February 2001

In this issue

3  Updating the LLA from a batch program
15 Wait function for REXX programs
21 Automated and interactive library update using edit macros
42 Compressing contiguous characters in a file
53 Using REXX and the Web without OpenEdition
72 MVS news

© Xephon plc 2001
Editor
Jaime Kaminski

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

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

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

© Xephon plc 2001. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.
Updating the LLA from a batch program

OVERVIEW

Refreshing the LLA can be a difficult process to manage at times, in particular if it contains user libraries that need to be refreshed regularly. Unless the entire LLA address space is bounced, any change needs to be specified in SYSx.PARMLIB. This is easy enough, except that users do not normally have access to this dataset, meaning that the systems programmer (not to mention the change manager!) has to be involved. The LLACOPY macro can be used to overcome this problem but, as is the case with most good things in OS/390, it requires the skills of an Assembler programmer to implement.

There are three ways to refresh the LLA. These include adding a library to the LLA list in PARMLIB and bouncing the entire started task, selectively adding, removing, or refreshing libraries by using the CSVLLAxx PARMLIB members, or by calling the LLACOPY macro. This program is an example of making use of the macro to do a REFRESH for a library that is already in the LLA. The user has the option to do any of the following:

• REFRESH an entire library.
• REFRESH one or more members of a library.
• Tell the LLA that one or more members have been deleted from the library and that they should be removed from the LLA’s directory.

The LLACOPY macro is the only documented API in the LLA service. It works very simply – all it requires is that the library to be refreshed is allocated and that register 1 is pointing at a BLDL list of the library in memory. The BLDL list is effectively a list telling LLA which members to refresh. Using the macro has an underlying ‘flaw’ though – if a member had been deleted from the library on the disk, a BLDL of the library does not contain the member name. This means that, when the LLACOPY macro is issued, LLA is not ‘told’ to refresh the deleted member and it remains in the LLA / VLF directories. The
name has to be specifically included in the list. This is important to remember when this utility is used because it is the only way to indicate to LLA that a member has been deleted. Our list of examples on how to use this program includes an example on how to delete a member in the LLA.

The program requires a DD-statement by the name of LLADSN to point to the dataset that needs to be refreshed (the user must have at least READ access to the dataset). Specifying the members to be updated can be done in one of two ways – either through a SYSIN DD statement or, in the case of a single or all (but not just a few) members, by means of a passed parameter in the format, ‘PARM=*’.

Examples of calling the utility are shown below:

```plaintext
//S1 EXEC PGM=LLABATCH, MEMBER='*' - Refreshes all members
//LLADSN DD DSN=dataset.to.be.refreshed, DISP=SHR

//S1 EXEC PGM=LLABATCH, MEMBER='memname' - Refreshes one member
//LLADSN DD DSN=dataset.to.be.refreshed, DISP=SHR

//S1 EXEC PGM=LLABATCH
//LLADSN DD DSN=dataset.to.be.refreshed, DISP=SHR
SYSIN DD *

//S1 EXEC PGM=LLABATCH
//LLADSN DD DSN=dataset.to.be.refreshed, DISP=SHR
SYSIN DD *
MEMØ1 - Refreshes members MEMØ1, MEMØ2, and MEMØ3

//S1 EXEC PGM=LLABATCH
//LLADSN DD DSN=dataset.to.be.refreshed, DISP=SHR
SYSIN DD *
MEMØ1 - Refreshes members MEMØ1
MEMØ2(D) and MEMØ4, and deletes MEMØ3(D) members MEMØ2 and MEMØ3
MEMØ4
```

Note that the program does not support the option to refresh all members and also delete members in the same step. This can be done in two steps, alternatively the option can be coded into the program by those keen enough to do so.
This program will refresh one or more (or all) members of the data set allocated to LLADSN. A single member name or "**" (all members) can be specified with a JCL PARM. A single member, a list of members or "**" (all members) can be specified as parameters in a SYSIN file. If a SYSIN card is specified, it takes preference over a PARM= statement on the EXEC card.

The program will also refresh LLA for one or more deleted members. This is requested by adding (D) at the back of a member name. This is however only supported for the SYSIN DD specification, not when the name is passed as a parameter.

If a member is specified (without (D)) but does not exist, an error is given but the program still attempts to refresh the other members.

In order to speed up the call to LLA, only one LLACOPY macro is ever issued. This is the sequence of events:

1) Build a sorted list of the requested member(s) to be refreshed
2) Do a BDL to get a list of all members in the PDS
3) Allocate a new list and merge matching entries from 1) and 2)
4) Issue LLACOPY for the new list established in 3)

MODE: Program runs authorized, is non-reentrant
INPUT: Member name or "**" to indicate all members, ignored if a SYSIN DD-statement is specified.
OUTPUT: None
AMODE: 31
RMODE: 24
Caller's mode: Any (This program does a BAKR / PR)
Error messages: Messages related to invalid/ incomplete parms
Messages related to refresh failures
Called routns: None
DD-cards: SYSIN DD-card optional. Contains list of members to refresh or "**" to refresh all members.
(D) at back of member name indicates member has been deleted from dataset and should be refreshed
Special regs: None
R12= Base register
R13= Pointer to getmained area and save areas

LLABATCH CSECT
LLABATCH AMODE 31
LLABATCH RMODE 24
BAKR R14,Ø .Save Caller's Status
BALR R12,Ø
USING LOAD,12

***************************************************************
* Main driver routine
*****************************************************************************
LOAD   L     R4,Ø(R1)            .Ptr to compressed command text addr
STORAGE L     R3,STORSIZE
STORAGE OBTAIN,LENGTH=(3),LOC=ANY
LR    R2,R1                .Point to getmained area
XR    R9,R9
XR    R8,R8
MVCL  R2,R8               .Propagate binary zeros
USING STORAREA,R1
ST    R13,SAVEAREA+4
LR    R13,R1
DROP  R1
USING STORAREA,R13        .Addressability to getmained area
BAS R14,GETPARM .Go get parm or SYSIN
BAS R14,RFRSHDSN .Go refresh the dataset in LLA
FREETABL L     R2,TABSTART         .Where the table starts
LTR R2,R2               .Was it allocated?
BZ    FREEPLST            .No
L     R3,MEMAREA          .Load size of table to allocate
STORAGE RELEASE,LENGTH=(3),ADDR=(2)
*
FREEPLST L     R2,PARTLIST         .Where the table starts
LTR R2,R2               .Was it allocated?
BZ    FREEWRKA            .No
L     R3,PARTSIZE         .Load size of table to allocate
STORAGE RELEASE,LENGTH=(3),ADDR=(2)
*
FREEWRKA L     R4,RETCODE          .Pick up return code
L     R3,STORSIZE         .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
*****************************************************************************
* This routine reads the parm or the SYSIN DD statement
*****************************************************************************
GETPARM  BAKR  R14,Ø
RDJFCB1  RDJFCB SYSIN
LTR R15,R15              .Did user code a SYSIN card?
BNZ  CHKPARM  .No, look at input parameters
*
SYSINOK  OPEN SYSin
TM    SYSin+48,X'10'      .Did the file open?
B0    SYSinGM           .Yes
WTO 'LLABATCH(E): -Unexpected error on SYSIN file',ROUTCDE=11
ABEND Ø001
*
SYSinGM L     R3,MEMAREA          .Load size of table to allocate
STORAGE OBTAIN,LENGTH=(3),LOC=BELow
LTR R15,R15              .Did we get the storage?
BZ LBL .Yes
NOSTOR WTO 'LLABATCH(E): -Could not obtain storage',ROUTCDE=11
ABEND 0002
*
LBL LR R4,R1 .Start of Table
ST R4,TABSTART .The size of the table
ST R3,TABSIZE .The size of the table
XR R2,R2 .Clear the counter
SYSINLP GET SYSIN
CLI Ø(R1),C'**' .All members?
BNE MOVENAME .Yes
OI ALLMEMIN,YES .Set all-member flag
B SYSINLP .Go set flag & get out
MOVENAME MVC Ø(8,R4),Ø(R1) .Move the name into the table
BAS R14,DELCHECK .Go see if the member is to be dlted
LA R4,9(R4) .Point to next slot in table
LA R2,1(R2) .Bump up counter
C R2,MAXMEMS .Have we reached the max?
BNH SYSINLPX .No
WTO 'LLABATCH(E): -Only 20000 member names allowed in SYSIN'.X
ROUTCDE=11
ABEND 0003
*
SYSINLPX B SYSINLP .Do for all SYSIN cards
ENDSYSIN CLOSE SYSIN
CHK#IN LTR R2,R2 .Did we get any cards?
BNZ NUMMSG .Yes
TM ALLMEMIN,YES .User ONLY asked for all members, OK
BO GETPARMX .Get out
WTO 'LLABATCH(E): -SYSIN card but no member names specified'.X
ROUTCDE=11
ABEND 0004
*
NUMMSG ST R2,NUMSYSIN .Remember number of table entries
TM ALLMEMIN,YES .Did user also ask for all (*)?
BNO GOSORT
WTO 'LLABATCH(E): -Conflicting request to do all(*) members X
and selective list of members'.ROUTECD=11
ABEND 0005
*
GOSORT BAS R14,SORTTAB .LLACOPY requires sorted list
B GETPARMX .Get out
CHKPARM CLC Ø(2,R4),=H'Ø' .Has a parm been passed?
BNE PARMPASS .Yes
WTO 'LLABATCH(E): -No SYSIN DD-card or parameters'.ROUTECD=11
ABEND 0006
*
PARMPASS XR R2,R2
ICM R2,3,Ø(R4) .Length of passed parm
LA R4,2(R4) .Point to start of data
PARMASTR CLI Ø(R4),C'**' .All members?
BNE SELECT .No
OI ALLMEMIN,YES .Yes, set flag
B GETPARMX .Don't scan further

SELECT WTO 'LLABATCH(I): -Selection from input parm', X
ROUTCDE=11
CH R2,=H'8' .Maximum length of member name
BNH REDUCE
WTO 'LLABATCH(W): -Only 1 member taken from PARM card', X
ROUTCDE=11
LA R15,4
ST R15,RC
OC RETCODE,RC
LA R2,8 .Reduce the length to 8
REDUCE BCTR R2,Ø .Correct the length
MVC MEMNAME,=8X'4Ø' .Make blanks in case length< 8 bytes
EX R2,MOVEINST .Update the move instruction
OI ONEMEM,YES .Set flag
B GETPARMX

MOVEINST MVC MEMNAME(Ø),Ø(R4) .Pick up the member name from parm

GETPARMX PR

*********************************************************************
*        This routine scans for "(D)" in the member name
*********************************************************************
DELCHECK BAKR R14,Ø
LA R2,1(R1) .Start scanning from second byte
LA R3,77 .Max # bytes to scan
DELLOOP CLC Ø(3,R2),=C'(D)' .Does it contain delete option?
BE DELFOUND .Yes
BCT R3,DELOLLOP .Scan entire card
B DELCHECX .Not found, get out
DELFIND MVC Ø(8,R4),=8C' ' .Clear the name
MVI B(R4),C'D' .Move a "D" into the 8th byte
SR R2,R1 .Offset where we found "(D)"
BCTR R2,Ø .Correct the length for MVC
STC R2,MVCNAME+1 .Update MVC instruction
MVCNAME MVC Ø(Ø,R4),Ø(R1) .Move the name back into field
DELCHECX PR

*********************************************************************
*        This routine sorts the table containing the member names
*********************************************************************
SORTTAB BAKR R14,Ø
CLC NUMSYSIN,=F'1' .Only one entry?
BNH SORTTABX .Yes, no need to sort
L R1,NUMSYSIN .Number of entries in table
LR R2,R1
BCTR R2,Ø
XR R3,R3 .Outer loop counter
OUTLOOP EQU *
XR R4,R4 .Inner loop counter
INLOOP EQU *
LR R5,R3
Length of an entry

In correct sequence?

Swap the 2 names

This routine opens the data set, calls LLACOPY and closes
the dataset.

Is the DD-statement there?

Did the file open?

Did it work?

Did we do a BLDL for the dataset?

Are we doing all members?

