         XR    R14,R14                    CLEAR R14
         LA    R14,$CBLEN                 GET LENGTH
         ST    R14,SVUCBLEN               SAVE THE LENGTH
UCBPNLD  L     R15,LINKADDR               LOAD ISPLINK ADDRESS
         L     R3,MSGADDR                 LOAD MESSAGE ADDRESS
         TM    UCBFLC,UCBDDRSW            DDR IN PROGRESS?
         BZ    CLEARCMD                   NO - CLEAR COMMAND AREA
         LA    R3,EDTUMØØ2                SET MESSAGE ADDRESS
CLEARCMD MVC   UCBCMD(6),=6C' '           CLEAR IT
         CALL  (15),(DISPLAY,PNLNAME,(R3)),VL,MF=(E,ISPLINK)
         LTR   R15,R15                    DISPLAY OK?
         BZ    CHKCMD                     YES - GET INFORMATION
         LA    R3,BLANKMSG                GET MESSAGE ADDRESS
         ST    R3,MSGADDR                 SAVE IT
         MVC   VERBAREA(8),=8C' '         CLEAR OUT TARGET AREA
         MVC   VL8(4),L8                  MOVE IN VERB LENGTH
         L     R15,LINKADDR               LOAD ISPLINK ADDRESS
         CALL  (15),(VCOPY,ZVERB,VL8,VERBAREA,MOVE),VL,MF=(E,ISPLINK)
         CLC   VERBAREA(6),=C'RETURN'     VERB = 'RETURN'?
         BE    RETURN                     YES - CLEAN UP & QUIT
         CLC   VERBAREA(3),=C'END'        VERB = 'END'?
         BE    CHKRTNC                    YES - GO CHECK RETURN CODE
         B     RETURN                     CLEAN UP AND QUIT
CHKCMD   EQU   *
         CLC   UCBCMD(6),=C'UPDATE'       UPDATE REQUEST?
         BNE   CHKDISP1                   NO - GO BACK
STARTACF EQU   *
         L     R15,16                     GET CVT ADDRESS
         L     R15,CVTTVT-CVT(,R15)       GET TSO VECTOR TABLE ADDRESS
         L     R15,TSVTASF-TSVT(,R15)     GET TSO SERVICE ROUTINE ADDR
         MVI   TSFFLAG,UCBUPDAT           SET THE 'UPDATE UCB' FLAG
         CALL  (15),(TSFFLAG1,PGMNAME1,BUFLEN1,RETCODE1,RSNCODE2,      X
               ABNDCOD1,PARMLST1),VL
```

```
         CLC   RETCODE1(4),=F'Ø'          UPDATE WAS SUCCESSFUL?
         BE    MSGDUMP                    YES - GO WRITE SOME MESSAGES
         CLC   RETCODE1(4),=F'4'          AN AUTHORIZATION ERROR?
         BE    AUTHERR                    YES - ISSUE THAT ERROR MSG
         CLC   RETCODE1(4),=F'8'          AN ERROR GETTING THE LOCK?
         BE    LOCKERR                    YES - ISSUE THAT ERROR MSG
MSGDUMP  EQU   *
         MVC   VL8(4),L8                  MOVE IN VERB LENGTH
         L     R15,LINKADDR               LOAD ISPLINK ADDRESS
         CALL  (15),(VCOPY,ZUSER,VL8,USERID,MOVE),VL,MF=(E,ISPLINK)
         ST    R7,DBL2                    STORE UCB ADDRESS
         UNPK  DBL1(9),DBL2(5)            UNPACK IT
         NC    DBL1(8),=8X'ØF'            CLEAR HIGH NIBBLES
         TR    DBL1(8),TRTABLE            MAKE READABLE
         MVC   WTOWORK1(WTOLEN1),WTOLIST1 MOVE IN WTO MASK
         MVC   WTOWORK1+22(8),DBL1        MOVE IN UCB ADDRESS
         MVC   WTOWORK1+42(8),USERID      MOVE IN USER ID
         WTO   MF=(E,WTOWORK1)            ISSUE THE WTO
         LA    R3,EDTUMØØØ                GET MESSAGE ADDRESS
         ST    R3,MSGADDR                 SAVE IT
         B     CHKDISP1                   GO BACK
CHKRTNC  EQU   *
         CLC   RTNCODE(4),=F'Ø'           ZERO?
         BE    TBDISP                     GO DISPLAY THE TABLE
         L     R15,LINKADDR               LOAD ISPLINK ADDRESS
         CALL  (15),(CONTROL,DISPLAY,RESTORE),VL,MF=(E,ISPLINK)
         L     R15,LINKADDR               LOAD ISPLINK ADDRESS
         CALL  (15),(TBDISPL,TBLNAME),VL,MF=(E,ISPLINK)
         B     CHKDISP                    GO DISPLAY NEXT ONE
CLOSETBL L     R15,LINKADDR               LOAD ISPLINK ADDRESS
         CALL  (15),(TBCLOSE,TBLNAME),VL,MF=(E,ISPLINK)
         LTR   R15,R15                    CLOSE OK?
         BNZ   RETURN                     NO - END
         B     DISPLAY1                   DISPLAY PANEL
UNITERR  LA    R5,EDTUMØØ3                GET MESSAGE ADDRESS
         ST    R5,MSGADDR                 SAVE IT
         B     DISPLAY1                   DISPLAY PANEL
UADDRERR LA    R5,EDTUMØØ4                GET MESSAGE ADDRESS
         ST    R5,MSGADDR                 SAVE IT
         B     DISPLAY1                   DISPLAY PANEL
BUILDERR LA    R3,EDTUMØØ5                GET MESSAGE ADDRESS
         ST    R3,MSGADDR                 SAVE IT
         OI    FLAG,BLDERR                SET ERROR FLAG
         B     DISPLAYØ                   DISPLAY PANEL
AUTHERR  LA    R14,EDTUMØØ7               GET MESSAGE ADDRESS
         ST    R14,MSGADDR                SAVE IT
         B     CHKDISP                    DISPLAY PANEL
LOCKERR  MODESET MODE=PROB,KEY=NZERO      GET BACK TO NORMAL
         LA    R3,EDTUMØØ8                GET MESSAGE ADDRESS
         ST    R3,MSGADDR                 SAVE IT
         B     CHKDISP1                   GO BACK
VOLERR   LA    R5,EDTUMØØ9                GET MESSAGE ADDRESS
         ST    R5,MSGADDR                 SAVE IT
         B     DISPLAY1                   DISPLAY PANEL
```

```
RETURN   EQU   *
         TM    FLAG,BLDERR               BLDERR FLAG SET?
         BO    END                       YES - DON'T FREE ANY STORAGE
         L     R15,16                    GET CVT ADDRESS
         L     R15,CVTTVT-CVT(,R15)      GET TSO VECTOR TABLE ADDRESS
         L     R15,TSVTASF-TSVT(,R15)    GET TSO SERVICE ROUTINE ADDR
         MVI   TSFFLAG,TBLFREE           SET THE 'FREE TABLE' FLAG
         CALL  (15),(TSFFLAG1,PGMNAME1,BUFLEN1,RETCODE1,RSNCODE2,      X
               ABNDCOD1,PARMLST1),VL
         B     END
END      EQU   *
         L     R13,SAVEAREA+4            GET SAVEAREA ADDRESS
         LM    R14,R12,12(R13)           RESTORE ENVIRONMENT
         XR    R15,R15                   CLEAR R15
         BR    R14                       RETURN
         $REQU
TRTABLE  DC    255X'8Ø'
         ORG   TRTABLE+Ø
         DC    C'Ø123456789ABCDEF'
         ORG   TRTABLE+193
         DC    X'ØAØBØCØDØEØF'
         ORG   TRTABLE+24Ø
         DC    X'ØØØ1Ø2Ø3Ø4Ø5Ø6Ø7Ø8Ø9'
         ORG   ,
*   ISPF CONSTANTS AND VARIABLES
XL2VARS  DC    C'(UNITADDR)'
CL3VARS  DC    C'(CMD CHARUADR)'
XL4VARS  DC    C'(DEVCODE UCBADDR)'
CL6VARS  DC    C'(VOLSER DEVGEN UCBCMD)'
CL7VARS  DC    C'(STATUS)'
CL8VARS  DC    C'(UNITNAME)'
CL23VARS DC    C'(DEFPATHS)'
CL8ØVARS DC    C'(ZCMD)'
UCBVARØ1 DC    C'($CBJBNR)'
UCBVARØ2 DC    C'($CBFL5)'
UCBVARØ3 DC    C'($CBID)'
UCBVARØ4 DC    C'($CBSTAT)'
UCBVARØ5 DC    C'($CBCHAN)'
UCBVARØ6 DC    C'($CBFLA)'
UCBVARØ7 DC    C'($CBFLB)'
UCBVARØ8 DC    C'($CBNXUCB)'
UCBVARØ9 DC    C'($CBWGT)'
UCBVAR1Ø DC    C'($CBNAM)'
UCBVAR11 DC    C'($CBTYP)'
UCBVAR12 DC    C'($CBFLC)'
UCBVAR13 DC    C'($CBEXTP)'
UCBVAR14 DC    C'($CBVTOC)'
UCBVAR15 DC    C'($CBVOLI)'
UCBVAR16 DC    C'($CBSTAB)'
UCBVAR17 DC    C'($CBDMCT)'
UCBVAR18 DC    C'($CBSQC)'
UCBVAR19 DC    C'($CBFL4)'
UCBVAR2Ø DC    C'($CBUSER)'
UCBVAR21 DC    C'($CBBASE)'
```

```
UCBVAR22 DC    C'($CBNEXP)'
UCBVAR23 DC    C'($CBFSCT)'
UCBVAR24 DC    C'($CBFSEQ)'
UCBVAR25 DC    C'($CBFSER)'
UCBVAR26 DC    C'($CBRDEV)'
UCBVAR27 DC    C'($CBTFL1)'
UCBVAR28 DC    C'($CBVOPT)'
UCBVAR29 DC    C'($CBXTNB)'
VDEFINE  DC    C'VDEFINE'
VCOPY    DC    C'VCOPY'
DISPLAY  DC    C'DISPLAY'
CONTROL  DC    C'CONTROL'
SAVE     DC    C'SAVE'
RESTORE  DC    C'RESTORE'
TBCREATE DC    C'TBCREATE'
NOWRITE  DC    C'NOWRITE'
REPLACE  DC    C'REPLACE'
NOKEY    DC    CL8Ø' '
ZVERB    DC    CL8'ZVERB'
ZUSER    DC    CL8'ZUSER'
MOVE     DC    CL8'MOVE'
TBADD    DC    C'TBADD'
TBTOP    DC    C'TBTOP'
TBCLOSE  DC    C'TBCLOSE'
TBDISPL  DC    C'TBDISPL'
TBLNAME  DS    CL8
TBLNAME1 DC    C'DEVTBLØ1'
TBLNAME2 DC    C'DEVTBLØ2'
TBLNAME3 DC    C'DEVTBLØ3'
TBLISTØ1 DC    C'(CMD UNITADDR VOLSER DEVCODE UCBADDR STATUS DEFPATHS)'
TBLISTØ2 DC    C'(UNITNAME)'
TBLPNLNM DS    CL8
DEVPNLØØ DC    C'DEVPNLØØ'
DEVPNLØ1 DC    C'DEVPNLØ1'
DEVPNLØ2 DC    C'DEVPNLØ2'
DEVPNLØ3 DC    C'DEVPNLØ3'
DEVPNLØ4 DC    C'DEVPNLØ4'
DEVPNLØ5 DC    C'DEVPNLØ5'
DEVPNLØ6 DC    C'DEVPNLØ6'
DEVPNLØ7 DC    C'DEVPNLØ7'
PNLNAME  DS    CL8
PRIMEPNL DS    CL8
BLANKMSG DC    CL8' '
EDTUMØØØ DC    C'EDTUMØØØ'
EDTUMØØ1 DC    C'EDTUMØØ1'
EDTUMØØ2 DC    C'EDTUMØØ2'
EDTUMØØ3 DC    C'EDTUMØØ3'
EDTUMØØ4 DC    C'EDTUMØØ4'
EDTUMØØ5 DC    C'EDTUMØØ5'
EDTUMØØ7 DC    C'EDTUMØØ7'
EDTUMØØ8 DC    C'EDTUMØØ8'
EDTUMØØ9 DC    C'EDTUMØØ9'
CHAR     DC    CL8'CHAR'
```