No, partial refresh only

Yes, use entire list

Yes, dataset is ready for update

Could not do a BLDL for the dataset'

Could not open LLADSN'

LLABATCH(E): -LLADSN DD-statement missing',ROUTCDE=11

LLABATCH(E): -Could not open LLADSN',ROUTCDE=11

LLABATCH(I): -Partial refresh to be done',ROUTCDE=11

LLABATCH(E): -Partial refresh to be done',ROUTCDE=11

LLABATCH(E): -Could not do a BLDL for the dataset', X

LLABATCH(E): -Could not open LLADSN',ROUTCDE=11

LLABATCH(E): -Could not open LLADSN',ROUTCDE=11

LLABATCH(E): -Partial refresh to be done',ROUTCDE=11
 USING PRTDSECT,R3  .Addressability to partial list
 CLC   PARTNUM,=H'Ø'  .Any qualifying entries?
 BNZ   REFRESH        .Yes
 WTO 'LLABATCH(W): -No members selected for LLACOPY',ROUTCDE=11
 LA   R15,12
 ST   R15,RC
 OC   RETCODE,RC
 B   CLOSDIR         .Get out
 REFRESH BAS R14,LLACPMAC  .Issue LLACOPY macro
 LTR   R15,R15       .Successful?
 BZ   GIVEOKMS       .Yes
 NOTOKMS LR R2,R15   .Preserve our return code
 MVC WTOAREA(ERWTOLEN),ERWTOMS
 CVD R15,DOUBLE      .The return code from LLACOPY
 UNPK DOUBLE(4),DOUBLE+6(2)
 OI   DOUBLE+3,X'FØ'  .Make printable
 MVC WTOAREA+50(4),DOUBLE
 CVD R0,DOUBLE       .The reason code from LLABATCH
 UNPK DOUBLE(4),DOUBLE+6(2)
 OI   DOUBLE+3,X'FØ'  .Make printable
 MVC WTOAREA+63(4),DOUBLE
 LA R1,WTOAREA
 WTO MF=(E,(1)),ROUTCDE=11
 LR R15,R2          .Reload the return code
 B   CLOSDIR         .Get out
 GIVEOKMS EQU *      
 MVC WTOAREA(OKWTOLEN),OKWTOMS
 MVC WTOAREA+48(44),DSNAME
 LA R1,WTOAREA
 WTO MF=(E,(1)),ROUTCDE=11
 CLOSDIR ST R15,RC   .Return code already > Ø?
 OC   RETCODE,RC     .Yes
 CLOSIT   CLOSE LLADSN
 RFRSHDSX PR

******************************************************************************
* This routine builds a list of all the members in the PDS
******************************************************************************
 BUILDLST BAKR R14,Ø
 LA R7,MEMLIST          .Point to start of member list
 XR R8,R8               .Number-of-members counter
 READDIR LA R2,LLADSN   .Get next block
 GET (2)                .Get next block
 LR R4,R1               .Starting address of input data
 LA R4,2(R4)            .First member name = offset 2
 LH R2,Ø(R1)            .Number of active bytes in block
 SH R2,=H'2'           .Subtract length field
 MEMLOOP CLC Ø(8,R4),=B8'FF'   .Is this the last name?
 BE   ENDOFLST          .Yes
 CLI Ø(R4),X'ØØ'       .Last name in the block?
 BNH   READDIR          .Yes, it is - go read directory
 MVC Ø(8,R7),Ø(R4)
 LA R7,12(R7)          .Bump up the list pointer
LA R8,1(R8)  .Count number of entries
C R8,-F'65535'  .Max number of members allowed
BNH CALCOFST  .Calculate offset
WTO 'LLABATCH(E): -Dataset contains more than 65535 members,X
dynamic refresh not supported',ROUTCDE=11
LA R15,8
ST R15,RC
OC RETCODE,RC
B BUILDLSX  .Get out

* Dir entry: XXXXXXXX__Y + No of bytes.  XXXXXXXX = Name
* |________________________|  Y X 2 = No of bytes
*
CALCOFST IC    R6,11(R4)           .Load y into reg 6
SLL R6,27               .Push other bits out of reg 6
SRL R6,26               .Move back + multiply by 2
LA R6,12(R6)           .Add fixed 12 bytes of dir entry
SR R2,R6
AR R4,R6
LTR R2,R2               .Still bytes left?
BNP READDIR             .Go read next directory block
B MEMLOOP             .Get next member name in this block
ENDOFLST STH R8,NUMENTRY         .Plug into BLDL parm list
LA R8,12               .Length of each entry
STH R8,ENTLENG          .Plug into BLDL parm list
XR R15,R15             .This routine successful
BUILDLSX PR

*********************************************************************
*        This routine merges the SYSIN names or the member name as
*        passed with PARM='MEMNAME' with the BLDL list into a new list
*********************************************************************

PARTLBLD BAKR R14,Ø
TM ONEMEM,YES          .One member passed as a parm?
BNO FROMCARD          .No, member list from SYSIN
LA R3,1               .Space for 1 member only
LA R4,1               .Scan 1 member only
LA R5,MEMNAME         .Entry from PARM= is here
B CALCSPCE          .Go calculate the space
FROMCARD L R3,NUMSYSIN .Number of cards on SYSIN
L R4,NUMSYSIN         .Scan this many members
L R5,TABSTART        .Entries from SYSIN are here
CALCSPCE MH    R3,=H'12'  .BLDL size for member
LA R3,4(R3)          .BLDL list header
STORAGE OBTAIN,LENGTH=(3),LOC=ANY  Obtain storage to merge
LTR R15,R15           .Did we get the storage?
BZ KEEPADDR         .Yes
B NOSTOR           .Go ABEND
KEEPADDR ST R1,PARTLIST  .Address of the partial list
ST R3,PARTSIZE       .Size of the list
LR R3,R1            .Point to start of list
LA R3,PENTRIES      .Point to start of list
XR R8,R8            .Counter in new table

LR R8,R1   .Pointer in new table
XR R1Ø,R1Ø  .Match-found counter

BIGLOOP EQU *

LH R6,NUMENTRY  .Number of members in PDS
LA R9,MEMLIST   .Start of BLDL obtained from PDS
CLI B(R5),C'D'  .Is member marked for deletion?
BNE SMLLOOP     .No, go scan for it

MVC Ø(R3),Ø(R5)  .Move the name in
MVC 8(4,R3),=4'ØØ' .Move binary zeroes in
LA R3,12(R3)    .Point to next slot
LA R1Ø,1(R1Ø)   .Add 1 to match-found counter
B BUMPUP

SMLLOOP CLC Ø(R5),Ø(R9)       .Compare SYSIN with BLDL entry
BE MATCHFND      .Same
LA R9,12(R9)     .Bump up BLDL list pointer
BCT R6,SMLLOOP   .Do for each member in BLDL list
MVC NOMATCH+29(8),Ø(R5) .Move the member name into the WTO

NOMATCH WTO 'LLABATCH(W): -Member xxxxxxxx is not in the PDS, refreshX
could not be done',ROUTCDE=11
OC RETCODE,=F'4'  .Set return code to 4
B BUMPUP

MATCHFND EQU *

LA R1Ø,1(R1Ø) .Add 1 to match-found counter
MVC Ø(12,R3),Ø(R9) .Move entry into the partial list
LA R3,12(R3)   .Bump up the pointer

BUMPUP LA R5,9(R5)            .Bump up SYSIN list pointer
BCT R4,BIGLOOP  .Do for each member in SYSIN
L R3,PARTLIST  .Start of partial list
STH R1Ø,PARTNUM .Number of qualifying entries
LA R1Ø,12     .Size of each entry
STH R1Ø,PENTLENG .Plug the length of each entry

PARTLBLX PR
*********************************************************************
*        This routine does the actual LLACOPY
*********************************************************************

LLACPMAC BAKR R14,Ø
MODESET MODE=SUP,KEY=ZERO
MVC LLAAREA(LLAMACL),LLAMAC
LA R2,LLAAREA
XR R6,R6
ICM R6,3,Ø(R3)   .(Save) the original number
XR R5,R5
ICM R5,3,Ø(R3)   .Pick up # of entries to cater for
CH R5,=X'7FFF'   .More than 32K?
BNH LLALOCL1    .No, proceed
SRL R5,1        .Divide it by 2
STH R5,Ø(R3)    .Update the number to refresh
OI SPLTCOPY=YES  .Set flag to show we are doing 2

LLALOCL1 LLACOPY DCB=LLADSN,BLDLLIST=(R3),MF=(E,(2))
ST R15,RC
OC RETCODE,RC
CH R15,=H'4'    .RC > 4?
BH    RESETMD2            .Yes, get out
TM    SPLTCOPY,YES        .Is there a second half?
BNO   RESETMD2            .No, get out
SR    R6,R5               .Subtract # already done
MH    R5,=H'12'           .Multiply length with # done
AR    R3,R5               ,Move 'start of list' pointer
STCM  R6,3,Ø(R3)          .The number of entries in 2nd half
MVC   2(2,R3),=X'000C'    .Plug the entry length
LA    R2,LLAAREA
LLALOCL2 LLACOPY DCB=LLADSN,BLDLLLIST=(R3),MF=(E,(2))
ST    R15,RC
RESETMD2 MODESET MODE=PROB,KEY=NZERO
LLALOCLX OC    RETCODE,RC
DOPR   PR
*********************************************************************
*        Constants follow
*********************************************************************
LLADSN   DCB   RECFM=FB,LRECL=256,DSORG=PS,MACRF=GL,DDNAME=LLADSN, *
          BLKSIZE=256,EODAD=ENDOFLLST,EXLST=JFCB@
SYSIN    DCB   RECFM=FB,DSORG=PS,MACRF=GL,DDNAME=SYSIN,                *
          LRECL=8Ø,EODAD=ENDSYSIN,EXLST=JFCB@
DS    ØF
OKWTOMS WTO   'LLABATCH(I): -LLA refresh completed for DSN=           X
                     ',ROUTCDE=11,MF=L
OKWTOLEN EQU   *-OKWTOMS
ERWTOMS WTO  'LLABATCH(E): -Unsuccessful LLA refresh, RCODE=xxxx, ReasX
              on=YYYY',ROUTCDE=11,MF=L
ERWTOLEN EQU   *-ERWTOMS
LLAMAC    LLACOPY MF=(L,BLDLLLIST)
LLAMACL   EQU   *-LLAMAC
STORSIZE DC   AL4(LISTSIZE+GETMSIZE)
TABSIZE   DC   AL4(LISTSIZE)
LOWVALS  DC    F'Ø'
DS    ØF
JFCB@    DC   X'87',AL3(JFCB)     .Address of JFCB work area
JFCB     DS   CL176
MEMAREA  DS    ØF
DC   AL4(MEMSPCE)
MEMSPCE  EQU   18Ø000    .Space for 20 000 members in table
                         .Each entry = 9 bytes: 8-byte name
                         *                  + 1 byte set to "D" if member has
                         *                  to be deleted.
MAXMEMS  DS    ØF
DC   AL4(MEMSPCE/9)      .Maximum # of entries
LTORG
*********************************************************************
*        DSECTS follow
*********************************************************************
STORAREA DSECT
SAVEAREA DS    18F           .General savearea
PARMSTRT DS    F             .Start address of passed parms
RC       DS    F             .Return code received
RETCODE DS F .Return code passed to caller
TABSTART DS F .Start of table with parm members
PARTLIST DS F .Start of partial list
PARTSIZE DS F .Size of partial list
DOUBLE DS D .Double word work area

* 
NUMSYSIN DS F .Number of SYSIN cards
DSNAME DS CL44 .Name of the dataset for LLACOPY
MEMNAME DS CL8 .Name of the member name from parm
TEMPNAME DS CL9 .Temporary name holder used to sort
ALLMEMIN DS C .Flag to indicate “all members”
ONEMEM DS C .Flag to indicate one mem from parm
WTOAREA DS CL(OKWTOLEN) .Work area for WTO macro
DS ØF
LLAAREA DS CL(LLAMAACL) .Area for LLACOPY macro
DS ØD
BLDLLIST DS ØF | .BLDL parameter list. Don't move
NUMENTSY DS H |—
ENTLENG DS H | .Flag to indicate doing 2 LLACOPYs
MEMLIST DS ØH |
SPLTCOPY DS C .Flag to indicate doing 2 LLACOPYs
LISTSIZE EQU 2+2+(12*65535) .Allow for up to X'FFFF' members
GETMSIZE EQU *-STORAREA
PRTDSECT DSECT .DSECT for partial list
PARTNUM DS H .Number of entries in list
PENTLENG DS H .Length of each entry in list
PENTRIES DS ØH .Start of entry names

* 
NO EQU X'00'
YES EQU X'04'
POSTPONE EQU X'84'
IKJDAPØ8
IEFJFCBN
R0 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
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
END
Sometimes it is necessary to wait for a period of time in your program. In Assembler programs this is not a problem because you can code an STIMER macro to wait for a desired time. We wanted to know how to achieve this with a REXX program.

You can do this in a REXX program by calling a load module in it. Another method is to write an Assembler function. I developed a function for this purpose. This function gets a parameter from the caller. The format of this parameter is hhmmss, as a waittime, then it waits without using resources. Sometimes you have to stop waiting. By using STIMER the only way to do this is Cancel the program. This is not desirable so I changed the structure of the program.