```
HEX      DC    CL8'HEX'
L1       DC    F'1'
L2       DC    F'2'
L3       DC    F'3'
L4       DC    F'4'
L6       DC    F'6'
L7       DC    F'7'
L8       DC    F'8'
L23      DC    F'23'
L8Ø      DC    F'8Ø'
VL8      DS    F
ISPLINK  CALL  ,(,,,,,,,),VL,MF=L
UNITADDR DS    XL2
CMD      DS    CL3
CHARUADR DS    CL3
         DS    ØF
DEVCODE  DS    XL4
UCBADDR  DS    XL4
VOLSER   DS    CL6
DEVGEN   DS    CL6
UCBCMD   DS    CL6
STATUS   DS    CL7
UNITNAME DS    CL8
DEFPATHS DS    CL23
ZCMD     DS    CL8Ø
$CBJBNR  DS    XL1
$CBFL5   DS    XL1
$CBID    DS    XL1
$CBSTAT  DS    XL1
$CBCHAN  DS    XL2
$CBFLA   DS    XL1
$CBFLB   DS    XL1
$CBNXUCB DS    XL4
$CBWGT   DS    XL1
$CBNAME  DS    CL3
$CBTYP   DS    XL4
$CBFLC   DS    XL1
$CBEXTP  DS    XL3
$CBLEN   EQU   *-$CBJBNR
UCBDDS   DS    CL24
         ORG   UCBDDS
$CBVTOC  DS    XL4
$CBVOLI  DS    CL6
$CBSTAB  DS    XL1
$CBDMCT  DS    XL1
$CBSQC   DS    XL1
$CBFL4   DS    XL1
$CBUSER  DS    XL2
$CBBASE  DS    XL4
$CBNEXP  DS    XL4
DASDDDSL EQU   *-$CBVTOC
         ORG   UCBDDS
$CBFSCT  DS    XL2
$CBFSEQ  DS    XL2
```

```
         ORG   UCBDDS+12
$CBFSER  DS    CL6
$CBRDEV  DS    XL1
$CBTFL1  DS    XL1
$CBVOPT  DS    XL1
$CBXTNB  DS    XL3
TAPEDDSL EQU   *-$CBFSCT
         ORG   ,
LINKADDR DS    F
MSGADDR  DS    F
CRP      DS    F
*   END OF ISPF CONSTANTS AND VARIABLES
USERID   DC    8C' '
VERBAREA DS    CL8
WTOLIST1 WTO   'EDTMØØ8I - UCB AT XXXXXXXX UPDATED BY XXXXXXXX',       X
               ROUTCDE=(1),DESC=(6),MF=L
WTOLEN1  EQU   *-WTOLIST1
WTOWORK1 DS    CL(WTOLEN1)
         DS    ØD
SCANWORK DS    CL512
AREA4UCB DS    CL48
SCANUCB  DS    F
RTNCODE  DS    F
DBL1     DS    2D
DBL2     DS    2D
SAVEAREA DS    18F
*   THE BLDEDT PROGRAM USES THE FOLLOWING STORAGE AREAS
*   THEY MUST REMAIN CONTIGUOUS OR RESULTS ARE UNPREDICTABLE
PARMAREA DC    A(MAINTBL)
MAINTBL  DS    F
NUMDEVT  DS    F
UCBCHAIN DS    F
TBLLEN   DS    F
AUTHUADR DC    A(UCBADDR)
AUTHUCB  DC    A($CBJBNR)
SVUCBLEN DS    F
TSFFLAG  DS    XL1
TBLBUILD EQU   X'8Ø'
TBLFREE  EQU   X'4Ø'
UCBUPDAT EQU   X'2Ø'
*   END OF CONTIGUOUS STORAGE AREA REQUIREMENT
FLAG     DC    F'Ø'
AMODE31  EQU   X'8Ø'
PRE22Ø   EQU   X'4Ø'
GOTEDT   EQU   X'2Ø'
BLDERR   EQU   X'Ø8'
         CVT   DSECT=YES
         IHAPSA DSECT=YES
         IKJTSVT
         DSECT
         IEFUCBOB
         END
```

BLDEDT PROGRAM

```
BLDEDT   CSECT
*   BLDEDT RECEIVES AS INPUT IN R1 THE ADDRESS OF A PARAMETER LIST.
*   THE PARAMETER LIST HAS ONE PARAMETER, A POINTER TO A 33-BYTE
*   AREA IN EDTISPF THAT THE PROGRAMS USE TO SHARE INFORMATION.
*
*   AS DOCUMENTED BY COMMENTS IN THE EDTISPF PROGRAM, THIS 33-BYTE
*   AREA MUST REMAIN CONTIGUOUS OR RESULTS ARE UNPREDICTABLE.
*
*   AT EXIT R15 WILL BE Ø IF THE TABLES WERE BUILT SUCCESSFULLY
*   AND NON-ZERO IF THERE WAS A PROBLEM.
         PRINT NOGEN
         STM   R14,R12,12(R13)          SAVE INCOMING ENVIRONMENT
         LR    R11,R15                  SAVE MODULE BASE ADDRESS
         USING BLDEDT,R11
         USING UCBOB,R7                 SET UCB BASE REG
         ST    R13,SAVEAREA+4           SAVE SAVEAREA ADDRESS
         LA    R13,SAVEAREA             LOAD R13 WITH NEW SAVEAREA
         ST    R1,PARMS                 SAVE R1
         L     R1,Ø(,R1)
         ICM   R1,15,2(R1)              GET ADDRESS OF PARM AREA
         L     R1,Ø(,R1)
         TM    28(R1),X'4Ø'             DO WE WANT TO FREE STORAGE?
         BO    FREERTN                  YES - GO FREE THE STORAGE
         TM    28(R1),X'2Ø'             DO WE WANT TO UPDATE A UCB?
         BO    UCBUPDAT                 YES - GO UPDATE THE UCB
         L     R3,16                    GET CVT ADDRESS
         USING CVT,R3                   SET ADDRESSABILITY
         MVC   UCBLUTBL(4),X'28'(R3)    MOVE IN LOOKUP TABLE ADDRESS
         MVC   LCHNTBL(4),X'8C'(R3)     MOVE IN LOGICAL CHN TBL ADDR
         L     R6,CVTCUCB               GET UCM ADDRESS
         TM    CVTDCB,CVTMVSE           MVS/XA?
         BZ    NEWMODE1                 NO - DON'T SET TO 31 BIT
         OI    FLAG,AMODE31             SET FLAG
         L     R14,=A(X'8ØØØØØØØ'+NEWMODE1) SET TO 31 BIT ADDRESSING
*        BSM   RØ,R14                   SET MODE
         DC    X'ØBØE'                  CODE IN HEX IN CASE OF ASM 'F'
NEWMODE1 DS    ØH
         L     R3,CVTJESCT              GET JESCT ADDRESS
         DROP  R3
         TM    X'46'(R6),X'Ø8'          OS/39Ø?
         BZ    NOTOS39Ø                 NO - CHECK OTHER FLAGS
         OI    FLAG,OS39Ø               SET OS/39Ø FLAG
         B     LIKE4XX                  TREAT LIKE A V4 EDT
NOTOS39Ø TM    X'46'(R6),X'Ø4'          SP 4.X.X?
         BZ    NOT4XX                   NO - DON'T GET V4 EDT ADDRESS
LIKE4XX  L     R15,X'78'(,R3)           GET DACA ADDRESS
         L     R15,X'6Ø'(,R15)          GET EDT LATCH ADDRESS
         L     R3,X'1Ø'(,R15)           GET PRIMARY EDT ADDRESS
         LTR   R3,R3                    DID WE GET AN EDT ADDRESS?
         BNZ   GOTEDT                   YES - WE'LL MAKE 1 MORE CHECK
         L     R3,X'14'(,R15)           GET SECONDARY EDT ADDRESS?
         LTR   R3,R3                    DID WE GET AN EDT ADDRESS?
```

```
         BZ    EDTERROR                  NO - ISSUE ERROR RETURN CODE
GOTEDT   EQU   *
         CLC   Ø(3,R3),=C'EDT'           IS THIS AN EDT?
         BNE   EDTERROR                  NO - ISSUE AN ERROR AND RETURN
         MVC   LUVLEN(4),=F'52'          LENGTH IS 52 NOW
         B     EDTPTRS                   GET EDT POINTERS
NOT4XX   L     R3,X'34'(,R3)             GET EDT ADDRESS
         TM    X'46'(R6),X'Ø3'           SP 2.2.X?
         BM    NOT22Ø                    NO - SET UP A LITTLE DIFFERENT
EDTPTRS  MVC   EDTLUVSP(32),X'1C'(R3)    MOVE IN ALL EDT TABLE POINTERS
         L     R4,X'1C'(,R3)             GET LOOKUP SECTION ADDRESS
         L     R5,8(,R4)                 GET NUMBER OF DEVICE TYPES
         ST    R5,NUMDEVT                SAVE NUMBER OF DEVICE TYPES
         LA    R4,16(,R4)                POINT TO FIRST ENTRY
         ST    R4,DEVTABLE               SAVE ADDRESS
         BAL   R14,BLDTBLS               BUILD UNIT TABLES
         XR    R15,R15                   SET RETURN CODE
         B     RETURN                    GO BACK
NOT22Ø   EQU   *
         L     R15,=F'2Ø'                SET RETURN CODE
         B     END                       GO BACK
RETURN   EQU   *
         TM    FLAG,AMODE31              AMODE=31?
         BZ    AMODEØ2                   NO - DON'T SET BACK
         LA    R14,*+6                   SET BRANCH ADDRESS
*        BSM   RØ,R14
         DC    X'ØBØE'                   CODE IN HEX IN CASE OF ASM 'F'
AMODEØ2  EQU   *
         L     R1,PARMS                  GET PARM ADDRESS
         L     R1,Ø(,R1)
         ICM   R1,15,2(R1)               GET ADDRESS OF PARM AREA
         L     R1,Ø(,R1)
         MVC   Ø(4,R1),MAINTBL           SAVE MAIN TABLE ADDRESS
         MVC   4(4,R1),NUMDEVT           SAVE UNIQUE UNITNAME COUNT
         MVC   8(4,R1),UCBCHAIN          SAVE DEVICE CHAIN ADDRESS
END      EQU   *
         L     R13,SAVEAREA+4            GET OLD SAVEAREA ADDRESS
         L     R14,12(,R13)              GET RETURN ADDRESS
         LM    RØ,R12,2Ø(R13)            RELOAD RØ-R12
         BR    R14                       RETURN
UCBERROR L     R15,=F'8'                 SET RETURN CODE
         B     END                       GO BACK
GETERR   L     R15,=F'12'                SET RETURN CODE
         B     END                       GO BACK
EDTERROR L     R15,=F'16'                SET RETURN CODE
         B     END                       GO BACK
BLDTBLS  EQU   *
         STM   RØ,R15,RTN1SAVE           SAVE THE ENVIRONMENT
         L     R5,NUMDEVT                GET NUMBER OF UNIT TYPES
         XR    R4,R4                     CLEAR R4
         M     R4,=F'12'                 MUTIPLY BY 12
         GETMAIN RU,LV=(R5),SP=251       GET MAIN TABLE STORAGE
         LTR   R15,R15                   STORAGE OK?
         BNZ   GETERR                    NO - SET ERROR RETURN CODE
```

```
         ST    R1,MAINTBL               SAVE TABLE ADDRESS
         L     R4,DEVTABLE              GET DEVICE TABLE ADDRESS
         L     R5,NUMDEVT               GET NUMBER OF UNIT TYPES
         LR    R8,R1                    GET MAIN TABLE ADDRESS
         ST    R4,CURDEV                SAVE CURRENT DEVICE TYPE PTR
BLDLOOP  EQU   *
         MVC   Ø(8,R8),Ø(R4)            MOVE IN UNIT TYPE
         XC    8(4,R8),8(R8)            CLEAR CHAIN POINTER
MVSXA22X EQU   *
*   THIS CODE FINDS THE DEVICE NUMBERS FOR AN MVS/XA 2.2.X SYSTEM
*   R4 CONTAINS THE LOOKUP VALUE SECTION ADDRESS
         XR    R6,R6                    CLEAR R6
         XR    R7,R7                    CLEAR R7
         LH    R7,X'1C'(,R4)            GET GENERIC TBL ENTRY NUMBER
GENLOOP  BCTR  R7,Ø                     REDUCE BY ONE
         M     R6,=F'12'                GENERATE OFFSET VALUE
         L     R4,EDTGENSP              GET GENERIC SECTION ADDRESS
         LA    R4,12(R7,R4)             POINT TO FIRST ENTRY
         MVC   NEXTGENP(2),1Ø(R4)       GET NEXT GENERIC SECTION ENTRY
*   R4 CONTAINS THE GENERIC SECTION ADDRESS
         XR    R6,R6                    CLEAR R6
         XR    R7,R7                    CLEAR R7
         LH    R7,X'8'(,R4)             GET GRP POINTER ENTRY NUMBER
GRPLOOP  BCTR  R7,Ø                     REDUCE BY ONE
         M     R6,=F'4'                 GENERATE OFFSET VALUE
         L     R4,EDTGRPPP              GET GRP PTR SECTION ADDRESS
         LA    R4,12(R7,R4)             POINT TO FIRST ENTRY
         MVC   NEXTGRPP(2),2(R4)        GET NEXT GROUP ENTRY
*   R4 CONTAINS THE GROUP POINTER SECTION ADDRESS
         XR    R6,R6                    CLEAR R6
         XR    R7,R7                    CLEAR R7
         LH    R7,Ø(,R4)                GET GROUP SECTION ENTRY NUMBER
         BCTR  R7,Ø                     REDUCE BY ONE
         M     R6,=F'12'                GENERATE OFFSET VALUE
         L     R4,EDTGRPSP              GET GROUP SECTION ADDRESS
         LA    R4,12(R7,R4)             POINT TO FIRST ENTRY
         MVC   NEXTGRDP(2),2(R4)        GET NEXT GROUP DESC ENTRY
*   R4 CONTAIN THE GROUP SECTION ADDRESS
         XR    R6,R6                    CLEAR R6
         XR    R7,R7                    CLEAR R7
         L     R7,4(,R4)                GET GROUP SECTION ENTRY NUMBER
DEVREDO  BCTR  R7,Ø                     REDUCE BY ONE
         M     R6,=F'8'                 GENERATE OFFSET VALUE
         L     R4,EDTUCBSP              GET GROUP SECTION ADDRESS
         LA    R4,12(R7,R4)             POINT TO FIRST ENTRY
*   R4 CONTAINS THE UCB SECTION ADDRESS
         L     R7,4(,R4)                GET NEXT ADDRESS NUMBER
         GETMAIN RU,LV=48,SP=251        GET ADDRESS ENTRY STORAGE
         LTR   R15,R15                  GOT STORAGE OK?
         BNZ   GETERR                   NO - SET ERROR RETURN CODE
         XC    Ø(4,R1),Ø(R1)            CLEAR POINTER
         MVC   4(4,R1),Ø(R4)            MOVE IN DEVICE ADDRESS
         MVC   16(32,R1),=32C' '        CLEAR OUT CHANNEL AREA
```