This function attaches another program to the wait process. The main part waits for the end of the attached program or operator command from the console. If either event happens then it returns to the caller. The return value of the function can be NOP (attached TCB ended), STOP (STOP command entered by the operator), or a value that is entered by the MODIFY command. Also it can return PARMERROR (wait time for the TCB is not given), FORMATERROR (parameter format should be hhmmss), and PARMNOTNUMERIC (parameter should be numeric). The source of the program is shown below.

```
WAITREXX

*******************************************************************
**    Module Name : WAITREXX                                     **
**                                                               **
**    Type        : Function                                     **
**    Parameters  : Parm1 ( Wait time as HHMMSS Format )         **
**                                                               **
**    It gets a parameter. Format of this paramater is HHMMSS.   **
**    It waits for a time period that is given to the function   **
**    as this parameter. It attaches a subroutine to this module.**
**    This subroutine uses STIMER macro for wait process. The    **
**    function also waits for MODIFY command. If time ECB is     **
**    posted (STIMER) macro, it returns NOP. If modify command    **
**    is entered, then this function returns modify command.      **
**    User can also enter STOP command to function, then it       **
**    returns STOP.                                                **
**                                                               **
```
** Return values of this function. **

** Return         Reason                             Message  **
**    ——————— ————————————————  ———— **
**    NOP            No Error                               -    **
**    STOP           No Error ( STOP Command issued )       -    **
**    PARMERROR      Parameter should be given          WAITERR1 **
**    FORMATERROR    Parameter format should be HHMMSS  WAITERR2 **
**    PARMNOTNUMERIC Parameter should be numeric        WAITERR3 **
**    -              Invalid MODIFY command             WAITERR4 **
**    ?????????????? No Error, modify command value         -    **

*******************************************************************
WAITREXX CSECT
WAITREXX AMODE 31
WAITREXX RMODE ANY
*******************************************************************
REQUATE                     .Rename registers
*******************************************************************
SAVE (14,12),,'WAITREXX,Date:&SYSDATE,Time:&SYSTIME,Ergun'
BASR  R12,Ø
USING *,R12
LA    R2,SAVE
ST    R13,4(R2)
ST    R2,8(R13)
LR    R13,R2
*******************************************************************
LR    R11,R1
L    R10,EFPLARG
USING ARGTABLE_ENTRY,R10  .Addressability of argument table
L    R8,EFPLEVAL
L    R9,Ø(Ø,R8)
USING EVALBLOCK,R9        .Addressability of EvalBlock
L    R8,ARGTABLE_ARGSTRING_PTR
C    R8,=F'-1'   .Parameter exists?
BNE   CHECKLEN            .If yes, check length
ERR1     $REXXMSG  WAITERR1,34        .No parameter supplied
LA    R5,9              .Length of return value
ST    R5,EVALBLOCK_EVLEN .Put length into variable
MVC   EVALBLOCK_EVDATA(9),=C'PARMERROR' .Return Value
B    SETRC                .Return to REXX pgm.
*******************************************************************
CHECKLEN  L    R7,ARGTABLE_ARGSTRING_LENGTH
C    R7,=F'6'          .Parameter length should be 6
BE   CHECKNUM          .If yes, check numeric?
ERR2     $REXXMSG  WAITERR2,42        .Parameter length error
LA    R5,11            .Length of return value
ST    R5,EVALBLOCK_EVLEN .Put length into variable
MVC   EVALBLOCK_EVDATA(11),=C'FORMATERROR'
B    SETRC                .Return to REXX pgm.
*******************************************************************
CHECKNUM EQU   *

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
MVC INPARM(6),Ø(R8)       .Get wait time in HHMMSS format
NC FULLZERO(8),INPARM    .Is parameter numeric?
CLC FULLZERO,='00000000'
BE ATC_TCB               .If yes, attach TCB
ERR3 $REXXMSG WAITERR3,36       .Parameter does not numeric
LA R5,14                 .Length of return value
ST R5,EVALBLOCK_EVLEN   .Put length into variable
MVC EVALBLOCK_EVDATA(14),='PARMNOTNUMERIC'
B SETRC                 .Return to REXX pgm.

* ———— SUPPLIED PARAMETER OK, CONTINUE ————————-
*

ATC_TCB LA R5,ADRRESPA           .Address of Response area
EXTRACT (R5),FIELDS=COMM    .Command Scheduler Comm. List
L R5,ADRRESPA               .Get returned address
USING COM,R5               .Let Assembler know IEZCOM
ATTACH EP=WAITTCB,ECB=TCBECB,JSTCB=YES,PARAM=INPARM
ST R1,SAVETCB             .R1 points address of attached TCB
ICM R3,15,COMCIBPT        .Address of CIB
USING CIB,R3                .Let Assembler know IEZCIB
BZ NOCIB                 .BIF not present
BAL R14,FREECIB           .Free CIB

NOCIB DS ØH
QEDIT ORIGIN=COMCIBPT,CIBCTR=1
L 1,COMECBP
O 1,HIBITON
ST 1,ECBS+4

WAIT DS ØH
WAIT 1,ECBLIST=ECBS       .Wait for an ECB post
TM TCBECB,X'40'         .Attached ended?
BO TCBENDED             .If yes, process it
B CIBCIB                .No, Modify or Stop entered
TCBENDED LA R5,3           .Return
ST R5,EVALBLOCK_EVLEN   .Evalblock
MVC EVALBLOCK_EVDATA(3),='NOP'
DETACH SAVETCB             .Detach TCB
BAL R14,FREECIB          .REXX program1
B SETRC

CIBCIB L R3,COMCIBPT      .Check for Stop or Modify
CLI CIBVERB,CIBSTOP      .Stop entered?
BNE DOCIB               .If not, get modify value

STOPRTN LA R5,4           .Length of the return value
ST R5,EVALBLOCK_EVLEN   .Put it into EvalBlock
MVC EVALBLOCK_EVDATA(4),='STOP'
DETACH SAVETCB             .Detach TCB
BAL R14,FREECIB          .Return to REXX Pgm.

DOCIB DS ØH
CLI CIBVERB,CIBMODFY     .Check for Modify command
BNE NOCIB               .Modify not present
LA 4,CIBDATA            .Address of Modify command
LH 8,CIBDATLN          .Length of the modify command
SKIPBLNK DS 0H  .Skip blanks in the command
CLI 0(4),X'40'  
BNE ENDSKPBL  
LA 4,1(4)  
BCT 8,SKIPBLNK
ENDSKPBL C 8,=F'0'  .Check length of the command
BE INVPARM  .If no command entered, warning
ST R8,EVALBLOCK_EVLEN  .Length of the return value
LA R7,EVALBLOCK_EVDATA  
EX R8,MOVEPARM  
DETACH SAVETCB  .Detach TCB
BAL R14,FREECIB  
B SETRC  .Return to REXX program
MOVEPARM MVC 0(0,R7),0(R4)  .Move command into EvalBlock
INVPARM DS 0H
ERR4 $REXXMSG WAITERR4,45  .Invalid modify command
BAL 14,FREECIB  
B NOCIB
SETRC LA R15,0  .Set return code of the function
L R13,SAVE+4  
L R14,12(,R13)  
LM R1,R12,24(R13)  
BR R14  .Go back to caller
FREECIB DS 0H
QEDIT ORIGIN=COMCIBPT,BLOCK=(3)
BR 14
*******************************************************************
** Parameters of the Function                                    **
*******************************************************************
DS 0F
SAVE DS 18F  .Save area for the function
ADRRESPA DS A  .Address of the response area
TCBECB DS A
SAVETCB DS A  .Address of Attached TCB
ECBS DC A(TCBECB)
DS A  
*  .Error messages of the function
WAITERR1 DC CL34'WAITERR: Parameter should be given'
WAITERR2 DC CL42'WAITERR: Parameter format should be HHMMSS'
WAITERR3 DC CL36'WAITERR: Parameter should be numeric'
WAITERR4 DC CL45'WAITERR: Invalid parameter for MODIFY command'
INPARM DC CL8'00000000'  .Input parameter
FULLZERO DC CL8'00000000'  .Check for numeric
HIBITON DC X'80000000'  .For ECB post processing
DS 0F
LTORG
IRXEFPL  .Mapping macro for Ext.Func.Parm
IRXARGTB  .Mapping macro for argument table
IRXEVALB  .Mapping macro for EvalBlock
COM DSECT
IEZCOM
WAITTCB

Attached program for Wait purposes:

**************************************************
** Module Name : WAITTCB                         **
** Type        : Program                         **
** Parameters  : Parm1 ( Wait time as HHMMSS Format ) **
** It get's a parameter. Format of this parameter is HHMMSS. **
** It waits for a time that is given to this program as a **
** parameter. It executes STIMER macro. At the end of the **
** wait time it returns to caller.                    **
**************************************************

WAITTCB CSECT
WAITTCB AMODE 31
WAITTCB RMODE 24
REQUATE                   .Rename registers
SAVE  (14,12),,'WAITTCB,DATE:&SYSDATE,TIME:&SYSTIME,ERGUN'
BASR R12,Ø
USING *,R12
LA   R2,SAVE
ST   R13,4(R2)
ST   R2,8(R13)
LR   R13,R2
L    R2,Ø(R1)
MVC  WAITTIME,Ø(R2) .Move wait time into WAITTIME
STIMER WAIT,DINTVL=WAITTIME .Wait till the end
L    R13,SAVE+4 .Return to caller
ST   R15,16(.R13)
LM   R14,12,12(R13)
BR   R14
DS   ØF
SAVE DS 18F             .Save area of the program
WAITTIME DC C'ØØØØØØØØ' .Wait time as hhmmssstt
END

$REXXMSG

Macro for REXX error messages:

***START OF SPECIFICATIONS***************************/
/* MACRO-NAME = $REXXMSG */
/* DESCRIPTIVE NAME = Displays a message in a REXX program, by */
/* using IRXSAY macro */
/***END OF SPECIFICATIONS***************************/
Sample use of the Function in a REXX program:

```rexx
/**REXX**************************************************************************/
/* WAITTIME : Sample REXX program for usage of WAITREXX function. */
*******************************************************************************/
Pull WaitTime /* Get Initial WaitTime from user */
If WaitTime = '' Then /* No parameter given */
    WaitTime = '000100' /* Default WaitTime, 000100 */
ReturnValue = '' /* To Control looping */
Do Until ReturnValue = |STOP|
    Say 'Time is now ' || Time() || ' . program will wait ' ||
        WaitTime || ' ( As HHMMSS format )'
    RETURNVALUE = WAITREXX(WaitTime)
    If RETURNVALUE = |NOP| Then
        NOP
    Else If RETURNVALUE = <STOP< Then
        NOP
    Else
        If Pos(RETURNVALUE,'|PARMERROR-FORMATERROR-PARMNOTNUMERIC|') = 0 Then
            Do
                Say 'Error in parameter'
                Exit(20)
            End
        Else
            WaitTime = RETURNVALUE
        End
End
Exit
```

Ergun Ozel
Project Manager
Vizyon Bilgi Islem (Turkey)
Automated and interactive library update using edit macros

INTRODUCTION

Often there is a need to change strings in a library in an MVS environment. But changing a specific string in libraries with many members, it would be a very hard task to do without using an edit macro.

For example, assume that we have several JCL libraries related to a product, and that members of these libraries are referring to a back-level product qualifier, which is BBIOA.V514, and all these occurrences have to be substituted with BBIOA.V611, which is the new version qualifier.

I have developed a small utility which consists of REXX programs, edit macros, several panels, and a message library member to automate this sort of task, in a panel-driven and interactive way.

WHAT IS THE ALGORITHM?

The edit macro used in this article is run against all members of a PDS. A loop in the REXX program uses a LISTDS command output to obtain the PDS dataset members’ names. For each member, an ISPEXEC EDIT command is issued with the initial MACRO keyword.

(Ispexec Edit Dataset(...) Macro(...) command.)

When the main REXX CHGREXX is called, it will display the panel PANEL0 to get all ‘Change’ command parameters from the user, including the PO dataset to be updated (it is called as SOURCE dataset), ‘From’ string, ‘To’ string, etc. Then, a backup copy of the SOURCE dataset is created for backout purposes. (It is called as BKPCPY dataset).

Later on, CHGREXX reads all members of the SOURCE dataset and calls the edit macro CHGMAC within an ‘ISPEXEC EDIT DATASET’ command for each member read. If that member has at least one
occurrence of the ‘From string’, then those strings are changed and a ‘Save’ operation is performed. If the member is saved successfully, ISPF statistics of that member are changed as well. The ‘Save’ command in the edit macro sets the ID field (the user ID that last modified the data) as the TSO user-id. But the REXX CHGREXX sets the UID field as CHGMAC so that, at a later time, the user can easily distinguish members which were changed by the edit macro while editing the dataset.

Then the result of the edit macro execution is written, member by member, into an ISPF table, which will be displayed in Panel 3 in the next step.

Then CHGREXX compares the SOURCE dataset, which may have have been updated, with the BKPCPY dataset, which is the original version of the SOURCE dataset, by a SuperC utility and displays the differences found to the user. It is meant to show the user differences between the initial and the final contents of the SOURCE dataset. Finally, Panel 2 is displayed and the user is asked if she/he accepts the changes just made by the utility. Depending on his/her response, the BKPCPY dataset is kept or deleted. The Panel 1 displays an explanatory message to the user during the execution of the utility. There is also a help panel, which is Panel H.

HOW TO RUN THE REXX CHGREXX

To make the use of this application easy, I recommend putting all REXXs, panels, the edit-macro, and the message member into a single library. In this article, all of them are members of the dataset EXP.CTM.REXX. Then it will be sufficient just to call the REXX CHGREXX in the following way:

```
TSO EX 'EXP.CTM.REXX(CHGREXX)'
```

The REXX CHGMAC will do all necessary allocations for ISPF message, panels, the REXX, and the macro by using a series of ALTLIB and LIBDEF commands.