```
          BAL   R14,ADDENTRY            ADD ENTRY TO TABLE
          XR    R6,R6                   CLEAR R6
          LTR   R7,R7                   ANY MORE?
          BNZ   DEVREDO                 YES - GO GET THEM
          XR    R6,R6                   CLEAR R6
          XR    R7,R7                   CLEAR R7
          LH    R7,NEXTGRPP             GET GRP POINTER ENTRY NUMBER
          LTR   R7,R7                   ANY MORE?
          BNZ   GRPLOOP                 YES - GO CHECK IT OUT
          XR    R6,R6                   CLEAR R6
          XR    R7,R7                   CLEAR R7
          LH    R7,NEXTGENP             GET GEN SECTION ENTRY NUMBER
          LTR   R7,R7                   ANY MORE?
          BNZ   GENLOOP                 YES - GO CHECK IT OUT
          BAL   R14,UCBSRCH             SEARCH FOR THE UCB
          L     R4,CURDEV               GET CURRENT UNIT POINTER
          L     R15,LUVLEN              GET LOOK UP VALUE ENTRY LENGTH
          LA    R4,Ø(R15,R4)            POINT TO NEXT
          ST    R4,CURDEV               SAVE IT
          LA    R8,12(,R8)              POINT TO NEXT TABLE POINTER
          BCT   R5,BLDLOOP              BUILD NEXT CHAIN
          L     R14,RTN1SAVE+56         RESTORE R14
          BR    R14                     RETURN
ADDENTRY  EQU   *
          ST    R14,R14SAVE             SAVE RETURN ADDRESS
          MVC   DBL1(4),4(R1)           MOVE IN DEVICE NUMBER
          TM    FLAG,OS39Ø              IS THIS OS/39Ø?
          BO    DEV#SET                 YES - WE'VE GOT THE RIGHT DEV#
          MVI   DBL1,C'Ø'               PAD FIRST BYTE
          MVC   DBL1+1(3),4(R1)         MOVE IN DEVICE NUMBER
DEV#SET   TR    DBL1(4),TRTABLE         TRANSLATE
          PACK  DBL2(8),DBL1(4)         PACK
          L     R2,DBL2+4               GET PACKED VALUE
          SRL   R2,4                    GET RID OF SIGN
          ST    R2,8(,R1)               SAVE VALUE
          L     R2,8(,R8)               GET ANCHOR ADDRESS
          ST    R2,SAVECRNT             SAVE IT
SORTLOOP  LTR   R2,R2                   END OF CHAIN?
          BZ    ADDITØ1                 YES - ADD TO CHAIN
          CLC   8(4,R2),8(R1)           WHERE ARE WE?
          BH    ADDITØ1                 HIGH - GO ADD TO CHAIN
          ST    R2,SAVEPREV             SAVE LAST ENTRY
          L     R2,Ø(,R2)               POINT TO NEXT ENTRY
          ST    R2,SAVECRNT             SAVE CURRENT ENTRY
          B     SORTLOOP                CHECK IT OUT
ADDITØ1   ST    R2,Ø(,R1)               SAVE NEXT POINTER
          CLC   8(4,R8),SAVECRNT        NEXT IS ANCHOR?
          BE    FIRSTØ1                 YES - CHANGE ANCHOR
          L     R3,SAVEPREV             GET PREVIOUS ENTRY ADDRESS
          ST    R1,Ø(,R3)               COMPLETE THE CHAIN
          B     CHKNEXT                 GO CHECK NEXT CHAIN
FIRSTØ1   ST    R1,8(,R8)               SAVE NEW ANCHOR ADDRESS
CHKNEXT   ST    R1,SAVEVAL              SAVE THIS VALUE
```

```
         L     R1Ø,UCBCHAIN             GET UCB CHAIN ANCHOR
         ST    R1Ø,SAVECRNT             SAVE CURRENT VALUE
UCBLP    LTR   R1Ø,R1Ø                  END OF CHAIN?
         BZ    GETAREA                  YES - GET STORAGE FOR NEW
         CLC   8(4,R1),Ø(R1Ø)           UNIT ADDRESS MATCH?
         BE    ADDUNIT                  YES - ADD IT TO CHAIN
         BL    GETAREA                  LOW - GET STORAGE FOR NEW
         ST    R1Ø,SAVEPREV             SAVE LAST ENTRY ADDRESS
         L     R1Ø,4(,R1Ø)              POINT TO NEXT ENTRY
         ST    R1Ø,SAVECRNT             SAVE CURRENT ENTRY ADDRESS
         B     UCBLP                    CHECK NEXT ONE
GETAREA  GETMAIN RU,LV=24,SP=251        GET STORAGE FOR NEW ENTRY
         LTR   R15,R15                  GET OK?
         BNZ   GETERR                   NO - SET ERROR RETURN CODE
         XC    Ø(24,R1),Ø(R1)           CLEAR AREA
         ST    R1Ø,4(,R1)               SAVE NEXT ENTRY ADDRESS
         L     R14,SAVEVAL              GET UNITNAME ENTRY ADDRESS
         MVC   Ø(4,R1),8(R14)           MOVE IN UNIT ADDRESS
         ST    R8,12(,R1)               SAVE UNITNAME ADDRESS
         ST    R14,8(,R1)               SAVE UNITNAME ENTRY ADDRESS
         CLC   UCBCHAIN(4),SAVECRNT     ANCHOR IS CURRENT?
         BE    FIRSTØ2                  YES - GO CHANGE ANCHOR
         L     R15,SAVEPREV             GET LAST ENTRY ADDRESS
         ST    R1,4(,R15)               COMPLETE THE CHAIN
         B     CHKNEXT1                 CHECK NEXT ONE
FIRSTØ2  ST    R1,UCBCHAIN              SAVE NEW ANCHOR
         B     CHKNEXT1                 CHECK NEXT ONE
ADDUNIT  EQU   *
         GETMAIN RU,LV=8,SP=251         GET STORAGE
         LTR   R15,R15                  GET OK?
         BNZ   GETERR                   NO - SET ERROR RETURN CODE
         XC    Ø(8,R1),Ø(R1)            CLEAR AREA
         L     R15,16(,R1Ø)             GET UNITNAME CHAIN ADDRESS
         LA    R14,16(,R1Ø)             SAVE ADDRESS
CHNLOOP  LTR   R15,R15                  LAST ENTRY?
         BZ    CHAINEND                 YES - WE'RE AT THE END
         LA    R14,4(,R15)              SAVE ADDRESS
         L     R15,4(,R15)              POINT TO NEXT
         B     CHNLOOP                  CHECK IT OUT
CHAINEND EQU   *
         ST    R1,Ø(,R14)               SAVE ENTRY ADDRESS
         ST    R8,Ø(,R1)                SAVE UNITNAME ADDRESS
CHKNEXT1 L     R14,R14SAVE              GET RETURN ADDRESS
         BR    R14                      RETURN
UCBSRCH  EQU   *
         ST    R14,R14SAVE              SAVE RETURN ADDRESS
         L     R2,8(,R8)                GET CHAIN START ADDRESS
         L     R3,UCBCHAIN              GET UCB CHAIN START ADDRESS
NEXTUCBØ LTR   R2,R2                    MORE TO GO?
         BZ    ENDUCB1                  NO - ALL DONE UCB SCANS
NEXTUCB1 LTR   R3,R3                    END OF UCB CHAIN?
         BZ    UCBERROR                 YES - SET ERROR RETURN CODE
         CLC   Ø(4,R3),8(R2)            DEVICE NUMBER MATCH?
```

```
         BE    CHKUADDR                  YES - CHECK IF UCB FOUND
         L     R3,4(,R3)                 GET NEXT ENTRY ADDRESS
         B     NEXTUCB1                  GO CHECK IT OUT
NEXTUCB2 EQU   *
         XC    SCANWORK(100),SCANWORK    CLEAR THE WORKAREA
         MVC   DEVN(2),10(R2)            SAVE THE DEVICE NUMBER
         XC    PINTOKEN(8),PINTOKEN      CLEAR TOKEN AREA
         MODESET KEY=ZERO,MODE=SUP       GET AUTHORIZED
         UCBLOOK DEVN=DEVN,UCBPTR=SCANUCB,DYNAMIC=YES,UCBPXPTR=PREPTR, X
               PIN,TEXT=PINMSG,PTOKEN=PINTOKEN
         ST    R15,R15SAVE               SAVE RETURN CODE
         LTR   R15,R15                   DID WE FIND A UCB?
         BNZ   NOUNPIN                   NO - WE DON'T HAVE TO UNPIN
         UCBPIN UNPIN,PTOKEN=PINTOKEN    UNPIN THE UCB
NOUNPIN  EQU   *
         MODESET MODE=PROB               GET UNAUTHORIZED
PROBMODE L     R15,R15SAVE               GET UCBSCAN RETURN CODE
         LTR   R15,R15                   DID WE FIND A UCB?
         BNZ   UCBERROR                  NO - ISSUE AN ERROR
         L     R7,SCANUCB                LOAD THE UCB ADDRESS
NEXTUCB5 ST    R7,12(,R2)                SAVE UCB ADDRESS
         ST    R7,20(,R3)                SAVE UCB ADDRESS
         L     R9,PREPTR                 GET ADDRESS OF UCB PREFIX
         LA    R9,16(,R9)                POSITION TO A POINT YOU KNOW
         LA    R1,8(,R9)                 POINT TO CHPID AREA
         LA    R10,16(,R2)               GET SAVE AREA ADDRESS
         L     R15,=F'8'                 SET LOOP COUNT
         L     R6,=F'128'                SET FIRST MASK VALUE
DEFTEST  EX    R6,MASK1TM                EXECUTE THE TM
         BZ    NEXT010                   NO - GET NEXT ONE
         BAL   R14,CNVT010               CONVERT TO READABLE
NEXT010  SRL   R6,1                      SHIFT OVER ONE BIT
         LA    R1,1(,R1)                 INCREMENT PATH AREA POINTER
         BCT   R15,DEFTEST               CHECK OUT NEXT ONE
         LA    R10,32(,R2)               GET SAVE AREA ADDRESS
         LA    R1,8(,R9)                 GET PATH AREA ADDRESS
         L     R15,=F'8'                 SET LOOP COUNT
         L     R6,=F'128'                SET FIRST MASK VALUE
ONTEST   EX    R6,MASK2TM                EXECUTE THE TM
         BZ    NEXT020                   NO - GET NEXT ONE
         BAL   R14,CNVT010               CONVERT TO READABLE
NEXT020  SRL   R6,1                      SHIFT OVER ONE BIT
         LA    R1,1(,R1)                 INCREMENT PATH AREA POINTER
         BCT   R15,ONTEST                CHECK OUT NEXT ONE
NEXTUCB3 L     R7,8(,R7)                 GET POINTER TO NEXT UCB
         L     R2,0(,R2)                 GET NEXT CHAIN ENTRY
         LTR   R2,R2                     MORE TO GO?
         BZ    ENDUCB1                   NO - ALL DONE UCB SCANS
         L     R3,4(,R3)                 GET NEXT ENTRY ADDRESS
         LTR   R7,R7                     NEXT UCB VALID?
         BZ    NEXTUCB0                  NO - GO FIND THE RIGHT ONE
         TM    FLAG,OS390                IS THIS OS/390?
         BO    NEXTUCB0                  YES - GET THE UCB INFO
```