Another way, of course, is to put all panel definition members into an ISPPLIB, the REXX and the edit-macro into a SYSEXEC, and the message member into an ISPMLIB concatenation library. Note that if this method is chosen, all ALTLIB and LIBDEF commands, which are
not necessary, have to be removed from the REXX CHGREXX. Then you can call the REXX simply by its name:

TSO %CHGREXX

But for easy maintenance, I would recommend using the first approach.

A SHORT OVERVIEW ON EDIT MACROS

You can use edit macros, which look like ordinary editor commands, to extend and customize the editor. You create an edit macro by placing a series of commands into a dataset or member of a partitioned dataset. Then you can run those commands as a single macro by typing the defined name in the ‘Command ===>’ line in an edit session or you call an edit macro from a REXX. The second method not only provides us with the capability of running an edit macro many times in a loop, but also saves us time by not having to edit each member to be able to run the edit macro against that member.

Edit macros can be either CLISTs or REXX EXECs written in the CLIST or REXX command language, or program macros written in a programming language (such as FORTRAN, PL/I, or COBOL).

Edit macros have access to the dialog manager and system services. Because edit macros are CLISTs, or REXX EXECs, programs, they have unlimited possibilities. Edit macros can be used to:

- Perform repeated tasks
- Simplify complex tasks
- Pass parameters
- Retrieve and return information.

The REXX edit macros must include a REXX comment line (/* REXX */) as the first line of each edit macro to distinguish them from CLIST edit macros. This comment line can contain other words or characters if necessary, but it must include the string REXX.

REXX edit macros must be in partitioned datasets. REXX edit macros can exist in the following concatenations: SYSUEXEC, ALTLIB (for datasets activated as EXECs), and SYSEXEC. Datasets in these concatenations can contain only REXX EXECs.
A REXX edit macro is made up of REXX statements. Each statement falls into one of the following categories:

- Edit macro commands
- CLIST or REXX command procedure statements and comments
- ISPF and PDF dialog service requests
- TSO commands.

You can run PDF edit macros in batch by submitting JCL which allocates all of the necessary ISPF libraries, and runs a command that calls the EDIT service with an initial macro. This initial macro can do anything that can be done by an initial macro in an interactive session. However, in batch, the macro should end with an ISREDIT END or ISREDIT CANCEL statement. These statements ensure that no attempt is made to display the edit screen in batch.

We can pass parameters to an edit macro. A parameter can be either a simple string or a quoted string. It can be passed by using the standard method of putting variables into shared and profile pools (use VPUT in dialogs and VGET in initial macros and vice versa). This method is best suited to parameters passed from one dialog to another, as in an edit macro. The edit macro and the REXX used in this utility use the VGET/VPUT commands to communicate with one another. For more information on edit macros see the IBM book *ISPF Edit and Edit Macros*.

**SOME NOTES ON THE UTILITY**

A ‘From string’ and a ‘To string’ have to be specified in PANEL1 of the application. If the string is a simple or delimited string (*a string that begins and ends with apostrophes or quotes*), the characters are treated as being both upper and lower case even if caps mode is off. For example, this command:

```
CHANGE ALL 'CONDITION NO. 1' '........'
```

This changes the following:

- CONDITION NO. 1
- Condition No. 1
• condition no. 1
• coNDitION nO. 1

Also, all of the following commands have the same effect:

\[
\text{CHANGE } '\text{Edit Commands}' '.........' \\
\text{CHANGE } '\text{EDIT COMMANDS}' '.........' \\
\text{CHANGE } '\text{edit commands}' '.........'
\]

For this reason, if you want the change command to be satisfied by an exact character-by-character match, lower case alphabetic characters matching only with lower case alphabetic characters, and upper case alphabetic characters matching only with uppercase, a character string must be used. For example, to change ‘aBc’ to ‘AbC’ you have to use the command:

\[
\text{Change C'aBc' 'AbC' ALL}
\]

Besides character strings, you can use picture strings to change a particular kind of character without regard for the specific character involved. For example to change any lower case character to upper case, use the command:

\[
\text{CHG p'<' p>}'
\]

To change a string of hexadecimal digits, you can use a hex string. For example, CHG 'c1c2'x 'a1a2'x. To change a character string regardless of whether alphabetic characters are upper or lower case, use a text string. For example, the following command changes the text «spf» to caps:

\[
\text{CHG t'spf' SPF}
\]

To limit the strings that are found, you can use qualifying parameters. This way, you will specify additional characteristics of string-1 (from string) by using the operands PREFIX, SUFFIX, CHARS, and WORD.

• CHARS – locates string-1 anywhere the characters match. This is the default.
• PREFIX – locates string-1 at the beginning of a word.
• SUFFIX – locates string-1 at the end of a word.
• WORD String-1 – is delimited on both sides by blanks or other non-alphanumeric characters.
The col-1 and col-2 operands allow the user to search only a portion of each line, rather than the entire line. These operands, which are numbers separated by a comma or by at least one blank, show the starting and ending columns for the search. The following rules apply:

- If you specify neither col-1 nor col-2, the search continues across all columns within the current boundary columns (BOUNDS line).
- If you specify col-1, the editor finds the string only if the string starts in the specified column.
- If you specify both col-1 and col-2, the editor finds the string only if it is entirely within the specified columns.
- If the second column specified is larger than the record size, this is an error condition and the edit macro of the utility (CHGMAC) will substitute the second column with the record length of the dataset being changed.

ADDITIONAL NOTES AND CONCLUSION

1 This edit-macro with its accompanying REXX program can be used in the migration processes. For example, it is helpful to make multi-changes in the libraries when a new version of a product is implemented. Also you can make the edit macro a little more complex by using more edit primary commands such as ‘Change’ or ‘Exclude’. For this, Panel 1 has to be accommodated to the changes made on the edit-macro since they work together. For example, to incorporate one more Change command, you would have to prepare a continuation panel to Panel 1.

Another advantage of using this utility is that it permits us to change strings of up to 255 characters in length. Remember that an ordinary change command on the ISPF edit command line is not able to do that due to the limitation of the edit command line.

2 If it seems that the PO dataset has too many members and most of them are candidates to have updates when the REXX CHGREXX runs, special care must be taken regarding S37 abend. The REXX CHGREXX compresses the SOURCE dataset at the beginning of the utility as a precaution. For this reason, it
is recommended that you enlarge the PO dataset to tolerate possible growth due to many possible `Isredit Save’ commands in the edit macro (CHGMAC). However, if you see an S37 abend condition on the result panel Panel 3 at the end of the execution, you can compress the SOURCE dataset in another split-screen and re-execute the utility.

3 If it is not necessary to work interactively, you can make changes in the PO libraries without using the panel as well. On the other hand you can make changes in several PO datasets at the same time. The REXX program BATCHREX and the edit macro BATCHMAC are given as an example abridged codes and do not contain any error recovery control.

To execute, use the command:

```
TSO EX 'EXP.CTM.REXX(BATCHREX)'
```

**REXX: CHGREXX**

```rexx
/*REXX*/
/*————————————————————————————————————————————————————————————————————*/
/* AUTOMATED AND INTERACTIVE LIBRARY UPDATE BY USING EDIT MACRO */
/*————————————————————————————————————————————————————————————————————*/
/* REXX program       : ChgRexx */
/* Edit macro called  : ChgMac */
/*————————————————————————————————————————————————————————————————————*/
Status = Msg('Off')
/*————————————————————————————————————————————————————————————————————*/
/* Let EXEC process errors. This way we can check the Return Codes */
/* and take the appropriate actions. */
/*————————————————————————————————————————————————————————————————————*/
|Ispexec Control Errors Return|
/*————————————————————————————————————————————————————————————————————*/
/* Make necessary library allocations for REXX/Panel/Message members.*/
/*————————————————————————————————————————————————————————————————————*/
|Altlib  Activate Application(Exec) Da(Exp.Ctm.Rexx)|
|Ispexec Libdef Ispplib Dataset     Id(Exp.Ctm.Rexx)|
|Ispexec Libdef Ispmlib Dataset     Id(Exp.Ctm.Rexx)|
/*————————————————————————————————————————————————————————————————————*/
/* Clean some profile variables. */
/*————————————————————————————————————————————————————————————————————*/
|Ispexec Verase (Mes,Mem,Col1,Col2,Rc1,Rc2,Rc3,Cnt,Chg,Err,Msg1,Msg2)|
REPEAT:
/*————————————————————————————————————————————————————————————————————*/
/* Display the panel PanelØ to get the dataset name, and |Ispf Edit */
/* command| parameters. */
```
/* ————————————————————————————————————————————————————————————————————— */
«Ispexec Display Panel(Panell0)»

/* Control that if PFØ3 or PFØ4 key is pressed on the Panel0. If so, */
/* deallocate the libraries allocated by Libdef & Altlib commands. */

«Ispexec Vget (Spfkey,Dsn,Col1,Col2,From,To) Profile»
If (Spfkey = PFØ3 3 Spfkey = PFØ4) Then Do
  «Ispexec Setmsg Msg(Edtm006)»
  «Altlib Deactivate Application(Exec)»
  «Ispexec Libdef Ispplib»
  «Ispexec Libdef Ispmlib»
  Exit
End

/* ————————————————————————————————————————————————————————————————————— */
/* Check if the dset name entered on Panel0 is an existing PO dset. */
/* From now on, we will be calling this dataset as 'SOURCE dset.' */

IF Sysdsn(Dsn) = 'OK' Then
  Do
    X = Listdsi(Dsn)
    If X = Ø Then
      Say 'Some Listdsi info not available. Function code = ' X
    Else Do
      Dsorg = Sysdsorg
      If Dsorg = PO Then
        Do
          «Ispexec Setmsg Msg(Edtm009D)»
          Signal Repeat
        End
      End
    End
  Else Do
    «Ispexec Setmsg Msg(Edtm009C)»
    Signal Repeat
  End

/* ————————————————————————————————————————————————————————————————————— */
/* Check whether the SOURCE dset has any member? */

x = Outtrap('Var.')
«Listds» Dsn «Members»
x = Outtrap('Off')        /* Turns trapping OFF */
Cnt = Var.Ø-6
«Ispexec Vput Cnt Profile»
If Cnt = Ø Then Do
  «Ispexec Setmsg Msg(Edtm009B)»
  Signal Repeat
End

/* ————————————————————————————————————————————————————————————————————— */
/* Check that if From or To string that are entered on Panel0 is */
/* greater than the record length of the SOURCE dataset. */
/* Check whether Col1 or Col2 is greater than the LRECL of the SOURCE */
/* dataset. If so, do not continue any more. In addition, display */
/* the LRECL of the SOURCE dset for informative purposes. */
If (Col2 > L2) 3 (Col1 > L2) Then Do
   «Ispexec Setmsg Msg(Edtm009J)»
   Mes = '*** Lrecl : 'L2
   «Ispexec Vput Mes Profile»
   Signal Repeat
End

/* Compress the SOURCE dataset to prevent it from producing S37 abend */
Chgiebcp Dsn Dsn

/* Allocate a back-up copy of the SOURCE dataset. From now on we */
/* will be calling it as BKPCPY dataset. */
Dsn_bkp = Dsn'.BKP'
«Alloc Da(«Dsn_bkp») Like(«Dsn»)»
If Rc ¬=Ø Then
   Do
      Say 'BKPCPY dset allocation is not successful'
      If Rc =12 Then Say 'The dset, ' Dsn_bkp ' already exists.'.
         'Please check it out.'
      Exit
   End
«Free Da(«Dsn_bkp»)»   /* Free BKPCPY dataset. */

/* Call the system utility IEBCOPY to copy the SOURCE dataset */
/* members into the BKPCPY dataset. */
Chgiebcp Dsn Dsn_bkp

/* At this point, we have BKPCPY dataset built. So we can start */
/* executing the Edit macro over the SOURCE dataset members to do */
/* batch string updates. */