```
         LR    R9,R7                     GET THE UCB ADDRESS
         S     R9,=F'48'     IT'S A STATIC UCB SO WE PREFIX LOCATION
         ST    R9,PREPTR                 SAVE THE PREFIX ADDRESS
         CLC   1Ø(2,R2),UCBCHAN          DEVICE NUMBER MATCH?
         BNE   NEXTUCBØ                  NO - GO FIND THE RIGHT ONE
NEXTUCB4 CLC   Ø(4,R3),8(R2)              DEVICE NUMBER MATCH?
         BE    NEXTUCB7                  YES - SAVE UCB ADDRESS
         L     R3,4(,R3)                 GET NEXT ENTRY ADDRESS
         B     NEXTUCB4                  GO BACK
CHKUADDR CLC   2Ø(4,R3),=4X'ØØ'          ANYTHING?
         BE    NEXTUCB2                  NO - GO FIND THE UCB ADDRESS
         L     R7,2Ø(,R3)                LOAD THE UCB ADDRESS
NEXTUCB8 MVC   12(4,R2),2Ø(R3)           MOVE IN THE UCB ADDRESS
         L     R1Ø,8(,R3)                GET UNITNAME ENTRY ADDRESS
         MVC   16(32,R2),16(R1Ø)         MOVE IN CHANNEL INFORMATION
         B     NEXTUCB3                  GO GET NEXT ONE
NEXTUCB7 CLC   2Ø(4,R3),=4X'ØØ'          ANYTHING?
         BE    NEXTUCB5                  NO - GO SAVE THE UCB ADDRESS
         B     NEXTUCB8                  GO GET NEXT ONE
ENDUCB1  EQU   *
         L     R14,R14SAVE               GET RETURN ADDRESS
         BR    R14                       RETURN
CNVTØ1Ø  EQU   *
         XC    DBL1(2),DBL1              CLEAR AREA
         MVC   DBL1+1(1),Ø(R1)           MOVE IN PATH ID
         UNPK  DBL2(5),DBL1(3)           UNPACK IT
         NC    DBL2(4),=4X'ØF'           CLEAR HIGH NIBBLE
         TR    DBL2(4),TRTABLE           TRANSLATE
         MVC   Ø(2,R1Ø),DBL2+2           MOVE IN READABLE VALUE
         LA    R1Ø,2(,R1Ø)               UPDATE POINTER
         BR    R14                       RETURN
FREERTN  EQU   *
         LR    R6,R1                     SAVE POINTER ADDRESSES
         L     R8,Ø(,R1)                 GET MAIN TABLE ADDRESS
         L     R5,4(,R1)                 GET MAX LOOP COUNT
         L     R2,8(,R8)                 GET CHAIN START ADDRESS
FREELOOP EQU   *
         LTR   R2,R2                     END OF CHAIN?
         BZ    NEXTCHN                   YES - GET START OF NEXT CHAIN
         L     R3,Ø(,R2)                 SAVE NEXT ADDRESS
         FREEMAIN RU,LV=48,A=(2),SP=251  FREE THE STORAGE
         LR    R2,R3                     GET NEXT ADDRESS
         B     FREELOOP                  GO FREE IT
NEXTCHN  EQU   *
         LA    R8,12(,R8)                POINT TO NEXT ENTRY
         L     R2,8(,R8)                 GET CHAIN START ADDRESS
         BCT   R5,FREELOOP               GO FREE CHAIN
         L     R1,Ø(,R6)                 GET MAIN TABLE ADDRESS
         L     R2,12(,R6)                GET TABLE LENGTH
         FREEMAIN RU,LV=(R2),A=(R1),SP=251 FREE THE STORAGE
         L     R1,8(,R6)                 GET UCB CHAIN HEADER
FREELPØ1 LTR   R1,R1                     ANYTHING?
         BZ    FREEENDØ                  NO - ALL DONE
```

```
         L     R3,4(,R1)                  SAVE NEXT POINTER
         L     R2,16(,R1)                 SAVE UNITNAME CHAIN POINTER
         FREEMAIN RU,LV=24,A=(R1),SP=251  FREE THE STORAGE
FREELPØ2 LR    R1,R2                      LOAD UNITNAME POINTER
         LTR   R1,R1                      ANYTHING?
         BZ    FREEEND1                   NO - DO NEXT UNIT ADDRESS
         L     R2,4(,R1)                  GET NEXT ENTRY POINTER
         FREEMAIN RU,LV=8,A=(R1),SP=251   FREE THE STORAGE
         B     FREELPØ2                   CHECK OUT NEXT ENTRY
FREEEND1 LR    R1,R3                      GET NEXT UNIT ADDRESS POINTER
         B     FREELPØ1                   CHECK IT OUT
FREEENDØ XR    R15,R15                    SET RETURN CODE
         B     END                        GO HOME
UCBUPDAT EQU   *
         LR    R6,R1                      SAVE POINTER ADDRESSES
         L     R7,16(,R1)                 GET ADDR OF UCB ADDRESS
         L     R7,Ø(,R7)                  GET UCB ADDRESS
         L     R5,24(,R1)                 GET LENGTH OF UCB
         L     R2,2Ø(,R1)                 GET ADDR OF UCB DATA AREA
         RACROUTE REQUEST=AUTH,ENTITY=AUTHDSN,CLASS='DATASET',         X
               ATTR=UPDATE,WORKA=SCANWORK
         LTR   R15,R15                    AUTHORIZED USER?
         BNZ   AUTHERR                    NO - ISSUE ERROR
         MODESET KEY=ZERO,MODE=SUP
         LA    R6,DBL1                    GET SAVE AREA ADDRESS
         STM   R11,R14,Ø(R6)              SAVE SOME REGS
         LR    R15,R7                     GET UCB ADDRESS
         S     R15,=F'8'                  POINT TO UCBLOCK
         LR    R11,R15                    MOVE TO R11
         SETLOCK OBTAIN,TYPE=IOSUCB,ADDR=(11),MODE=UNCOND,REGS=USE,    X
               RELATED=(UCBLOCK)
         LM    R11,R14,Ø(R6)              RESTORE REGS
         C     R15,=F'4'                  GOT THE LOCK?
         BH    LOCKERR                    NO - ISSUE MESSAGE
         LR    R14,R5                     GET THE LENGTH
         BCTR  R14,Ø                      SUBTRACT ONE
         EX    R14,UCBMVC                 MOVE IN UPDATED UCB
         STM   R11,R14,Ø(R6)              SAVE SOME REGS
         LR    R15,R7                     GET UCB ADDRESS
         S     R15,=F'8'                  POINT TO UCBLOCK
         LR    R11,R15                    MOVE TO R11
         SETLOCK RELEASE,TYPE=IOSUCB,ADDR=(11),RELATED=(UCBLOCK)
         LM    R11,R14,Ø(R6)              RESTORE REGS
         MODESET MODE=PROB,KEY=NZERO      GET BACK TO NORMAL
         L     R15,=F'Ø'                  SET RETURN CODE
         B     END                        RETURN
AUTHERR  EQU   *
         L     R15,=F'4'                  SET RETURN CODE
         B     END                        RETURN
LOCKERR  EQU   *
         L     R15,=F'8'                  SET RETURN CODE
         B     END                        RETURN
         $REQU
AUTHDSN  DC    CL44'UCB.UPDATE'
```

```
MASK1TM  TM    7(R9),Ø
MASK2TM  TM    4(R9),Ø
UCBMVC   MVC   Ø(1,R7),Ø(R2)
TRTABLE  DC    255X'8Ø'
         ORG   TRTABLE+Ø
         DC    C'Ø123456789ABCDEF'
         ORG   TRTABLE+193
         DC    X'ØAØBØCØDØEØF'
         ORG   TRTABLE+24Ø
         DC    X'ØØØ1Ø2Ø3Ø4Ø5Ø6Ø7Ø8Ø9'
         ORG   ,
LUVLEN   DC    F'32'                    LENGTH OF 32 FOR XA 2.2.Ø
DBL1     DS    D
DBL2     DS    D
SAVEVAL  DS    F
UCBCHAIN DC    F'Ø'                     INIT TO ZERO
PARMS    DS    F
R14SAVE  DS    F
SAVEAREA DS    18F
RTN1SAVE DS    16F
NUMDEVT  DS    F
DEVTABLE DS    F
CURDEV   DS    F
NUMGRPS  DS    F
NUMGENS  DS    F
SAVECRNT DS    F
SAVEPREV DS    F
CURNTGEN DS    F
CURNTGRP DS    F
MAINTBL  DS    F
UCBLUTBL DS    F
LCHNTBL  DS    F
EDTLUVSP DS    F
EDTGENSP DS    F
EDTGRPSP DS    F
EDTUCBSP DS    F
EDTMSKTP DS    F
EDTGRPPP DS    F
EDTPREFP DS    F
EDTTAPEP DS    F
NEXTGENP DS    H
NEXTGRPP DS    H
NEXTGRDP DS    H
FLAG     DC    F'Ø'
DEVN     DS    D
SCANWORK DS    CL512
R15SAVE  DS    F
SCANUCB  DS    F
PREPTR   DS    F
PINTOKEN DS    D
PINMSG   DC    CL58'UCB IS PINNED BY ISPF BLDEDTCB PROGRAM'
AMODE31  EQU   X'8Ø'
PRE22Ø   EQU   X'4Ø'
```

```
OS390    EQU   X'01'
         CVT   DSECT=YES
         IHAPSA DSECT=YES
         DSECT
         IEFUCBOB PREFIX=YES
         IOSDUPFX LIST=YES
         END
```

Jim Lautner
MVS Software Analyst (Canada)

Year 2000 testing facilities

INTRODUCTION

The article *Year 2000 testing*, in *MVS Update* Issue 105 (June 1995), shows a program of mine which was designed to facilitate Y2K testing by front-ending SVC 11 and dynamically changing the dates that jobs used. Since this was originally developed, the Y2K effort at our site has increased considerably, and the program has been massively enhanced to address the additional requirements of the extended user-base. This enhancement has reached the point where I felt it might be worth re-supplying the package for other users.

TACHYONS – FUNCTIONALITY

The program offers the following functionality:

1 Capability to control the date for up to 20 different job prefixes. Each job prefix can have a unique date, and the job prefix can be provided in a wildcard format. Hence a suite of jobs all beginning PROD can be given a group date, as can all jobs with (say) a P in the first character and a D as the fourth by specifying a name of P##D as the prefix. (See panel ADDREXXH for further details on wildcarding.)

2 ISPF dialog to provide a simple means of defining jobs, and for stopping the system.

3 Batch-based bulk set-up facility (ie the ability to provide a library member containing all the jobs and dates that need to be added to

save keying when the system is stopped and re-started). The set-up function can also be invoked through the ISPF dialog to allow various member set-ups to occur.

4 Self detection. Because the system relocates SVCs, it cannot afford to be run twice. As a result the system knows if has been activated and prevents this condition.

5 Emergency TSO command to repair damage to SVCTABLE should the control address space be deleted.

INSTALLATION GUIDELINES

In order to complete the installation it will be necessary to include code from two previous editions of *MVS Update*. These are:

- REXWTO as documented in the article *Timed Job Submission* from *MVS Update* Issue 142 July 1998 (you may also require the dynamic APF on and off SVCs from this article).
- WTOLIST (along with the macros) from the article *Display WTORs in TSO* from *MVS Update* Issue 143 August 1998.

The following article supplies all the panels, REXX, and jobs necessary for running TACHYONS along with the replacement SVC, the SVC loader, the special REXX function to analyse the current date situation, and the SVC reset code. Only the SVC loader (the actual TACHYONS program) requires any special linkage, and it needs to be linked AC(1) and placed in an APF library.

To invoke the system once the code has been installed, all that is required is a very simple job (see TIMESTAR below). This loads the replacement SVC11 and moves the existing SVC11 to an empty entry in your SVC table (this entry is your choice – just change the SVCFREE equate in TACHYONS, TACHRES, and SVC11SVC to the number required prior to assembly). This job can be submitted by anyone with access to the load library and it does not matter if it gets submitted more than once because TACHYONS will ensure that only one user will actually be able to start it. There is one small caveat regarding this job in that the user must be able to run jobs with TIME=1440 coded. This is because TACHYONS drops into an immediate wait after the SVC is loaded and you do not want it

abending 522 and leaving a rogue intercept active. Should this happen by mistake, or should the job be incorrectly cancelled, see the TSO command TACHRES below. This will allow the SVC table to be put back to normal.

Once the system is initiated, a WTOR will be displayed on the console TACHYONS: ENTER COMMAND. It is then possible to add jobs into TACHYONS control. Now, although this operator interface can be used independently, it is unlikely it would be used because the ISPF dialog is much easier to use. Hence, issue the command TSO TACHREX1 from your TSO terminal. You will then be presented with a screen that looks something like this (assuming some jobs have been already added, of course):

```
                  TACHYONS is watching the following:       Row 1 to 2 of 2
Command ===>                                               Scroll ===> PAGE

Note TACHYONS is running with a jobname of Y2KINTER

Available commands are ADD DEL SETUP and TERM (press PF1 for details)

Jobname     Length        Date
JOB1        4             2002063F
JOB222      6             1999100F
JOB###C     7             1999001F
******************************** Bottom of data ****************************
```

The above screen shows that all jobs beginning with JOB1 in their name will have their date set to day 63 of 2002! Jobs beginning JOB222 will be set to day 100 of 1999, and jobs beginning JOB and having a C in the seventh character and being at least seven characters long will have their date set to 1 January 1999. Note that although the dates are shown in Julian format, entry of the dates is in a more standard DD/MM/YYYY format with all necessary validation. Jobs are added and deleted using the ADD and DEL commands, and can be entered in bulk using the SETUP command. This latter command enables the user to specify a PDS and to select a set-up member from the directory list. SETUP members consist of a job and date per card line (separated by at least one blank), and comments can be included if the first character on a line is a '*'. TERM is used to shut down the system cleanly (SVC11 is restored to normal processing). All the screens shown have full help panels and these should fill in any gaps in the detail of how to use the system.

The following is a list of the Assembler routines followed by the actual code. Please review the comments at the front of each module for instructions as to any special linkage considerations, or assembly information.

- SVC11LOD – TACHYONS loader.
- SVC11SVC – TACHYONS SVC11.
- TACHRES – TSO for SVC table rebuilding in the case of error.
- SVC11REX – REXX for retrieving TACHYONS job information.