/* A table which will show edit-macro execution status for each */
/* member of the SOURCE dataset will be created. */
/* */
/* Table_output_library = Ispf profile dataset. (File Name = ISPTABL) */
/* Table_name = TABLExxx (xxx = 1,...,100) */
/* */
/* A user can ask to run this application more than once on a split */
/* screen. Each time a different table name will be generated. */
/* */
Prfxusr = Sysvar(sysuid)
«Alloc Fi(Isptabl) Da('«33 Prfxusr 33 «.Ispf.Ispprof') Shr»
Ran = Random(1,100)
Table_name = 'TABLE' 33 ran
«Ispexec Tbcreate» Table_name «Names(Mem Chg Err Msg1 Rc1 Rc2 Msg2)»
/* */
/* Get members of SOURCE dset and execute the edit-macro in a loop. */
/* */
Do i = 7 To Var.Ø /* Loop-Martapv */
    Mem = Strip(Var.i)
    Call Editmac
    Rcl = Result
    «Ispexec Vput (Mem,Rc1) Profile»
    «Ispexec Vget (Chg,Err) Profile»
    «Ispexec Vget (Msg2,Rc2,Rc3) Profile» /* Get «ISPEXEC EDIT» and */
    /* «SAVE» command Rc. */
    Select
        When (Rcl = Ø ) Then Msg1= 'Member updated. '
        When (((Rc1=4) & (Rc3>=4)) Then Msg1= 'No save.Abend S37'
        When (Rc1 = 4 ) Then Msg1= 'Member not saved.'
        When (Rc1 = 14) Then Msg1= 'Member in use. '
        When (Rc1 = 20) Then Msg1= 'Severe error. '
        Otherwise Nop
    End
    «Ispexec Vput Msg1 Profile»
/* */
/* Set the ISPF statistics for the SOURCE dataset members that are */
/* updated by the edit macro. (Changed members will have the |CHGMAC| */
/* string in their ID field. ) */
/* */
If RC3 = Ø Then
    Do
        «Alloc Fi(Stats) Da(«Dsn») Shr Reuse»
        «Ispexec Lmminit Dataid(Sta) Ddname(Stats) Enq(Shr)»
        «Ispexec Lmmstats Dataid(«Sta») Member(«Mem») User(Chgmac)»
«Ispexec Lmfree  Dataid("Sta")»
End

/* Add a new row to the «Edit-macro Result Table». */
/* ———————————————————————————————————————————————————*/
«Ispexec Tbadd» Table_name

End  /* End-of-Loop-Martapv */
/* ———————————————————————————————————————————————————*/
/* At last, display the «Edit-macro Result Table» and delete the */
/* virtual storage copy of this table. */
/* ———————————————————————————————————————————————————*/
«Ispexec Tbbox» Table_name
«Ispexec Tbbox» Table_name «Panel(Panel3)»
«Ispexec Tbbox» Table_name

/* ———————————————————————————————————————————————————*/
/* Compare the SOURCE and BKPCPY datasets with the SuperC utility. */
/* ———————————————————————————————————————————————————*/
«Alloc Fi(Newdd) Da("Dsn")  Shr Reuse»
«Alloc Fi(Olddd) Da("Dsn_bkp")  Shr Reuse»
«Alloc Fi(Outdd) Space(1,1) Cylinders New Keep Dsorg(Ps)»

/* ———————————————————————————————————————————————————*/
/* If RC is zero, it means that no change has been made to the SOURCE */
/* dataset. So we can delete the BKPCPY dataset and terminate the */
/* dialog. If RC is not zero, in this case this means that some */
/* changes have been made to SOURCE dataset. */
/* ———————————————————————————————————————————————————*/
Address Tso «Call 'Isp.Sisplpa(Isrsupc)' 'LONGL,LINECMP'»
If Rc = 0 Then
  Do
    «Ispexec Setmsg Msg(Edtm009I)>>
    Delete Dsn_bkp
    Signal De_Alloc
  End

«Ispexec Lminit Dataid(Villar) Ddname(Outdd) Enq(Shr)>>
Rc7 = Rc
If Rc7 = 0 Then
  Do
    «Ispexec Vput Rc7 Profile»
    «Ispexec Setmsg Msg(Edtm009G)>>
    Exit
  End

/* ———————————————————————————————————————————————————*/
/* Call POPUP1 PROC to display an explanatory message. */
/* ———————————————————————————————————————————————————*/
Call Popup1
/* ———————————————————————————————————————————————————*/
/* Display the SuperC output to the user. */
/*ispexec Browse Dataid("Villar")*/

/* Call POPUP2 to ask the user if he/she agrees with the changes on SOURCE dataset. If he/she does, BKPCPY dataset will be deleted. */

Call Popup2
De_Alloc:
«ispexec Lmfree Dataid("Villar")»
«Free File(Newdd,Olddd,Outdd)»
«Altlib Deactivate Application(Exec)»
«ispexec Libdef Ispplib»
«ispexec Libdef Ispmlib»

EXIT /* End-of-the-main-Rexx */

POPUP1:
Zwintt1 =
«ispexec Addpop Row(6) Column(25)»
«ispexec Display Panel(Pannel1)»
«ispexec Vget Spfkey Profile»
«ispexec Rempop»
RETURN /* End-of-the-procedure-POPUP1 */

POPUP2:
Zwintt1 = CONFIRMATION SCREEN
«ispexec Addpop Row(6) Column(25)»
«ispexec Display Panel(Pannel2)»
«ispexec Vget Spfkey Profile»

/* Keep displaying the Panel2 until the user enters 'Y' or 'N'. */

Do While( Spfkey=PFØ3 3 Spfkey = PFØ4 )
  «ispexec Setmsg Msg(EdtmØØ9A)»
  «ispexec Rempop»
  «ispexec Browse Dataid("Villar")»
  «ispexec Addpop Row(6) Column(25)»
  «ispexec Display Panel(Pannel2)»
  «ispexec Vget Spfkey Profile»
End
«ispexec Rempop»
«ispexec Vget Resp Profile»
If Resp = 'Y' Then Do
  Delete Dsn_bkp
  «ispexec Setmsg Msg(EdtmØØ9B)»
End
Else «ispexec Setmsg Msg(EdtmØØ9C)»
RETURN /* End-of-the-procedure-POPUP2 */

EDITMAC:
/* Control if the member is still there. */
Rs = Sysdsn(Dsn'(«Mem»)')
If Rs«OK» Then Nop
   Else Do
      Say 'The specified member is not found in dataset.'
      Return
   End
«Ispexec Edit Dataset('«Dsn»(«Mem»)') Macro(ChgMac)»
Return Rc /* End-of-the-procedure-EDITMAC */

REXX : BATCHREX

/*REXX*/
/--------------------------------------------------------------------------*/
/* REXX program : Batchrex */
/* Edit macro called : Batchmac */
/* Purpose : Change multiple strings in 3 libraries. */
/--------------------------------------------------------------------------*/
«Prof nopref»
«Altlib Activate Application(Exec) Da(Exp.Ctm.Rexx)»
Mpv.Ø = 1
Mpv.1 = Exp.Ctm.Test1
Mpv.2 = Exp.Ctm.Test2
Mpv.3 = Exp.Ctm.Test3
Do i = 1 to 3
   Dsn = Mpv.i
   x = Outtrap('Var.')
   «Listds» Dsn «Members»
   x = Outtrap('Off')       /* Turns trapping OFF */
/--------------------------------------------------------------------------*/
/* Get members of SOURCE dset and execute the edit-macro in a loop. */
/--------------------------------------------------------------------------*/
Do j = 7  To Var.Ø
   Mem = Strip(Var.j)
   «Ispexec Edit Dataset('«Dsn»(«Mem»)') Macro(Batchmac)»
   End
End
«Altlib Deactivate Application(Exec)» /* Deallocate the REXX library */
Exit

REXX PROCEDURE : CHGIEBCP

/*REXX*/
/--------------------------------------------------------------------------*/
/* Allocate files needed by IEBCOPY. */
/--------------------------------------------------------------------------*/
Arg Dsn1 Dsn2
«Alloc Fi(Input) Da('«Dsn1») Shr Reuse»
«Alloc Fi(Output) Da('«Dsn2») Shr Reuse»
«Alloc Fi(Sysout) Dummy Reuse»
/* Build the SYSIN control statements of the IEBCOPY utility. */
/* —————————————————————————————————————————————————————————————————————*/
/* Alloc Fi(Sysin) Unit(VIO) Space(1,0) Blksize(80) Lrecl(80), */
/* Recfm(F B) Dsorg(PS) New Delete Reuse */
/* Alloc Fi(Sysprint) Dummy Reuse */
/* Alloc Fi(Sysut1) Unit(VIO) Space(1,1) Cyl New Delete Reuse */
/* Alloc Fi(Sysut2) Unit(VIO) Space(1,1) Cyl New Delete Reuse */
/* Alloc Fi(Sysut3) Unit(VIO) Space(1,1) Cyl New Delete Reuse */
/* Alloc Fi(Sysut4) Unit(VIO) Space(1,1) Cyl New Delete Reuse */

/* ——————————————————————————————————————————————————————————————————*/
!
Rc5 = Rc
If Rc5 = 0 Then
   Do
      «Ispexec Vput Rc5 Profile»
      «Ispexec Setmsg Msg(Edtm009E)»
      Exit
   End
Else
   Do
      Card = ' COPY OUTDD=OUTPUT,INDD=INPUT'
      «Ispexec Lmopen Dataid('Martapv') Option(Output)»
      «Ispexec Lmput Dataid('Martapv'), */
      Dataloc(Card) Datalen(80) Mode(Invar)»
      «Ispexec Lmclose Dataid('Martapv')»
      «Ispexec Lmfree Dataid('Martapv')»
      Rc4 = Rc
   End
   If Rc4 = 0 Then Do
      «Ispexec Vput Rc4 Profile»
      «Ispexec Setmsg Msg(Edtm009F)»
      Exit
   End
Else Do
   «Ispexec Select Pgm(IEBCOPY)»
   Rc6 = Rc
   If Rc6 = 0 Then Do
      «Ispexec Vput Rc6 Profile»
      «Ispexec Setmsg Msg(Edtm009H)»
      Exit
   End
/* ———————————————————————————————————————————————————————————————————*/
/* Free all Ddnames of Iebcopy. */
/* ———————————————————————————————————————————————————————————————————*/
«Free File(Input,Output,Sysin,Sysprint,Sysut1,Sysut2,Sysut3,Sysut4)»
Exit
/* End-of_procedure */

EDIT MACRO : CHGMAC

«Isredit Macro»
/* ———————————————————————————————————————————————————————————————————*/
/* Edit macro    : ChgMac */
/* Called from   : ChgRexx */
/* Purpose       : Change strings in a member of a given dataset. */

Status = Msg('Off')
«Ispexec Control Errors Return» /* Let EXEC process errors */
Chg = Ø; Err = Ø /* Change-count, Error-count = Ø */

/* Get the Number Mode. */
/* Edit-macro has to take into account that if Number_Mode is 'ON' */
/* then the number of editable characters is Lrec1-8. Because right- */
/* end side will include «sequence numbers». (NUMBER FIELDS) */
/* So if the user enters a Column2 value that is bigger than */
/* Lrec1-8, It is modified by this edit macro so that we will */
/* prevent errors. */

«Isredit (Mode) = Number»

/* Get the variables from the Profile Variable Pool. These are the */
/* strings which will be used on the 'Ispf Change' edit command. */

«Ispexec Vget (From,To,Mem,Qual,Col1,Col2,L2) Profile»
If ((Mode = ON) & (Col2>(L2-8))) Then Col2 = L2-8

/* Build the limiting keyword. */
Select
  When Qual = 1  Then Qual = CHARS
  When Qual = 2  Then Qual = PREFIX
  When Qual = 3  Then Qual = SUFFIX
  When Qual = 4  Then Qual = WORD
  Otherwise      Nop
End

If Length(From) = 1 Then
  «Isredit Change» «'«From»'» To «ALL» Qual Col1 Col2
Else
  «Isredit Change» From To «ALL» Qual Col1 Col2
Rc2 = Rc
End

«Isredit (Chg,Err) = CHANGE_COUNTS»
Chg = Abs(Chg)
Err = Abs(Err)
If Rc2 = 4 Then Msg2='The string is not found.'
If Rc2 = 8 Then Msg2='Str2 is longer than Str1.'
If Rc2 =12 Then Msg2='Inconsistent parameters.'
If Rc2 =20 Then Msg2='Severe error.'
If Rc2 = Ø Then Do
  Msg2='Change macro command Rc=Ø.'
End

/* If Error-Count is Ø, then we can save the current member in the SOURCE dataset. */
If Err= Ø Then Do
    "Isredit Save"
    Rc3 = Rc /* Rc3 = Save Return Code */
End

"Ispexec Vput (Chg,Err,Msg2,Rc2,Rc3) Profile"

/* No matter it was saved or not, exit from the member to be able to process the next member in the SOURCE dataset. */
"Isredit Cancel"
Return

EDIT MACRO : BATCHMAC
"Isredit Macro"
"Isredit Change V312 V410 ALL"
"Isredit Change '.LIB312' '.LIB410' ALL"
"Isredit Save"
"Isredit Cancel"

ISPF PANEL : PANEL0
)
  )ATTR
    < TYPE(INPUT) INTENS(HIGH) COLOR(YELLOW) CAPS(OFF)
    ( TYPE(INPUT) INTENS(HIGH) COLOR(GREEN) CAPS(OFF)
    ) TYPE(TEXT) INTENS(HIGH) COLOR(PINK)
    $ TYPE(TEXT) INTENS(HIGH) COLOR(PINK)
    % TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
    + TYPE(TEXT) INTENS(Low)
    _ TYPE(INPUT) INTENS(HIGH)
    # TYPE(INPUT) INTENS(HIGH) COLOR(WHITE)
    [ TYPE(TEXT) INTENS(HIGH) COLOR(RED) HILITE(REVERSE)

)BODY
%——————————[ AUTOMATED LIBRARY UPDATE %——————————
%
% Please enter the dataset name on which you will make changes:
+ Dataset %===>_Dsn %
#Mes
%
% Please enter a «FROM» and a «TO» string: (Without quotation marks)
+ From string %===>{Z
%
% To string %===><Z

% Please qualify the search string : (Default is «CHARS»)
%(<Z%) $1-CHARS  2-PREFIX  3-SUFFIX  4-WORD %
% Please enter the Column Limitations : (Optional)
+ Column-1 %===><Z  + Column-2 %===><Z
)
)Hit%ENTER}to proceed. Hit%PF3)0%PF04)key to exit from the application.
)INIT
  .CURSOR = DSN
  .ZVARS = '( From To Qual Col1 Col2)'
  &Qual = '1'
)REINIT
  Refresh(Mes)
)
)PROC
&MES = '
&SPFKEY = .PFKEY
  Ver (&Dsn,Nonblank,Msg=EDTM000)
  Ver (&Dsn,Dsname,Msg=EDTM003)

  &A = Trunc(&Dsn,1)
  Ver (&A,Pict,'A')

  Ver (&From,Nonblank,Msm=EDTM001)
  Ver (&To,Nonblank,Msg=EDTM002)
  Ver (&Qual,Nonblank,Msg=EDTM004)
  Ver (&Qual,List,1,2,3,4,Msg=EDTM005)
  Ver (&Col1,Num,Msg=EDTM007)
  Ver (&Col2,Num,Msg=EDTM007)

  IF (&Col1 > &Col2 & &Col2 NE '')
    .Msg = EDTM009K
  Vput(Spfkey From To Dsn Qual Col1 Col2 ) Profile
)END

ISPF PANEL : PANEL1
)ATTR
  % TYPE(TEXT)  COLOR(WHITE)  CAPS(OFF)  HILITE(USCORE)  INTENS(LOW)
  @ TYPE(TEXT)  COLOR(RED)   CAPS(OFF)  HILITE(USCORE)
  + TYPE(TEXT)  COLOR(TURQ)  CAPS(OFF)  JUST(LEFT)
$ TYPE(TEXT)  COLOR(GREEN) HILITE(REVERSE)
)BODY WINDOW(65,9)
+
$           ****  ATTENTION!  ****                     +
+                                                             +
+ You are about the browse the listing of the comparison of +
+ SOURCE and BKPCPY datasets. Please check the changes that +
+ have been made on the SOURCE dataset. After browsing, you +
+ will be presented with another panel which asks if you agree +
+ with the changes.                                          +
%       Hit@<Enter>%to continue.                +
)INIT
)PROC
&SPFKEY = .PFKEY
   Vput Spfkey Profile
)END

ISPF PANEL : PANEL2

)ATTR
  % TYPE(TEXT)  COLOR(WHITE) CAPS(OFF) HILITE(USCORE) INTENS(LOW)
  @ TYPE(TEXT)  COLOR(RED)   CAPS(OFF) HILITE(USCORE)   +
  $ TYPE(TEXT)  COLOR(TURQ)  CAPS(OFF) JUST(LEFT)        +
  $ TYPE(TEXT)  COLOR(GREEN) HILITE(REVERSE)            +
  [ TYPE(INPUT) COLOR(PINK)  CAPS(ON)  HILITE(REVERSE)    +
)BODY WINDOW(32,9)
+
$   DO YOU ACCEPT THE
$   CHANGES ON THE
$   SOURCE DATASET?
+
+   Yes(YS) / No(N)......:[Z+ +
+ % Hit@<Enter>%to proceed.+     
)INIT
  .Zvars = '( Resp )'
  &Resp = 'N'
  .Cursor = Resp
)PROC
&Spfkey = .Pfkey
Vput Spfkey Profile
Ver (&Resp,Nonblank,Msg=EDTMØØ9L)
Ver (&Resp,List,Y,N,Msg=EDTMØØ9M)
Vput (Resp) Profile
)END

ISPF PANEL : PANEL3

)ATTR DEFAULT(%+_)  
   @ TYPE(OUTPUT) INTENS(HIGH) JUST(ASIS) COLOR(GREEN)
%TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
$ TYPE(OUTPUT) INTENS(HIGH) JUST(ASIS) COLOR(YELLOW) CAPS(OFF)
* TYPE(TEXT) INTENS(HIGH) JUST(ASIS) COLOR(RED)
# TYPE(TEXT) INTENS(HIGH) JUST(ASIS) COLOR(WHITE)
[ TYPE(TEXT) INTENS(HIGH) COLOR(GREEN) HILITE(REVERSE)

)BODY
%——————————[ AUTOMATED LIBRARY UPDATE %——————————
%
] EXECUTION RESULTS %
+
%Command ===>_ZCMD
+
* DATASET :$DSN
:+&ZDATE
* MEMBER-COUNT :$CNT*
:+&ZUSER
* FROM :$FROM
* TO :$TO
* COL1 & COL2 :$COL1  *$COL2  
+ 
* NOTE:#R2 = Isredit Change command Rc.  R1 = Ispexec Edit command Rc.
+      Hit#<F1>+to get more info on Return Codes.
+
+%MEMBER %CHANGE%ERROR%MESSAGE1 %RC%RC%MESSAGE2
+%NAME %COUNT %COUNT% %1 %2 %
+%—%——%——%———————— %—%—%———————————
+
)MODEL
@MEM  @CHG  @ERR  @MSG1  @Z @Z  @MSG2
)INIT
&Zcmd =''
.Help = Panelh
.Zvars = '(Rc1 Rc2)'
)PROC
)END

ISPF PANEL : PANELH

)ATTR DEFAULT(%+_)
% TYPE(TEXT) INTENS(HIGH) COLOR(GREEN)
< TYPE(TEXT) INTENS(HIGH) COLOR(PINK) HILITE(REVERSE)
* TYPE(TEXT) INTENS(HIGH) JUST(ASIS) COLOR(RED)
[ TYPE(TEXT) INTENS(HIGH) COLOR(GREEN) HILITE(REVERSE)

)BODY
%——————————[ AUTOMATED LIBRARY UPDATE %——————————
%
*  Rc1 Message1  («Ispexec Edit Dataset» command Return Codes)
*
+  Ø  Member updated.
+  4  Member not saved. / No save. Abend S37'
+  14  Member in use.

+ 2Ø Severe error.
+ < NOTE : In case you see "S37" on the Message1 text, it's necessary
+      +to compress or enlarge the SOURCE dataset.
+ +
+ * Rc2 Message2 ("Isredit Change" command Return Codes)
+ -------------------------------------------------------------
+ Ø Change macro command normal completion.
+ 4 The string is not found.
+ 8 String-2 is longer than String-1 and
+ substitution was not performed on at least one change.
+ 12 Inconsistent parameters. The string to be found does
+ not fit between the specified columns.
+ 2Ø Severe error.
+
+ )INIT
+ )PROC
+ )END

ISPF MESSAGES : EDTM00

* EDTM00 MESSAGE DEFINITIONS
*
* EDTM00 'Enter a dataset name. ' .HELP=* .ALARM=YES
'Enter a dataset in which you’d like to run an edit macro for its each member'
* EDTM001 'Enter a FROM string. ' .HELP=* .ALARM=YES
'Enter a FROM string that will be used for the 'ISPF Change' Edit command.'
* EDTM002 'Enter a TO string. ' .HELP=* .ALARM=YES
'Enter a TO string that will be used for the 'ISPF Change' Edit command.'
* EDTM003 'Invalid Dset-qualifier.' .HELP=* .ALARM=YES
'Each qualifier must be 1-8 alphanumeric chars & first one must be alphabetic.'
* EDTM004 'Enter 1, 2, 3, or 4.' .HELP=* .ALARM=YES
'Enter a number between 1 and 4 which corresponds to the qualifying keyword.'
* EDTM005 'Out of range.' .HELP=* .ALARM=YES
'You should enter a value between 1 and 4 for this field.'
* EDTM006 'No changes are made. ' .HELP=* .ALARM=YES
'No any changes are made in the specified SOURCE dataset.'
* EDTM007 'Enter a numeric value. ' .HELP=* .ALARM=YES
'Column limitation that will be used for ''CHANGE'' command, must be numeric.'
* EDTM008 '<FROM> string > Lrecl. ' .HELP=* .ALARM=YES
'<FROM> string (&L1) can not be longer than the dataset's Record Length (&L2)'
EDTMOØ9 ' string > Lrecl.' .HELP=* .ALARME=YS
'<string> string (\&L1) cannot be longer than the dataset's Record Length (\&L2)'

EDTMOØ9A 'Hit <PF03+Enter> ' .HELP=* .ALARME=YS
'Hit <Enter> then <PF03> to exit from the Comparison output.'

EDTMOØ9B 'PO dset has no member.' .HELP=* .ALARME=YS
'The PO dset you entered has no member. Please enter it again.'

EDTMOØ9C 'Dset does not exist.' .HELP=* .ALARME=YS
'The PO dset you entered does not exist. Please enter it again.'

EDTMOØ9D 'Dataset is not PO.' .HELP=** .ALARME=YS
'The PO dset you entered is not a PO. Please enter it again.'

EDTMOØ9E 'Lminit Rc = \&Rc5 ' .HELP=* .ALARME=YS
'1st Lminit is not successful.'

EDTMOØ9F 'Lmfree error.' .HELP=* .ALARME=YS
'Lmfree error. Rc = \&Rc4'

EDTMOØ9G 'Lminit Rc = \&Rc7 ' .HELP=* .ALARME=YS
'2nd Lminit is not successful.'

EDTMOØ9H 'Iebcopy error.' .HELP=* .ALARME=YS
'IEBCOPY Return Code = \&Rc6'

EDTMOØ9I 'No change was made.' .HELP=* .ALARME=YS
'The dataset was not updated, since there was no string change.'

EDTMOØ9J 'Col value is incorrect.' .HELP=* .ALARME=YS
'Col1 or Col2 can't be bigger than the Lrecl of the SOURCE dataset.'

EDTMOØ9K 'Col2 must be bigger.' .HELP=* .ALARME=YS
'Col2 value must be bigger than Col1 value. Please enter it again.'

EDTMOØ9L 'Enter Y or N. ' .HELP=* .ALARME=YS
'This field must be either Y or N.'

EDTMOØ9M 'Enter Y to accept. ' .HELP=* .ALARME=YS
'Enter "Y" to accept changes on SOURCE dset and delete BKPCPY dset.'

EDTMOØ9N 'BKPCPY dset kept.' .HELP=* .ALARME=YS
'BKPCPY dataset is kept. SOURCE dataset has now changes.'

EDTMOØ9O 'BKPCPY dset deleted.' .HELP=* .ALARME=YS
'BKPCPY dataset is deleted. SOURCE dataset has now changes.'

Systems Programmer (UK) © Xephon 2001
Compressing contiguous characters in a file

There are many files that contain a large number of spaces. Program source codes, output listings, and others have a large percentage of contiguous spaces. For transmission purposes, archiving, or other reasons, one might want to compress them to reduce their size.

The two programs shown below do just that. The first, ZIPCHAR, takes a contiguous sequence of a given character and reduces it to a single byte. The reverse program, UNZIPCHA, does the opposite operation and restores the file. The character is the space by default, but you can choose any other character simply by passing it as a parameter to ZIPCHAR. For example, low-value can also be a good candidate for compression, in certain files. If the character you want to compress is not displayable, as is the case with low-value, then you must edit your JCL in hexadecimal to insert it correctly as a parameter.

The technique is simple. The program reads an input file sequentially, as a continuous stream of bytes, and produces an output file. If there are three or more contiguous spaces (or whatever character you requested), then it replaces them by a control byte whose lower seven bits represent the number of contiguous spaces, up to 127. The high-order bit acts as a flag, and in this case it is zero. If there are more than 127 contiguous spaces, a second control byte is used to count for the additional ones, and so on.

All the other characters in the file are preceded by a control byte whose high-order bit is one and whose remaining seven bits represent the number of ‘literal’ characters that follow. If there are more than 127 non-space characters, then another control byte is used, and so on. This count does not include the control byte itself. When the input file ends, a zero control byte is written, to mark the end of it; in fact, a zero fullword. The output file is always a fixed 80-byte file. The input file must be a non-VSAM fixed-length file, with any LRECL up to 32KB.

It is your responsibility to know the correct LRECL of the original file when you uncompress it. The compressed file has no information whatsoever about the original record length or where each record ends; it only contains a stream of bytes. The first byte of the compressed file is the character that was compressed. This way the
uncompress program always knows what character to expand, so it needs no parameter.

You can compress an already compressed file for a different character. For example, you can compress a file for spaces and then for zeros. In this case, compress first for the character that occurs more, and then for the other(s), to achieve a better compression ratio.

Below is an example job for a double compression. The input file is assumed to have a record length of 2,500. The first step is for the space, the default, so no parameter is needed. The second step is for character zero, assuming the input file has a lot of contiguous spaces and zeros. The expansion job is also shown. Note that in this case no PARM is needed.