```
                        SVC11LOD - THE TACHYONS LOADER
*****************************************************************
*
* VERSION 2 OF TACHYONS
*
* IN THIS VERSION TACHYONS CAN FRONT END SVC11 FOR PART OR WHOLE
* JOBNAMES, AND CAN PROVIDE AN INDIVIDUAL DATE FOR A JOB
*
* TO OPERATE THE FOLLOWING COMMANDS ARE AVAILABLE
* ADD=LJJJJJJJJYYYYDDD
*      WHERE L IS THE LENGTH OF THE JOBNAME COMPARATOR
*            J IS THE JOBNAME
*            YYYYDDD IS THE DATE
* DEL=JJJJJJJ IS THE JOB TO REMOVE
* TERM IS THE COMMAND TO REMOVE THE FRONT END.
*
* NOTE: THIS PROGRAM MUST BE LINK EDITED AC(1) AND RUN FROM AN
*       APF AUTHORIZED LIBRARY.
*
* ===== CHANGE THE EQU SVCFREE TO MATCH THE EMPTY SLOT ====
* ===== IN THE SVCTABLE!!!!                            ====
*****************************************************************
         TITLE 'TACHYONS - INITIALIZATION'
         PRINT NOGEN
TACHYONS AMODE 31
TACHYONS CSECT
         BAKR 14,Ø
SVCTIME  EQU 11
SVCFREE  EQU 24Ø
         REXREGS
*
* LOCATE SVC TABLE AND POINT
*
         LR R12,R15
         USING TACHYONS,12
         LA    R1,Ø                  POINT TO PSA
         USING PSA,R1                MAP PSA
```

```
         L     R1,FLCCVT             POINT TO CVT
         USING CVT,R1                MAP CVT
         L     R1,CVTABEND           POINT TO SCVT
         USING SCVTSECT,R1           MAP SCVT
         L     R8,SCVTSVCT           POINT TO SVC TABLE
*
         TITLE 'LOAD CSA CODE'
         MODESET MODE=SUP,KEY=ZERO   GET INTO SUPERVISOR STATE
         LOAD EP=SVC11SVC,GLOBAL=(YES,F),EOM=YES,LOADPT=LOADPT
         LR    R2,RØ       R2 CONTAINS ADDRESS OF NEW SVC.
         LA    R3,(SVCTIME*8)(,R8)   POINT TO SVC 11 POSITION IN TABLE
         L     R4,Ø(R3)              GET R11 ADDRESS
         LR    R9,R4                 PUT IT IN R9 FOR DEFENSE CODE
         S     R9,=F'16'             GO BACK TO FIND THE LITERAL
         CLC   Ø(8,R9),=C'YEAR2ØØØ'  * IS SVC 11 ALREADY SET?
         BNE   SETUP                 * NO, SO CHANGE IT
         WTO 'TACHYONS: SVC 11 INTERCEPT ALREADY ACTIVE'
         WTO 'TACHYONS: PROGRAM TERMINATING'
         PR
SETUP    DS ØH
*
*** ASSUMING THE SVC IS TO BE FRONT-ENDED, PUT THE JOBNAME OF THE
*** CONTROLLING TACHYONS INTO THE SVC AREA
*** THE LOCATION FOR THIS WILL BE DERIVED FROM 4 BEFORE THE CSECT
*
         L     R5,LOADPT           ADDRESS JOBNAME FIELD
         L     R5,12(R5)
*
*** NOW GET MYJOBNAME
*
         EXTRACT RETAD,,FIELDS=TIOT
         L     R8,RETAD            GET TIOT ADDRESS
         MVC   Ø(8,5),Ø(8)         AND STORE THE JOBNAME
         MVC   SVC11(8),Ø(R3)       SAVE CONTENTS OF SVC TABLE ENTRY
         LA    R8,MAJENQ             *
         LA    R9,MINENQ             * FORCE SERIALISATION
         ENQ   ((8),(9),E,8,SYSTEM)  *
         SVCUPDTE SVCFREE,REPLACE,TYPE=4,EP=(4) * RELOCATE 11
         SVCUPDTE SVCTIME,REPLACE,TYPE=4,EP=(2) * PLACE NEW 11
         LA    R8,MAJENQ
         LA    R9,MINENQ             FORCE SERIALIZATION
         DEQ   ((8),(9),8,SYSTEM)    DEQUEUE ROUTINE
         WTO   'TACHYONS: TIME UPDATED '
ISSCOM   DS  ØH
         MVC REPLY,BLANKS            * ENSURE REPLY FIELD CLEAR
         XC  REPECB,REPECB
         WTOR 'TACHYONS: ENTER COMMAND',REPLY,3Ø,REPECB
         WAIT  ECB=REPECB
REPROUT  DS ØH
* ADDRESS THE JOBTABLE WITH REGISTER 5
* AND KEEP TABLE SIZE IN REGISTER 6
         LA R6,2Ø                    * EQUIVALENT TO SVC ENTRY
```

```
         L R5,LOADPT
         L R5,8(R5) ADDRESS JOBTABLE
         OC  REPLY,BLANKS            * ENSURE COMMAND BLANK FILLED
         CLC REPLY(4),TERM           * WAS TERMINATE ENTERED
         BE  ENDPROG
         CLC REPLY(4),ADDCOM         * WAS IT AN ADD COMMAND?
         BE  DOADD                   *
         CLC REPLY(4),DELCOM         * WAS IT A DELETE?
         BE  DODEL                   *
         WTO 'TACHYONS: UNKNOWN COMMAND PLEASE TRY AGAIN'
         B  ISSCOM
DOADD    DS ØH
         CLC Ø(8,R5),BLANKS          * IS THE ENTRY USED?
         BE  DOADD1                  * NO SO PLACE NEW PROGRAM NAME
         LA  R5,13(,R5)              * INCREMENT TABLE SEARCH
         BCT R6,DOADD                * KEEP LOOKING
DOADD1   DS ØH
         CLI Ø(R5),C'%'              * IS IT TABLE END?
         BNE DOTIME                  * NO ITS A SPARE ENTRY
*                                    * SO CHECK TIME NEXT
         WTO 'TACHYONS: FULL JOB TABLE, ISSUE DELETE FIRST'
         B ISSCOM
DOTIME DS ØH
* NEED TO BE SURE OF VALID DATE
         MVC CHARTIME,REPLY+13       * STORE TIME FOR PACKING.
         XC  NUMFIELD,NUMFIELD      * ZERO NUMFIELD
         MVZ NUMFIELD,CHARTIME      * CHECK FOR HIGH NIBBLE F
         CLC NUMFIELD,ZEROS         * IF NOT ZERO THEN VALUE INCORRECT
         BNE NOTADATE
*
* CAN'T TYPE IN X'FA' TO X'FF' VALUES THEREFORE NUMERIC
*
         PACK PDATE,CHARTIME        *
         CLI PDATE,X'2Ø'            * SUPPORT 2ØØØ DATES
         BH  NOTADATE
         CLI PDATE,X'19'            * SUPPORT 19ØØ DATES
         BL  NOTADATE
         CP  PDATE+2(2),=PL2'ØØØ'   * CHECK FOR ZERO DATE
         BE  NOTADATE
         CP  PDATE+2(2),=PL2'366'   * AND UP TO DAY 366
         BH  NOTADATE
         CLI REPLY+13,C'2'          * YEAR 2ØØØ REQUIRED?
         BNE OLDTIME                * NO SO SET Ø CENTURY
         MVI PDATE,X'Ø1'            * SET CENTURY INDICATOR
         B   UPDTIME
OLDTIME  DS ØH
         MVI PDATE,X'ØØ'            *
UPDTIME  DS ØH
         MVC 9(4,R5),PDATE          * TRANSFER DATE TO SVC ROUTINE
         NI REPLY+4,X'ØF'           * SWITCH OFF TOP NIBBLE
         MVC Ø(9,R5),REPLY+4        * AND PUT IN LENGTH ETC.
         B ISSCOM
```

```
NOTADATE DS  ØH
         WTO 'TACHYONS: INVALID DATE SUPPLIED'
         B ISSCOM
DODEL    DS ØH
         CLC 1(8,R5),REPLY+4         * PROGRAM FOUND?
         BE DODEL1                   * YES SO REMOVE
         LA R5,13(,R5)
         BCT R6,DODEL
         WTO 'TACHYONS: PROGRAM NOT IN TABLE'
         B ISSCOM
DODEL1   DS ØH
         CLI Ø(R5),C'%'              * MAKE SURE NOT MESSING WITH
         BNE DODEL2                  * TABLE END
         WTO 'TACHYONS: MUST NOT DELETE TABLE END MARKER'
         B ISSCOM
DODEL2   DS ØH
         MVC Ø(13,R5),BLANKS
         B ISSCOM
ENDPROG  DS ØH
         LA    R8,MAJENQ             *
         LA    R9,MINENQ             * FORCE SERIALIZATION
         ENQ   ((8),(9),E,8,SYSTEM)  *
         L     R5,SVC11              RESET R5
         SVCUPDTE SVCTIME,REPLACE,TYPE=4,EP=(5)
         LA    R8,MAJENQ
         LA    R9,MINENQ              FORCE SERIALIZATION
         DEQ   ((8),(9),8,SYSTEM)     DEQUEUE ROUTINE
         PR
SVC11    DC    D'Ø'                   SVC 11 ADDRESS
RETAD    DC    F'Ø'
REPECB   DC    F'Ø'
REPLY    DS    CL3Ø
MAJENQ   DC    CL8'SYSZSVC'
MINENQ   DC    CL8'TABLE'
LOADPT   DS    F
CHARTIME DS    CL7
NUMFIELD DS    XL7
PDATE    DS    PL4
BLANKS   DC    CL3Ø' '
ZEROS    DC    1ØC'Ø'
TERM     DC    C'TERM'
ADDCOM   DC    C'ADD='
DELCOM   DC    C'DEL='
         LTORG
         PRINT NOGEN
         IHAPSA
         CVT    DSECT=YES
         IHASCVT
         END
```

```
*************************************************************
* VERSION 2 OF TACHYONS SVC MODULE
* IN THIS VERSION THERE ARE 2Ø POTENTIAL JOB ENTRIES, EACH
* WITH AN ASSOCIATED DATE. NOT ONLY THAT BUT THERE IS A LENGTH
* COMPARISON FIELD SO THAT I KNOW HOW MUCH OF THE JOBNAME COUNTS
* WHEN CARRYING OUT COMPARISONS.
* NOTE THAT THE # CHARACTER COUNTS AS A WILDCARD AND IS SKIPPED
* IN THE COMPARISON TESTS
*
* NOTE: ENSURE THAT THE SVCFREE EQUATE MATCHES THE ENTRY CHOSEN
*       FOR THE SVC11 RELOCATE WITHIN THE SVC TABLE.
*************************************************************
SVC11SVC AMODE 31
SVCFREE  EQU 24Ø           * THIS IS THE SVC TO WHERE TIME HAS
*                          * BEEN SHIFTED.
ID       DC CL8'YEAR2ØØØ'
POBTABLE DC AL4(JOBTAB)
MYJOB    DC AL4(MYNAME)
SVC11SVC CSECT
         USING *,6
         USING TCB,4
         LR 9,14
         SVC SVCFREE         * SHOULD BE OK FOR TIME
         L 5,TCBTIO
         LA 1Ø,JOBTAB        * ADDRESS JOBTABLE
         USING TABMAP,1Ø
         LA 11,TABSIZE       * CREATE A COUNTER
*
** NOW LETS LOOK TO SEE IF THE DATE SHOULD BE CHANGED
*
JOBCHK   DS ØH
         CLI JOBLEN,C' '     * IF NO JOB ASSIGNED
         BE  TRY_NEXT        * THEN SKIP ONWARDS.
         XR  7,7             * CLEAR LENGTH REGISTER
         ICM 7,B'ØØØ1',JOBLEN
*
*** NOW LOOP ALONG THE JOB NAME TO SEE IF THIS IS OK.
*** ANY '#' CHARACTERS COUNT AS A WILD CARD AND ARE NOT COMPARED
*
NAME_LOOP DS ØH
         LA  8,JOBNAME(7)    * POINT TO CHARACTER
         LA  2,Ø(7,5)        * AND IN THE CURRENT JOB
         CLI Ø(2),C' '       * IF THE INCOMING JOBNAME ISN'T LONG ENOUGH
         BE  TRY_NEXT        * THEN IT SHOULDN'T BE TWIDDLED.
         CLI Ø(8),C'#'       * WILD CARD?
         BE  GO_NEXT         * YES SO SKIP CHECK
         CLC Ø(1,2),Ø(8)     * CHARACTER MATCH?
         BNE TRY_NEXT        * NO SO TRY NEXT ENTRY
GO_NEXT  DS ØH
         BCT 7,NAME_LOOP     * WORK BACKWARDS
*** JUST NEED TO CHECK THE FIRST CHARACTER AS WELL
```