//JOB11 JOB REGION=1000K,MSGCLASS=X,MSGLEVEL=(1,1)
/*@*/
/*@ Example for space and "Ø" compression */
/*@*/
//JOBLIB DD DISP=SHR,DSN=my.loadlib
//STEP1 EXEC PGM=ZIPCHAR
//SYSPRINT DD SYSPRINT DD SYSOUT=* 
//INFILE DD DISP=SHR,DSN=my.input.file
//OUTFILE DD DSN=&TEMP1,DISP=(NEW,PASS),LRECL=80,RECFM=FB,
// BLKSIZE=27200,SPACE=(TRK,(300,300),RLSE),UNIT=SYSDA
/*@*/
//STEP2 EXEC PGM=ZIPCHAR,PARM='Ø'
//SYSPRINT DD SYSPRINT DD SYSOUT=* 
//INFILE DD DISP=(OLD,DELETE),DSN=&TEMP1
//OUTFILE DD DSN=my.compressed.file,DISP=(NEW,CATLG,DELETE),
// SPACE=(TRK,(200,200),RLSE),UNIT=SYSDA,
// LRECL=80,RECFM=FB,BLKSIZE=27200
/*@*/

//JOB22 JOB REGION=1000K,MSGCLASS=X,MSGLEVEL=(1,1)
/*@*/
/*@ Example for restoring a twice-compressed file */
/*@*/
//JOBLIB DD DISP=SHR,DSN=my.loadlib
//STEP1 EXEC PGM=UNZIPCHA
//SYSPRINT DD SYSPRINT DD SYSOUT=* 
//INFILE DD DISP=SHR,DSN=my.compressed.file
//OUTFILE DD DSN=&TEMP1,DISP=(NEW,PASS),LRECL=80,RECFM=FB,
// BLKSIZE=27200,SPACE=(TRK,(300,300),RLSE),UNIT=SYSDA
/*@*/
//STEP2 EXEC PGM=UNZIPCHA
//SYSPRINT DD SYSPRINT DD SYSOUT=* 
//INFILE DD DISP=(OLD,DELETE),DSN=&TEMP1
ZIPCHAR SOURCE

*ZIPCHAR - Reads an INPUT file and writes a compressed OUTPUT file *
* The char indicated by "&CHAR" is compressed, by default *
* Optional parameter: char to compress. The compressed char is written in the first byte of the output file. *
* Decompress program: UNZIPCHR. *

&PROGRAM SETC 'ZIPCHAR'
&CHAR SETC '40'     Hexadecimal of default char
&PROGRAM AMODE 31
&PROGRAM RMODE 24
&PROGRAM CSECT
    SAVE (14,12)
    LR  R12,R15
    USING &PROGRAM,R12
    USING IHADCB,R11
    ST  R13,SAVEA+4
    LA  R11,SAVEA
    ST  R11,8(R13)
    LR  R13,R11
    B   GETPARM
    DC  CL16' &PROGRAM 1.6'
    DC  CL8'&SYSDATE'

*GETPARM DS ØF
    LR  R2,R1
    OPEN (SYSPRINT,OUTPUT)
    L   R2,Ø(Ø,R2)
    L   R3,Ø(R2)
    LR  R9,R3
    SRL R9,16                Get parm length in R9
    LTR R9,R9                 Any parm?
    BZ  OPENFILE               No
    SRL R3,8                  Parm in lower byte of R3
    STC R3,COMPRE              Store it
    CLI COMPRE,X'&CHAR'       Is parm the default character?
    BE  OPENFILE               Yes
    MVC CHANGE1+1(1),COMPRE   No, change CLI instructions.
    MVC CHANGE2+1(1),COMPRE
    MVC CHANGE3+1(1),COMPRE
    MVC CHANGE4+1(1),COMPRE

*OPENFILE EQU *
    SR  R7,R7
    OPEN (INFILE,INPUT)
LTR  R15,R15
BNZ  ERRO1
LA  R11,INFILE          Address IHADCB of input file.
LH  R10,DCBLRECL        Get maximum length
LA  R10,4(Ø,R10)        Keep maximum length plus 4
ST  R10,MAXLRECL
OPEN (OUTFILE,OUTPUT)
LTR  R15,R15
BNZ  ERRO2

* STORAGE OBTAIN,                                     X
LENGTH=68000,                                         X
ADDR=(R2)
ST  R2,POINTER1
A  R2,=F'32768'                                      Just one 32k record as limit
ST  R2,POINTER2

* LA  R5,OUTBUF1          R5: Address output buffer
L  R3,POINTER1         R3: Current byte at inbuffer
XR  R4,R4               R4: Bytes available at inbuffer
XR  R6,R6               R6: Number chars in out buffer
XR  R8,R8               R8: Char or space counter
MVC  Ø(1,R5),COMPRE     Put compressed char in outbuf
LA  R5,1(Ø,R5)          Inc output pointer
LA  R6,1(Ø,R6)          Inc output counter
BAL  R10,READIN         Go read some records to inbuffer

* Paragraphe Bnn deal with compressed char, Cnn deal with the others

C000  EQU *          Deal with characters:
BAL  R10,WSPACES       Write spaces first
C00  EQU *          If next three bytes are spaces,
CHANGE1 CLI  Ø(R3),X'&CHAR' branch to spaces.
          BNE  C01        If not, consider them chars
CHANGE2 CLI  1(R3),X'&CHAR'
          BNE  C01
CHANGE3 CLI  2(R3),X'&CHAR'
          BE  B000
C01  EQU *
CH  R8,=H'127'        Counter limit attained?
BL  C02             No, jump
BAL  R10,WCHARS      Yes, write out chars
C02  EQU *
LA  R8,1(Ø,R8)       Inc char number
LA  R3,1(Ø,R3)       Inc current byte pointer
SH  R4,=H'1'         Dec input buffer still available
LTR  R4,R4           End of buffer and of file?
BE  ENDCCHARS        Yes, go deal with what is left.
CH  R4,=H'130'       Input buffer nearing 127?
BH  C03              No
BAL  R10,READIN      Yes, read more records
C03  EQU *
B     C00  Continue

B000  EQU  *  Deal with spaces:
        BAL  R10,WCHARS  write characters first

B00  EQU  *

CHANGE4  CLI  0(R3),X'&CHAR'  Current byte space?
        BNE  C000  No, jump to chars
        CH  R8,=H'127'  Counter limit attained?
        BL  B02  No
        BAL  R10,WSpaces  Yes, write spaces

B02  EQU  *
        LA  R8,1(0,R8)  Inc space counter
        LA  R3,1(0,R3)  Inc current byte
        SH  R4,=H'1'  Dec available buffer
        LTR  R4,R4  End of buffer and file?
        BE  ENDSpace  Yes, go deal with what is left
        CH  R4,=H'130'  Input buffer nearing 127?
        BH  B03  No
        BAL  R10,READIN  Yes, read more records

B03  EQU  *

*====================================================================*
*  After input file ended, write out what's left and exit
*====================================================================*

ENDSPACE  EQU  *
        BAL  R10,WSpaces  Put remaining spaces
        B  ENDALL

ENDCHARS  EQU  *
        BAL  R10,WCHARS  Put remaining chars in buffer
        B  ENDALL

*  ENDALL  EQU  *
        XR  R1,R1  Put a zero fullword as eof mark
        ST  R1,0(R5)  Add 4 bytes to output
        BAL  R8,WRITEOUT  Write last buffer or two
        B  EXIT1  and exit

*  EXIT1  EQU  *
        L  R2,POINTER1
        STORAGE  RELEASE,  X
        LENGTH=68000,  X
        ADDR=(R2)
        CLOSE INFILE
        CLOSE OUTFILE
        CLOSE SYSPRINT
        L  R13,SAVEA+4
        LM  R14,R12,12(R13)
        XR  R15,R15
        BR  R14

*====================================================================*
* Subroutines

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
WCHARS EQU * Write out (R8) chars
LTR R8,R8 Zero chars?
BZR R1Ø Yes, go back
STCM R8,B'0001',Ø(R5) Store length
OI Ø(R5),X'80' First bit on indicates chars.
LA R5,1(Ø,R5) Inc output pointer
LA R6,1(Ø,R6) Inc output counter
SR R3,R8 R3 points chars to move
SH R8,=H'1'
EX R8,MVC2 move chars
LA R8,1(Ø,R8) restore length
AR R3,R8 Restore current pointer
AR R5,R8 Advance output pointer
AR R6,R8 R6 number of bytes in outbuf
CH R6,=H'80' record complete?
BL WCHARS1 Not yet
BAL R8,WRITEOUT Yes, write out file

WCHARS1 EQU *
XR R8,R8 Reset zero length
BR R1Ø Return

WSPACES EQU * Write out (R8) spaces
LTR R8,R8 If zero, return
BZR R1Ø Return
STCM R8,B'0001',Ø(R5) write lower byte of R8
LA R5,1(Ø,R5) Inc out pointer
LA R6,1(Ø,R6) Inc number of bytes in outbuf
XR R8,R8 Reset zero length
CH R6,=H'80' record complete?
BL WSPACES1 Not yet
BAL R8,WRITEOUT Yes, write out file

WSPACES1 EQU *
XR R8,R8 Reset zero length
BR R1Ø Return

READIN EQU * Read input file subroutine
CLI ENDINPUT,C'1' Input file ended?
BER R1Ø Yes, return
LR R9,R3 Current input location
AR R9,R4 Add remaining input for next free.
L R2,POINTER2 End of read area
SR R2,R9 R2 = Remaining read area
CH R2,MAXLRECL Space for another record?
BH READIN1 Yes, jump around move
* Move data to beginning of area:
L R9,POINTER1 Point beginning of area
LR R7,R3 Current byte
SR R7,R8 R7 initial byte to move
SH R8,=H'1' Prepare move length
EX R8,MVC1 Move data before R3
LA R8,1(Ø,R8) Restore length
AR R9,R8 Point after data moved
LR R7,R3               Move data starting at R3
SH R4,=H'1'            Prepare move length (R4)
EX R4,MVC1             Move data after R3
LA R4,1(R0,R4)         Restore length
L R3,POINTER1          Beginning of area
AR R3,R8               R3: new current location
B READIN

READIN1 EQU *          Read input file
GET INFILE,(R9)
AH R4,DCBRECL         Add record leng to R4 total leng
AH R9,DCBRECL         Add record leng to R9 pointer
SH R2,DCBRECL         And subtract from space left (R2)
CH R2,MAXRECL         Space for another record?
BH READIN1            Yes, jump around move

READEND EQU *          Return
BR R10

WRITEOUT EQU *         Write output file
PUT OUTFILE,OUTBUF1    Write buffer1
C R6,=F'160'           Second buffer filled?
BL WRITEOU1           No, jump ahead
PUT OUTFILE,OUTBUF2    Write buffer 2
MVC OUTBUF1,OUTBUF3    Move back buffer 3
S R6,=F'160'           from length and pointer
LA R5,OUTBUF1
AR R5,R6
BR R8

WRITEOU1 EQU *         Return
MVC OUTBUF1,OUTBUF2    Move back buffer 2
S R6,=F'80'            Subtract 80 bytes
LA R5,OUTBUF1
AR R5,R6
BR R8

* ERROR1 EQU *         Error opening input file'
PUT SYSPRINT,=CL80'>>> Error opening input file'
B EXIT1

* ERROR2 EQU *         Error opening output file'
PUT SYSPRINT,=CL80'>>> Error opening output file'
B EXIT1

* ENDFILE EQU *         Return of read subroutine
MVI ENDFILE,C'1'       Set eof flag
B READEND

* INFILE DCB DSORG=PS,MACRF=(GM),
EODAD=ENDFILE,
DDNAME=INFILE

* OUTFILE DCB DSORG=PS,MACRF=(PM),
LRECL=80,
DDNAME=OUTFILE

* SYSPRINT DCB DSORG=PS,MACRF=(PM),
UNZIPCHA SOURCE

*====================================================================*
* UNZIPCHA - Expands files compressed by ZIPCHAR. Reads an INPUT      *
* compressed 80 byte file and writes an OUTPUT.                     *
* Parameter: none.                                                  *
*====================================================================*

&PROGRAM SETC 'UNZIPCHA'
&CHAR SETC '40'
&PROGRAM AMODE 31
&PROGRAM RMODE 24
&PROGRAM CSECT

SAVE (14,12)
LR R12,R15
USING &PROGRAM,R12
USING IHADCB,R11
ST R13,SAVEA+4
LA R11,SAVEA
ST R11,B(R13)
LR R13,R11
B OPENFILS
DC CL16' &PROGRAM 1.6'
DC CL8'&SYSDATE'

* OPENFILS DS 0F
OPEN (SYSPRINT,OUTPUT)
OPEN (INFILE,INPUT)
LTR R15,R15
BNZ ERRO1
OPEN (OUTFILE,OUTPUT)
LTR R15,R15
BNZ ERRO2
LA R11,OUTFILE Address IHADCB of input file.
MVC MAXLRECL,DCBLRECL Keep maximum length