```
         CLI JOBNAME,C'#'    * IF ITS A WILDCARD, FORGET IT
         BE  TIMEALT
         CLC JOBNAME(1),Ø(5) * CHECK FIRST CHARACTER
         BE  TIMEALT
TRY_NEXT DS ØH               * NO MATCH SO FAR. KEEP LOOKING
         LA  1Ø,13(,1Ø)      * NEXT ENTRY
         BCT 11,JOBCHK
         B GOOUT
TIMEALT  DS ØH
         L   1,JOBDATE
GOOUT DS ØH
         BR 9
         LTORG
         DS ØD
MYNAME   DC 8C' '
JOBTAB   DS ØF
TABSIZE  EQU 2Ø
         DC  (13*TABSIZE)C' '
         DC  13C'%'          * END OF TABLE MARKER TO ENSURE THAT
*                            * WE DO NOT RUN OFF THE END SHOULD
*                            * THERE BE AN ERROR IN TABLE SIZES.
TABMAP   DSECT
JOBLEN   DS X
JOBNAME  DS CL8
JOBDATE  DS XL4
         IKJTCB
         END
```

TACHRES – THE EMERGENCY TSO REPAIR COMMAND

```
                TITLE 'TACHRES - INITIALIZATION'
*********************************************************************
*
* EMERGENCY RESET COMMAND FOR TACHYONS TO BE USED UNDER TSO.
* ISSUE TSO TACHRES TO CLEAR TACHYONS FROM THE SYSTEM. IF
* TACHYONS NOT ACTIVE, A MESSAGE WILL BE ISSUED AND NO CHANGES
* WILL TAKE PLACE.
*
* === SET SVCAUTH TO YOUR APF ON SVC ===
* === SET SVCDAUTH TO YOUR APF OFF SVC ===
* === SET SVCFREE TO THE POSITION IN THE SVC TABLE THAT 11 WENT TO ==
*
*********************************************************************
         PRINT NOGEN
TACHRES  AMODE 31
TACHRES  CSECT
         BAKR 14,Ø
SVCAUTH  EQU 235
SVCDAUTH EQU 236
SVCFREE  EQU 24Ø
         REXREGS
*
```

```
* LOCATE SVC TABLE AND POINT
*
         LR R12,R15
         USING TACHRES,12
         LA    R1,Ø                  POINT TO PSA
         USING PSA,R1                MAP PSA
         L     R1,FLCCVT             POINT TO CVT
         USING CVT,R1                MAP CVT
         L     R1,CVTABEND           POINT TO SCVT
         USING SCVTSECT,R1           MAP SCVT
         L     R8,SCVTSVCT           POINT TO SVC TABLE
         TITLE 'LOAD CSA CODE'
         SVC SVCAUTH
         MODESET MODE=SUP,KEY=ZERO   GET INTO SUPERVISOR STATE
         LA    R3,(SVCFREE*8)(,R8)   POINT TO SVC 24Ø POSITION IN TABLE
         LA    R5,(11*8)(,R8)        POINT TO SVC 11 POSITION IN TABLE
         L     R4,Ø(R3)              GET R11 ADDRESS
         L     R5,Ø(R5)              * CHECK IF ALREADY INTERCEPTED
         S     R5,=F'16'             * BY LOOKING FOR A START LITERAL
         CLC   Ø(8,R5),=C'YEAR2ØØØ'  * INTERCEPT THERE?
         BNE   ENDIT                 * NO SO LEAVE AS IS.
         LA    R8,MAJENQ             *
         LA    R9,MINENQ             * FORCE SERIALIZATION
         ENQ   ((8),(9),E,8,SYSTEM)  *
         SVCUPDTE 11,REPLACE,TYPE=4,EP=(4) * RELOCATE 11
         DEQ   ((8),(9),8,SYSTEM)    DEQUEUE ROUTINE
         SVC SVCDAUTH
         TPUT MESS,3Ø
         PR
ENDIT    DS ØH
         TPUT MESS1,3Ø
         PR
MESS     DC    CL3Ø'SVC RESET'
MESS1    DC    CL3Ø'SVC ALREADY RESET'
MAJENQ   DC    CL8'SYSZSVC'
MINENQ   DC    CL8'TABLE'
         LTORG
         PRINT NOGEN
         IHAPSA
         CVT    DSECT=YES
         IHASCVT
         END
```

SVC11REX – THE TACHYONS REXX DATA RETRIEVER

```
************************************************************************
* SVC11REX: A REXX FUNCTION TO DISPLAY USERS IN TACHYONS
*
* USAGE: CALL SVC11REX
*
```

```
* NOTE:  SVC11REX WILL RETURN THE FOLLOWING INFORMATION:
*        JOB_STATUS ........... ACTIVE IF TACHYONS RUNNING. INACTIVE IF
*                               NOT.
*        TACHYONS_JOBNAME ..... JOB NAME TACHYONS RUNNING UNDER.
*        JOB_LENGTH.X ......... LENGTH OF JOB NAME IF ARRAY ENTRY.
*        JOB_NAME.X ........... JOB NAME PREFIX BEING COVERED.
*        JOB_DATE.X ........... DATE JOB BEING SET TO.
*********************************************************************
SVC11REX TITLE 'REXX FUNCTION TO RETRIEVE TACHYONS INFO'
SVC11REX AMODE 31
SVC11REX RMODE ANY
SVC11REX CSECT
         REXREGS
         PRINT GEN
         BAKR 14,Ø
         LR   12,15
         USING SVC11REX,12
         PRINT GEN
         LR    R1Ø,RØ                  *R1Ø -> A(ENVIRONMENT BLOCK)
         USING ENVBLOCK,R1Ø
         L     R9,ENVBLOCK_IRXEXTE     *R9 -> A(EXTERNAL EP TABLE)
         USING IRXEXTE,R9
* GET A WORK AREA FOR REXX OUTPUT
* MAP WITH R2 ... NEED TO DO THIS BEFORE ANY ROUTING TO POSSIBLE
* REXX VARIABLE OUTPUT (EG ROUTINE ABENDØØ1)
         STORAGE OBTAIN,LENGTH=COMSLEN,ADDR=(2)
         USING WORKAREA,2
*
* PREPARE THE REXX AREA FOR USE
*
         XC  COMS(COMSLEN),COMS * SET TO LOW VALUES
         LA  15,COMID
         ST  15,COMS
         LA  15,COMDUMMY
         ST  15,COMS+4
         ST  15,COMS+8
         LA  15,COMSHVB
         ST  15,COMS+12
         LA  15,COMRET
         ST  15,COMS+16
         OI  COMS+16,X'8Ø'
         MVC COMID,=C'IRXEXCOM'
         LA    R7,Ø                  POINT TO PSA
         USING PSA,R7                MAP PSA
         L     R7,FLCCVT             POINT TO CVT
         USING CVT,R7                MAP CVT
         L     R7,CVTABEND           POINT TO SCVT
         USING SCVTSECT,R7           MAP SCVT
         L     R7,SCVTSVCT           POINT TO SVC TABLE
         LA    R7,(11*8)(,R7)        POINT TO SVC 11 POSITION IN TABLE
         L     R7,Ø(R7)              GET R11 ADDRESS
         LR    R4,R7
```

```
         S     R4,=F'16'             CHECK TACHYONS ACTIVE
         CLC   Ø(8,R4),=C'YEAR2ØØØ'
         BE    ITSOK
         SHOW  INACTIVE,JOB_STATUS
         B RETURN1
ITSOK    DS ØH
         SHOW  ACTIVE,JOB_STATUS
         S     R7,=F'4'              GET TACHYONS JOBNAME
         L     R8,Ø(R7)              BY PICKING UP ITS ADDRESS
         SHOW Ø(R8),TACHYONS_JOBNAME,LEN=8
         S     R7,=F'4'              GET JOBTABLE ADDRESS
         L     R7,Ø(R7)
         USING JOBTAB,R7             MAP JOB TABLE
*
*** R7 NOW POINTS TO START OF JOBTABLE ADDRESS
***
         LA R8,2Ø * MAX NUMBER OF JOBS
LOOPTIME DS ØH
         SHOWARAY JOBLEN,JOB_LENGTH
         SHOWARAY JOBNAME,JOB_NAME
         SHOWARAY JOBDATE,JOB_DATE
         LA R7,13(,R7) POINT TO NEXT ENTRY
         BCT R8,LOOPTIME
RETURN1  DS ØH
         STORAGE RELEASE,LENGTH=COMSLEN,ADDR=(2)
         PR
ABENDØØ1 DS    ØH
         ABEND 1
*************************************************************************
***      WORKING STORAGE ETC                                          ***
*************************************************************************
         TITLE  'WORKING STORAGE / DSECTS'
ACTIVE   DC CL8'ACTIVE'
INACTIVE DC CL8'INACTIVE'
         LTORG
JOBTAB   DSECT
JOBLEN   DS C
JOBNAME  DS CL8
JOBDATE  DS CL4
WORKAREA DSECT
*      IRXEXCOM PARAMETER AREA
         DS  ØD
COMS     DS  5AL4
COMID    DS  CL8
COMDUMMY DS  AL4              * NOT USED
COMSHVB  DS  (SHVBLEN)X       * IRXEXCOM SHVBLOCK (LENGTH FROM DSECT)
COMRET   DS  AL4              * IRXECOM RC
         DS  ØD
COMSLEN  EQU *-COMS
         IHAPSA
         CVT DSECT=YES
         IHASCVT
```

```
         IARRCE
         IRXEFPL
         IRXARGTB
         IRXEVALB
         IRXENVB
         IRXEXTE
         IRXSHVB
         END
```

TACHREX1 – THE TACHYONS INVOKING REXX

```
/* REXX */
/* */
/* Prepare a table for display purposes */
/* */
ADDRESS ISPEXEC
'LIBDEF ISPPLIB DATASET ID(your.tachyons.panel.library)' /* CHANGE!!! */
looper:
'TBCREATE SVC11J NAMES(JOBNAME JOBLEN JOBDATE) NOWRITE REPLACE'
/* */
/* Call Assembler support routine to obtain relevant information */
/* about the jobs being front ended                              */
/* */
CALL SVC11REX
/* */
IF job_status='INACTIVE' THEN DO
   zedsmsg='TACHYONS not active'
   zedlmsg='Issue TIMESTAR to start TACHYONS'
   ADDRESS ISPEXEC 'SETMSG MSG(ISRZØØ1)'
   EXIT
END
tachjob=tachyons_jobname /* save tachjobname for panel */
/* */
/* Now loop around to list all the SVC11Jtems in table form */
/* */
DO x=1 TO 2Ø
IF job_name.x=' ' THEN ITERATE
   jobname=job_name.x
   joblen=RIGHT(C2X(job_length.x),1)+1
   jobdate=C2X(job_date.x)
   IF left(jobdate,2)='Ø1' THEN DO
      jobdate=OVERLAY('2Ø',jobdate,1)
      END
      ELSE jobdate=OVERLAY('19',jobdate,1)
      'TBADD SVC11J'
END
'TBTOP SVC11J'
'TBDISPL SVC11J PANEL(TACHPAN1)'
IF reply='END' THEN EXIT
IF ZCMD='ADD' THEN 'SELECT CMD(%ADDREXX)'
ELSE IF ZCMD='DEL' THEN 'SELECT CMD(%DELREXX)'
```

```
ELSE IF ZCMD='TERM' THEN 'SELECT CMD(%TERMREXX)'
ELSE IF ZCMD='SETUP' THEN 'SELECT CMD(%SETUPREX)'
ELSE IF ZCMD/='' THEN CALL mess_dets
/* */
/* Meanwhile back at the table display! */
/* */
SIGNAL looper
/* */
mess_dets:
zedsmsg=zcmd 'unknown command'
zedlmsg='Commands are ADD TERM SETUP DEL'
ADDRESS ISPEXEC 'SETMSG MSG(ISRZØØ1)'
RETURN
```

TACHPAN1 – THE TACHYONS PRIMARY PANEL

```
)Attr Default(%+_)
   ! type(output) intens(high) caps(on ) just(left )
)Body  Expand(//)
/ /%   TACHYONS is watching the following: / /
%Command ===>_zcmd                                / /%Scroll ===>_amt +
+
+Note%TACHYONS+is running with a jobname of!tachjob
+
+ Available commands are%ADD%DEL%SETUP+and%TERM+(press PF1 for details)
+
Jobname     Length        Date
)Model
!z          !z            !z
)Init
   .Help = tachpanh               /* insert name of tutorial panel  */
   .ZVARS = '(jobname joblen jobdate)'
   &amt = PAGE
)PROC
&REPLY = .RESP
)End
```

TACHPANH – HELP PANEL FOR PRIMARY PANEL

```
)ATTR
' TYPE(PT)                        /* panel title line            */
? TYPE(PIN)                       /* panel instruction line      */
# TYPE(NT)                        /* normal text attribute       */
} TYPE(ET)                        /* emphasized text attribute   */
! TYPE(DT)                        /* description text            */
† AREA(SCRL)                      /* scrollable area attribute   */
)BODY
'———————— Help Panel For TACHYONS ————————————
%(T)ake (A)nd (C)ontrol (H)ow (Y)ears (O)riginate (N)ew (S)ystem
+
```

```
+This dialog displays all the jobs currently being monitored by TACHYONS
+
+The first line identifies the job name which TACHYONS has been started
+with, should it become necessary to talk to TACHYONS directly, rather
+then through the dialog front end.
+
+Valid commands on this screen are:
+=============================================================================
ƒpnarea                                                                     †
ƒ                                                                           †
ƒ                                                                           †
ƒ                                                                           †
ƒ                                                                           †
ƒ                                                                           †
ƒ                                                                           †
ƒ                                                                           †
ƒ                                                                           †
+=============================================================================
+
%Use ENTER to scroll downwards through the available data.
)AREA pnarea
#
}ADD+    This will allow a new job to be added to TACHYONS control.
+        Note that this name will be a prefix. ie TACHYONS will
+        track all jobs beginning with the string ADDed.
#
}DEL+    This will allow a job to be deleted from TACHYONS's control.
#
}SETUP+  This presents a panel to permit the user to specify a dataset
+        containing a member which has a list of preset job prefix and
+        dates to be used.
#
}TERM+   This shuts TACHYONS down.
+
+        Note that all commands will present an intermediary panel to
+        further identify what to do to use the command.
)PROC
.help=isp00004
)END
```

ADDREXX - REXX FOR THE ADD COMMAND

```
/* REXX */
/* */
/* ADDREXX: drive additions of new jobs into TACHYONS control */
/* first thing to do is build a table of days in months for   */
/* converting dd/mm/yyyy into the yyyddd format for the       */
/* commands.                                                  */
/* */
month.1=0                    /* jan */
month.2=31                   /* feb */
month.3=59                   /* mar */
month.4=90                   /* apr */
```

```
month.5=120                    /* may */
month.6=151                    /* jun */
month.7=181                    /* jul */
month.8=212                    /* aug */
month.9=243                    /* sep */
month.10=273                   /* oct */
month.11=304                   /* nov */
month.12=334                   /* dec */
/* */
mon.1=31                     /* jan */
mon.2=28                     /* feb */
mon.3=31                     /* mar */
mon.4=30                     /* apr */
mon.5=31                     /* may */
mon.6=30                     /* jun */
mon.7=31                     /* jul */
mon.8=31                     /* aug */
mon.9=30                     /* sep */
mon.10=31                    /* oct */
mon.11=30                    /* nov */
mon.12=31                    /* dec */
/* */
/* prepare a suitable add command for tachyons */
/* */
CALL SVC11REX
IF job_status='INACTIVE' THEN DO /* check if tachyons operating */
   CALL mess_details
   EXIT
   END
retry:
/* now actual day numbers */
ADDRESS ISPEXEC
'ADDPOP ROW(1) COLUMN(9)'
'DISPLAY PANEL(ADDREXXP)'
'REMPOP'
IF REPLY='END' THEN DO
   zedsmsg='Add aborted'
   zedlmsg='No action taken'
   ADDRESS ISPEXEC 'SETMSG MSG(ISRZ001)'
   EXIT
   END
jobl=LENGTH(name)-1
name=LEFT(name,8)
/* */
/* ensure that the job being added isn't already there */
/* remember that as the comparison is for prefixes     */
/* that we must compare on left part of the name       */
/* */
DO x=1 to 20 /* maximum 20 jobnames */
compare=STRIP(job_name.x) /* get the name minus blanks */
IF LENGTH(compare)<1 THEN ITERATE /* must be something to compare */
IF LEFT(name,LENGTH(compare))=compare THEN DO
   zedsmsg='Name already covered'
```

```
   zedlmsg='by the prefix' job_name.x
   ADDRESS ISPEXEC 'SETMSG MSG(ISRZØØ1)'
   SIGNAL retry
   END
END
/* */
/* date must be numeric (already panel validated) */
/* but is it a valid date?                        */
/* */
IF LENGTH(date)<1Ø THEN DO
   zedsmsg='Not all of date supplied'
   zedlmsg='Please enter a date of the form YYYYDDD'
   ADDRESS ISPEXEC 'SETMSG MSG(ISRZØØ1)'
   SIGNAL retry
   END
 /* */
IF RIGHT(date,4)>2ØØ4 | RIGHT(date,4)<1996 THEN DO
   zedsmsg='Year out of range'
   zedlmsg='only 1996 to 2ØØ4 allowed'
   ADDRESS ISPEXEC 'SETMSG MSG(ISRZØØ1)'
   SIGNAL retry
   END
ELSE year=RIGHT(date,4) /* need the year */
/* */
/* calculate if this is a leap year */
/* */
IF year//4 = Ø THEN DO       /* this is a leap year */
   fudge=1
   mon.2=29
   END
ELSE DO                      /* no it isn't */
   fudge=Ø
   mon.2=28
   END
/* */
IF SUBSTR(date,4,2)>12 | SUBSTR(date,4,2)=Ø THEN DO
   zedsmsg='Unknown month'
   zedlmsg='There are 12 months in a year!'
   ADDRESS ISPEXEC 'SETMSG MSG(ISRZØØ1)'
   SIGNAL retry
   END
ELSE month=STRIP(SUBSTR(date,4,2),'L','Ø') /* need the month number */
/* */
day=LEFT(date,2)      /* now calculate day correctness */
/* */
IF day=Ø THEN DO
   zedsmsg='No day number supplied'
   zedlmsg='Please enter a correct date'
   ADDRESS ISPEXEC 'SETMSG MSG(ISRZØØ1)'
   SIGNAL retry
   END
/* */
IF day>mon.month THEN DO
   zedsmsg='Invalid day number'
```

```
   zedlmsg="this month doesn't have that many days"
   ADDRESS ISPEXEC 'SETMSG MSG(ISRZØØ1)'
   SIGNAL retry
   END
/* */
/* now calculate the date (finally!) */
/* */
IF month<3 THEN fudge=Ø /* fudge only applies after february */
daynum=RIGHT('ØØØ'||month.month+day+fudge,3)
date=year||daynum
/* */
CALL WTOLIST      /* look for reply id to use */
/* */
IF rc=4 THEN DO   /* rc 4 means no replies to issue */
   CALL mess_details
   EXIT
   END
/* */
/* if we reach here the tachyons appears to be functional */
/* now scan the wtor queue for the message reply number */
/* */
DO x=1 TO job_name.Ø
IF job_name.x=tachyons_jobname THEN DO
   literal=reply_id.x"ADD="jobl||name||date
   CALL REXWTO literal
   EXIT
   END
END
EXIT
mess_details:
zedsmsg='TACHYONS not active'
zedlmsg='Issue TIMESTAR to start TACHYONS'
ADDRESS ISPEXEC 'SETMSG MSG(ISRZØØ1)'
RETURN
```

ADDREXXP - PANEL FOR THE ADD COMMAND

```
)Attr default(%+_)
)Body Window(7Ø,6)
+
+Please enter a jobname and associated date
+
+Jobname%===>_name    +
+Date   %===>_date      + (DD/MM/YYYY)
+
)init
.help=addrexxh
)proc
&reply=.resp
   VER (&NAME,NB,NAME)
   VER (&date,PICT,99/99/9999)
   VER (&date,NB)
)End
```

ADDREXXH – HELP PANEL FOR THE ADD COMMAND

```
)ATTR
' TYPE(PT)                          /* panel title line            */
)BODY
'─────── Help Panel For ADD ──────────────────
+
+Please specify a job name prefix of up to 8 characters for TACHYONS
+to monitor, and a date in the range Ø1/Ø1/1996 to 31/12/2ØØ4 for that
+job.
+Note that it is possible to use a 'wildcard' approach to specifying
+jobnames by using the '#' character. Hence a jobname of AJOB#A would
+mean trap any jobname that was at least 6 characters long, and had
+AJOB to begin with, anything in the 5th character, and A in the 6th.
+
+Please ensure that the date is entered in the form DD/MM/YYYY.
)PROC
.help=isp00004
)END
```

DELREXX - REXX FOR THE DEL COMMAND

```
/* REXX */
/* */
/* prepare a suitable delete command for tachyons */
/* */
CALL SVC11REX
IF job_status='INACTIVE' THEN DO /* check if tachyons operating */
   CALL mess_details
   EXIT
   END
ADDRESS ISPEXEC
'ADDPOP ROW(1) COLUMN(9)'
'DISPLAY PANEL(DELREXXP)'
'REMPOP'
IF REPLY='END' THEN DO
   zedsmsg='Del aborted'
   zedlmsg='No action taken'
   ADDRESS ISPEXEC 'SETMSG MSG(ISRZØØ1)'
   EXIT
   END
/* */
/* as both svc11rex and wtolist use job_name for a variable */
/* it is necessary to rename the entries from svc11rex      */
/* because we will need them later                          */
/* */
DO x=1 TO 2Ø     /* svc11rex always generates 2Ø entries */
svc11_job.x=job_name.x
END
/* */
DROP job_name.    /* clear the array for reuse */
/* */
CALL WTOLIST      /* look for reply id to use */
```

```
IF rc=4 THEN DO   /* rc 4 means no replies to issue */
   CALL mess_details
   EXIT
   END
/* */
/* if we reach here the tachyons appears to be functional */
/* now scan the wtor queue for the message reply number */
/* */
DO x=1 TO job_name.Ø
IF job_name.x=tachyons_jobname THEN LEAVE /* now have the reply */
END
replynum=reply_id.x
DO x=1 TO 2Ø    /* now scan to confirm that job is deletable */
IF svc11_job.x=name THEN DO
   literal=replynum"DEL="name
   CALL REXWTO literal
   EXIT
   END
   END
zedsmsg=name 'not found'
zedlmsg='Delete not issued to TACHYONS'
ADDRESS ISPEXEC 'SETMSG MSG(ISRZØØ1)'
EXIT
mess_details:
zedsmsg='TACHYONS not active'
zedlmsg='Issue TIMESTAR to start TACHYONS'
ADDRESS ISPEXEC 'SETMSG MSG(ISRZØØ1)'
RETURN
```

DELREXXP - PANEL FOR THE DELETE COMMAND

```
)Attr default(%+_)
)Body Window(7Ø,5)
+
+Please enter jobname to be deleted
+
+Jobname%===>_name    +
+
)init
.help=delrexxh
)proc
&reply=.resp
   VER (&NAME,NB,NAME)
)End
```

DELREXXH – HELP PANEL FOR THE DEL COMMAND

```
)ATTR
' TYPE(PT)                          /* panel title line            */
)BODY
'──────── Help Panel For DEL ──────────────────
+
```

```
+Simply specify the jobname to be deleted. This must precisely match
+a name already displayed on the dialog.
+
+Failure to enter an exact match will terminate the DEL with an error
+message.
)PROC
.help=isp00004
)END
```

TERMREXX – REXX FOR SHUTTING DOWN TACHYONS

```
/* REXX */
/* */
/* prepare a suitable terminate command for tachyons */
/* */
CALL SVC11REX
IF job_status='INACTIVE' THEN DO /* check if tachyons operating */
   CALL mess_details
   EXIT
   END
ADDRESS ISPEXEC
'ADDPOP ROW(1) COLUMN(9)'
'DISPLAY PANEL(TERMREXP)'
'REMPOP'
IF REPLY='END' THEN DO
   zedsmsg='Term aborted'
   zedlmsg='No action taken'
   ADDRESS ISPEXEC 'SETMSG MSG(ISRZ001)'
   EXIT
   END
CALL WTOLIST
DO x=1 to job_name.0
IF job_name.x=tachyons_jobname THEN DO
   literal=reply_id.x”TERM”
   CALL REXWTO literal
   EXIT
   END
CALL mess_details
EXIT
mess_details:
zedsmsg='TACHYONS not active'
zedlmsg='TERM therefore not required'
ADDRESS ISPEXEC 'SETMSG MSG(ISRZ001)'
RETURN
```

TERMREXP – CONFIRMATION PANEL FOR THE TERM COMMAND

```
)Attr default(%+_)
)Body Window(70,4)
+
+Press ENTER to confirm TACHYONS termination
```

```
+Press END to cancel termination
+
)init
.help=termrexh
)proc
&reply=.resp
)End
```

TERMREXH – HELP PANEL FOR TERM COMMAND

```
)ATTR
' TYPE(PT)                          /* panel title line            */
)BODY
'──────── Help Panel For TERM ──────────────────
+
+Please press ENTER if you wish to remove TACHYONS from the system,
+and PF3 if you wish to allow TACHYONS to remain active.
+
+Terminating TACHYONS will remove all date front ending from the system.
)PROC
.help=isp00004
)END
```

SETUPREX – REXX TO DRIVE THE SET-UP COMMAND

```
/* REXX */
retry:
ADDRESS ISPEXEC
zwinttl='Specify set-up library'
'ADDPOP ROW(1) COLUMN(1)'
'DISPLAY PANEL(SETUPP1)'
'REMPOP'
IF REPLY='END' THEN EXIT
IF SYSDSN(dsname)¬='OK' THEN DO
   zedsmsg=dsname 'not found'
   zedlmsg='Please specify another library'
   'SETMSG MSG(ISRZ001)'
   SIGNAL retry
   END
'TBCREATE SET NAMES(mem) NOWRITE REPLACE'
ADDRESS TSO 'ALLOC FI(DD1) DA('dsname') SHR REUS'
MEM=''
'LMINIT DATAID(ABC) DDNAME(DD1) ENQ(SHR) '
'LMOPEN DATAID('ABC') OPTION(INPUT)'
DO FOREVER
   'LMMLIST DATAID('ABC') OPTION(LIST) MEMBER(MEM)'
   IF rc\=0 THEN LEAVE
   'TBADD SET'
END
'LMCLOSE DATAID('ABC')'
'LMFREE DATAID('ABC')'
```

```
'TBTOP SET'
re_display:
'TBDISPL SET PANEL(SETUPP2)'
IF reply='END' THEN DO
   ADDRESS TSO 'FREE FI(DD1)'
   EXIT
   END
IF ztdsels=Ø THEN SIGNAL re_display
IF ztdsels = 1 THEN DO
   ADDRESS TSO 'FREE FI(DD1)'
   mem=STRIP(mem)
   ADDRESS TSO '%SETUP' dsname'('mem')'
   zedsmsg='Additions completed'
   zedlmsg='Returning to main display'
   'SETMSG MSG(ISRZØØ1)'
   EXIT
   END
IF ztdsels > 1 THEN DO
   zedsmsg='Please select only one member'
   zedlmsg='Too many entries specified'
   'SETMSG MSG(ISRZØØ1)'
   select=''
   SIGNAL re_display
   END
```

SETUPP1 – SELECT PANEL FOR SPECIFYING LIBRARY CONTAINING SET-UP MEMBER

```
)Attr Default(%+_)
)Body Window(74,1)
%>_dsname
)init
.help=setuph1
)proc
&reply=.resp
VER (&dsname,NB,DSNAME)
)End
```

SETUPP2 – SELECT PANEL TO IDENTIFY THE MEMBER TO USE

```
)Attr Default(%+_)
   ! type(output) intens(high) caps(on ) just(left )
)Body  Expand(//)
/ /%   Member Choice List / /
%Command ===>_zcmd                                / /%Scroll ===>_amt +
+
+Please select%1+member only from!dsname
+
   Member name
)Model
_z!z
```

```
)Init
   .Help = setuph2                 /* insert name of tutorial panel  */
   .ZVARS = '(select mem)'
   &amt = PAGE
)PROC
&REPLY = .RESP
)End
```

SETUPH1 – HELP PANEL FOR PANEL SETUPP1

```
)BODY
'──────── Help Panel For SETUP DSN ─────────────────
+
+Please specify the name of the dataset containing the member where
+you have specified your job prefixes and dates. The system will
+validate the existence of the dataset before continuing on to the
+select member screen.
)PROC
.help=isp00004
)END
```

SETUPH2 – HELP PANEL FOR PANEL SETUPP2

```
)BODY
'──────── Help Panel For Member Choice List ────────────
+
+Please place a character at the side of the member which you wish to
+use to set up TACHYONS. If during the addition any errors are detected,
+messages will be SAYed to the screen for each problem job or date.
+
+Press PF3 to abort the process.
)PROC
.help=isp00004
)END
```

SETUP – REXX COMMAND FOR BULK SET-UP (CAN BE USED IN BATCH THROUGH WAITER)

```
/* REXX */
 RG dsn
IF dsn='' THEN dsn='your.PDS.library(member)' /* default setup member */
/* */
/* check if tachyons is active. if not abort now. */
/* */
CALL mess_details
/* */
'ALLOC FI(SPONGE) DA('DSN') SHR REUS'
'EXECIO * DISKR SPONGE (FINIS'
/* */
```

```
/* now preset the date variables for validation purposes */
/* */
CALL presets
/* */
/* retrieve the data to be added and issue the appropriate commands */
/* */
DO QUEUED()  /* loop around the setup details */
   skip_flag='N'
   PULL line
   name=WORD(line,1)
   IF LEFT(name,1)='*' THEN ITERATE /* line is a comment line */
   date=WORD(line,2)
   jobl=LENGTH(name)-1
   name=LEFT(name,8)
/* */
/* ensure that the job being added isn't already there */
/* remember that as the comparison is for prefixes     */
/* that we must compare on left part of the name       */
/* */
   DO x=1 TO 2Ø /* maximum 2Ø jobnames */
   compare=STRIP(job_name.x) /* get the name minus blanks */
   IF length(compare)<1 THEN ITERATE /* must be something to compare */
   IF LEFT(name,length(compare))=compare THEN DO
      SAY name 'already covered by prefix' job_name.x
      skip_flag='Y'
      END
   END
 /* */
 /* date must be numeric and of the form dd/mm/yyyy */
 /* but is it a valid date?                         */
/* */
   IF SUBSTR(DATE,3,1)¬='/' | SUBSTR(date,6,1)¬='/' THEN DO
      SAY 'Invalid format date for' name
      skip_flag='Y'
      END
/* */
      IF LENGTH(date)<1Ø THEN DO
      SAY 'Not all of date supplied for' name
      skip_flag='Y'
      END
/* */
   IF RIGHT(date,4)>2ØØ4 | RIGHT(date,4)<1996 THEN DO
      SAY 'Year out of range for' name 'only 1996 to 2ØØ4 allowed'
      skip_flag='Y'
      END
   ELSE year=RIGHT(date,4) /* need the year */
/* */
/* calculate if this is a leap year */
/* */
   IF year//4 = Ø THEN DO       /* this is a leap year */
      fudge=1
      mon.2=29
      END
```

```
   ELSE DO                        /* no it isn't */
      fudge=0
      mon.2=28
      END
   /* */
      IF SUBSTR(date,4,2)>12 | SUBSTR(date,4,2)=0 THEN DO
      say 'Unknown month for' name
      skip_flag='Y'
      END
   ELSE month=STRIP(SUBSTR(date,4,2),'L','0')
/* */
   day=LEFT(date,2)      /* now calculate day correctness */
/* */
   IF day=0 THEN DO
      SAY 'No day number supplied for' name
      skip_flag='Y'
      END
   /* */
   IF day>mon.month THEN DO
      SAY 'Invalid day number for' name
      skip_flag='Y'
      END
/* */
/* now calculate the date (finally!) */
/* */
   IF skip_flag='N' THEN DO   /* if all ok then issue command */
      IF month<3 THEN fudge=0 /* fudge only applies after february */
      daynum=RIGHT('000'||month.month+day+fudge,3)
      date=year||daynum
      message_sent='N'
/* */
/* keep looping around the wtor Q until the add command has been */
/* done. Put in a loop protect of 100000 just in case.           */
/* */
      check=1
      DO UNTIL message_sent='Y' | check > 100000
/* */
         CALL WTOLIST      /* look for reply id to use */
         count=job_name.0
         IF count=0 THEN ITERATE
/* */
/* if we reach here the tachyons appears to be functional */
/* now scan the wtor queue for the message reply number */
/* */
        DO x=1 TO count
           IF message_sent='Y' THEN ITERATE
           IF job_name.x=tachyons_jobname THEN DO
              CALL mess_details /* reset job name to in core values */
              DO y=1 to 20
                 IF name=job_name.y THEN DO
                    y=20
                    message_sent='Y'
                    END
              END
```

```
               IF message_sent¬='Y' THEN DO
                  literal=reply_id.x”ADD=”jobl||name||date
                  CALL REXWTO literal
                  END
               END
         END
         check=check+1
      END
   IF check=100000 THEN DO
      SAY 'additions taking too long. please try again'
      EXIT
      END
   END
END
EXIT
CALL mess_details
EXIT
/* */
/* validate if tachyons is active */
/* */
mess_details:
CALL SVC11REX
IF job_status='INACTIVE' THEN DO /* check if tachyons operating */
   IF SYSVAR(SYSENV)='FORE' THEN DO
      zedsmsg='TACHYONS not active'
      zedlmsg='Issue TIMESTAR to start TACHYONS'
      ADDRESS ISPEXEC 'SETMSG MSG(ISRZ001)'
      END
   ELSE DO
      SAY 'TACHYONS not active'
      SAY 'Issue TIMESTAR to start TACHYONS'
      END
   EXIT
   END
RETURN
/* */
presets:
/* */
/* first thing to do is build a table of days in months for */
/* converting dd/mm/yyyy into the yyyddd format for the     */
/* commands.                                                */
/* */
month.1=0                    /* jan */
month.2=31                   /* feb */
month.3=59                   /* mar */
month.4=90                   /* apr */
month.5=120                  /* may */
month.6=151                  /* jun */
month.7=181                  /* jul */
month.8=212                  /* aug */
month.9=243                  /* sep */
month.10=273                 /* oct */
month.11=304                 /* nov */
```

```
month.12=334                  /* dec */
/* */
mon.1=31                    /* jan */
mon.2=28                    /* feb */
mon.3=31                    /* mar */
mon.4=3Ø                    /* apr */
mon.5=31                    /* may */
mon.6=3Ø                    /* jun */
mon.7=31                    /* jul */
mon.8=31                    /* aug */
mon.9=3Ø                    /* sep */
mon.1Ø=31                   /* oct */
mon.11=3Ø                   /* nov */
mon.12=31                   /* dec */
RETURN
```

WAITER - REXX DRIVER FOR SETUP. THIS WAITS FOR TACHYONS TO BE ACTIVE

```
/* REXX */
ARG DSN
CHECK=Ø
DO UNTIL CHECK=1ØØØØØ | JOB_STATUS='ACTIVE'
CALL SVC11REX
END
IF JOB_STATUS='ACTIVE' THEN DO
   ADDRESS TSO '%SETUP' DSN
   EXIT
   END
ELSE EXIT 4
```

THE JCL TO START TACHYONS

```
//jobname JOB your job card
//A EXEC PGM=TACHYONS
//STEPLIB DD DSN=your.load.library,DISP=SHR
```

MVS news

Sterling Software has announced the release of VISION:Phaseshift which insulates MVS applications from Year 2000 date issues. VISION:Phaseshift is based on the premise that if applications do not see the transition from 99 to 00, then associated logic problems can be avoided. VISION:Phaseshift encapsulates both application code and data, dynamically shifting dates back in time so that all dates to be processed fall within the same century.

VISION:Phaseshift is a language independent run-time utility, which resides between the application and the operating system and automatically intervenes as data is read into an application to shift dates back in time. As data is written from an application,VISION:Phaseshift intervenes and shifts dates forward. Because date shifting occurs dynamically at the I/O level, both application programs and data are insulated from change.

VISION:Phaseshift for MVS/OS 390 supports QSAM, VSAM, BSAM, BDAM, IMS/DB, DB2, CICS, IMS/DC, and TSO.

For further information contact:
Sterling Software, Applications Development Division, 3340 Peachtree Road, NE, Suite 1100, Atlanta, GA 30326-1050, USA.
Tel: (404) 231 8575
Fax: (404) 364 0522.

Sterling Software (UK) Ltd,
Applications International Division,
Littleton Road, Ashford, Middlesex, TW15 1TZ, UK.
Tel: (01784) 212000
Fax: (01784) 212121.

* * *

IBM has announced the Runtime Analyser for MVS and OS/390 year 2000 analysis tool. The *free* code helps identify date exposures at run-time and provides a run-time remediation mechanism. It works during program execution, rather than with source code, and runs concurrently with other applications on production systems. It can test multiple applications in different jobs concurrently and accepts a range of input search criteria.

The execution time-based audit tool is designed to work on load modules and can be used with or without source code. It allows user-written filter and post-processing programs, and handles on-line and batch applications to cover all the different types of application.

Also, it's processor independent, running on any existing System/390 year 2000-ready processors. And it supports batch, started tasks, CICS, and IMS applications. It requires either MVS/ESA SP Version 5 or later, or OS/390 Version 1 or later.

Contact your local IBM representative for further information.