* STORAGE OBTAIN, Enough for 32k X
LENGTH=33000, X
ADDR=(R2)
ST R2,POINTER1
L R3,POINTER1 R3: current output pointer
XR R4,R4 R4: Bytes at outbuf
XR R6,R6 R6: Bytes at input
BAL R10,READIN Read first record
MVC COMPRE,0(R5) First byte is the compressed char
LA R5,1(0,R5) Inc input pointer
SH R6,=H'1' Decrease input counter
CLI COMPRE,X'&CHAR' Default char?
BE TESTBYTE Yes
MVC CHANGE1+1(1),COMPRE No, change MVI instruction

* TESTBYTE EQU *
MVC BYTE,0(R5) Get first byte
LA R5,1(0,R5) Advance input pointer
SH R6,=H'1' Decrease input counter
CLI BYTE,X'00' Byte is null?
BE EXIT1 Yes, exit
CLI BYTE,X'80' High-order bit 1?
BL SPACES No

* BYTES EQU *
NI BYTE,X'7F' Reduce by 128
XR R7,R7 Clear R7
IC R7,BYTE Get value at byte

* BYTES1 EQU *
CR R7,R6 More chars than left in record?
BNH BYTES2 No
SH R6,=H'1' Decrease for execute
EX R6,MVC1 Move R6 characters
LA R6,1(0,R6) Restore
AR R4,R6 Inc output chars (R4)
AR R3,R6 Inc output pointer (r3)
SR R7,R6 Length that remains yet
BAL R10,READIN Go read more
B BYTES1

* BYTES2 EQU *
SH R7,=H'1' Decrease for execute
EX R7,MVC1 Move R7 characters
LA R7,1(0,R7) Restore
AR R3,R7 Inc output pointer (r3)
AR R4,R7 Inc output chars (R4)
AR   R5,R7             Inc input pointer
SR   R6,R7             Decrease input
B    BYTEEEND
*
SPACES EQU *
XR   R7,R7              Clear R7
IC   R7,BYTE            Get number of copies at byte
*
SPACES1 EQU *
CHANGE1 MVI Ø(R3),X'&CHAR'     Put compressed char
LA   R3,1(Ø,R3)          Inc out pointer
LA   R4,1(Ø,R4)          Inc out counter
BCT  R7,SPACES1          Loop for number of spaces
*
BYTEEND EQU *
CR   R5,R8              Input record ended?
BL   BYTEEEND1           No
BAL   R1Ø,READIN         Yes, read more
*
BYTEEEND1 EQU *
BAL   R1Ø,TESTOUT        Test for output record
B    TESTBYTE
*
EXIT1 EQU *
L    R2,POINTER1
STORAGE RELEASE,                                              X
LENGTH=33000,                                           X
ADDR=(R2)
CLOSE INFILE
CLOSE OUTFILE
CLOSE SYSPRINT
L    R13,SAVEA+4
LM   R14,R12,12(R13)
XR   R15,R15
BR   R14
*====================================================================*
*        Subroutines
*====================================================================*
TESTOUT EQU *
CH   R4,DCBLRECL        Out record complete?
BLR  R1Ø                Not yet, return
PUT   OUTFILE,(R2)       Record complete, write it
SH   R4,DCBLRECL        See what's left
LTR  R4,R4              Anything?
BZ   TESTOUT1           No, return
SR   R3,R4              Move pointer back
SH   R4,=H'1'
EX   R4,MVC2            Move extra to beginning of new rec
LA   R4,1(Ø,R4)
*
TESTOUT1 EQU *
LR   R3,R2             Restore current pointer
AR R3,R4 Add excess, if any
B TESTOUT and go see if there are more

* READIN EQU * Read input
CLI ENDINPUT,C'1' File ended?
BER R1Ø yes, return
GET INFILE get locate
LR R5,R1 R5 input pointer
LA R8,8Ø(Ø,R5) R8 End of input record
LA R6,8Ø R6 8Ø bytes available
BR R1Ø return

ERR01 EQU *
PUT SYSPRINT,=CL8Ø'>>> Error opening input file'
B EXIT1

ERR02 EQU *
PUT SYSPRINT,=CL8Ø'>>> Error opening output file'
B EXIT1

ENDFILE EQU *
MVI ENDINPUT,C'1' Set eof flag
BR R1Ø

INFILE DCB DSORG=PS,MACRF=(GL),
EODAD=ENDFILE,
DDNAME=INFILE

OUTFILE DCB DSORG=PS,MACRF=(PM),
DDNAME=OUTFILE

SYSPRINT DCB DSORG=PS,MACRF=(PM),
LRECL=8Ø,
DDNAME=SYSPRINT

LTORG
DS ØF
MVC1 MVC Ø(Ø,R3),Ø(R5)
MVC2 MVC Ø(Ø,R2),Ø(R3)
BYTE DS C
ENDINPUT DC C'Ø'
COMPRE DS C
SAVEA DS 18F
POINTER1 DS F
POINTER2 DS F
MAXLRECL DS H
DCBD DSORG=PS
YREGS
END

Systems Programmer (UK) © Xephon 2001
Using REXX and the Web without OpenEdition

In *MVS Update* issues 168 and 169 (September and October 2000) I provided some code that allowed REXX to talk LPAR to LPAR (or machine) across TCP/IP using socket calls. While I was developing that function it had struck me that it would be nice to get the same function operating over the Web. Unfortunately I did not know anything about OpenEdition, and I had always assumed that this was required to achieve any form of Web development. Plus I did not have a clue about how to get from the Web to my server address space anyway. Then while reading an Infoman manual I came across a reference to setting up the Infoman Web server which happened to explain how to code the HTTP request to contact a particular server. Apparently this involves specifying the IP address of the relevant host followed by a ':' and the port number of the server address space. Out of curiosity I tried using this technique with the LPARANSR routine of the previous article. A connection occurred, but only ‘rubbish’ was returned. However, as it was clear that a connection was possible, all that was required now was to resolve the issues of EBCDIC to ASCII conversion, and how to create HTML for shipping down to a PC and for it to be possible to have a Web server without needing O/E and HFS files. This article shows the results of that development, which has resulted in a Web dialog for carrying out a number of functions such as TSO commands, DISKSPACE displays, and TAPE displays. Note that if you wish to implement this code with all functions active, then you will need the RVOLDATA program from the previous article for the DISKSPACE command. Furthermore, if you are Web-naive, as I was when starting this development, then it is worth reviewing that article for some other considerations to watch for when starting the REXX server.

**OPERATION**

In order to get this server working you will need to start the REXX code as a batch TSO command where the SYSPROC points to the HTMLREXX code, and where the STEPLIB contains the RVOLDATA, UCBTAPE, and, if necessary, the library containing the RXSOCKET load module (again refer to the previous article). To access the server, simply specify the IP:PORT address. Note that the
REXX code supplied contains a comment regarding the use of a META tag to initiate auto-refresh of the TAPE display screen. If this is re-instated to the code and used in conjunction with the additional PC functionality to provide frames access, then it is possible to have one PC screen which ‘looks’ at multiple servers and keeps updating its screens. Overall such functionality may be of most use if using the supplied code as a means of providing your own Web functionality.

This article includes the following items:

- Sample screens as displayed from the server. Please note that the screens shown in this article are modified to keep my site information confidential, and as such what is shown is produced from a display of the HTML and not from the server. However, the format is identical.

- The HTMLREXX source code (this is the actual server code).

- A sample job for starting the server – note that the previous article also includes similar jobs and it also includes examples of FTP jobs for starting the server on multiple machines/LPARs at one go.

- The source code for the tape information retrieval module (UCBTAPE).

- An example of a PC based HTML routine that can be used to initiate frames access over a browser to pull multiple servers together on one screen (five machines are brought together in the example). Note these are not required for basic PC-to-mainframe communication, they are just a potentially useful add-on.

```
/* REXX */
/*****************************/
/* This is the basic server REXX for developing Web serving routines.*/
/* It is called from the Web and is passed a command as part of a */
/* CGI script in the form /?Command=etc. Where etc. is the command */
/* that is actually required. This command string is then parsed to */
/* determine what is required. */
/* */
/* If the string DISKSPACE is passed then information on the info */
```


/* on the currently on-line DASD is returned. */
/*
/* If the string TAPEINFO is passed then information on the info */
/* on the current tape set-up is returned. */
/*
/* If the string commences with TSO then all that follows this will */
/* be issued as a TSO command and the data trapped and returned to */
/* the caller (eg the results of a LIST UCAT). */
/*
/* If the string SHUTDOWN is passed then the server will terminate */
/*
/* If none of the above is passed then a basic Web page is served to */
/* request a command from a user. */
/*
/* Once the command has been obtained and the CGI bits parsed, the */
/* remaining string will be parsed based on the assumption that it */
/* will be of the following form:
/* label (action) */
/*
/* Where label is the routine to invoke, and action is the argument */
/* to that label. See the SELECT statement later to see how to */
/* implement additional function in this server. */
*****************************************************************************/
/*
SIGNAL ON syntax
/* */
linecount.=Ø
/* initialize control information */
port = '1952' /* The port used for the service */
/* */
/* now obtain the name of the LPAR this server is running on */
/* */
CVTECVT=D2X(C2D(STORAGE(1Ø,4))+14Ø) /* point to cvtsysad */
lparname=STRIP(STORAGE(D2X(C2D(STORAGE(CVTECVT,4))+344),8))
/* Begin setup */
SAY 'RSSERVER: initializing'
/* */
/* a call to scoket will return a string which gives an rcode */
/* followed by the unique name for this task (in this case */
/* RSSERVER) followed by the maximum number of tasks and */
/* finally the name of the IP started task. */
/* */
x= 'SOCKET'('Initialize','RSSERVER')
IF WORD(x,1)¬='Ø' THEN DO
  SAY 'ERROR while initialising'
  EXIT
END
/* */
/* We now need to get the host IP address. This is done with a */
/* gethostid request. In a similar manner to other requests the */
/* first character returned is a success or failure indicator */
/* and in this case the second word is the IP address */
ipaddress='SOCKET'('GetHostId')

IF WORD(ipaddress,1)¬='Ø' THEN DO
   SAY 'ERROR while getting hostid'
   EXIT
END

ipaddress=WORD(ipaddress,2)

SAY 'RSSERVER: initialized: ipaddress='ipaddress 'port='port

/* obtain a socket id. This is word 2 of the request. */
sock = 'SOCKET'('Socket')

IF WORD(sock,1)
   SAY 'ERROR while getting socket'
   EXIT
END

sock=WORD(sock,2)

/* In case IP hasn't cleared itself up by the time the server */
/* restarts, set the reuse option to prevent the server being */
/* unable to start. */

x = 'SOCKET'('SetSockOpt',sock,'Sol_Socket','So_REUSEADDR','On')

/* now its time to issue a bind. Only a single character RC */
/* should be returned this time. */

x='SOCKET'('Bind',sock,'AF_INET' port ipaddress)

IF x¬=Ø THEN DO
   SAY 'error during af_inet'
   EXIT
END

/* now time to listen. */

x='SOCKET'('Listen',sock)

IF x¬=Ø THEN DO
   SAY 'error during listen'
   EXIT
END

/* now set the io control mode to with blocking. */
/ * */  
x = 'SOCKET'('Ioctl', sock, 'FIONBIO', 'ON')  
/* */  
IF x ≠ Ø THEN DO 
   SAY 'error during set of io control mode' 
   EXIT 
END  
/* */  
x = 'SOCKET'('Fcntl', sock, 'F_SETFL', 'BLOCKING')  
/* */  
IF x ≠ Ø THEN DO 
   SAY 'error during set of io control mode' 
   EXIT 
END  
/* */  
/* Wait for new connections and send lines. The array linecount will */ 
/* be used to keep track of data sent to each caller. */  
/* */  
linecount. = Ø  
/* */  
DO FOREVER  
/* */  
sellist = 'SOCKET'('SELECT', 'Write * Read * Exception')  
/* */  
PARSE UPPER VAR sellist . 'READ' rsock . 'WRITE' wsock . 'EXCEPTION' .  
/* */  
/* Now receive the information. If the socket id passed is the same */ 
/* as the one we are listening on, then we need to accept the */ 
/* new connection. */  
/* */  
IF rsock ≠ '' THEN DO 
   IF rsock = sock THEN DO 
      x = 'SOCKET'('Accept', rsock) 
      IF WORD(x,1) ≠ 'Ø' THEN DO 
         SAY 'error adding another socket' 
         EXIT 
      END ELSE rsock = WORD(x,2) 
   END 
/* */  
/* Ensure that the incoming data is converted from ASCII to EBCDIC */ 
/* */  
x = 'SOCKET'('SetSockOpt', rsock, 'Sol_Socket', 'So_ASCII', 'On')  
/* */  
/* Then read the data */  
/* */  
x = 'SOCKET'('Recv', rsock)  
/* */  
PARSE VAR x . user string  
/* */  
IF x ≠ 'Ø' THEN DO 
   SAY 'Connection lost'
x='SOCKET'('Close',rsock)
END
ELSE DO
  stringuser.rsock=user
  stringword.rsock=string
  SAY 'User' user 'issued command' string 'at' TIME() DATE('E')
END
END

/* */
/* Retrieve the command for this socket request and build the */
/* information in the variable array msg.wsock.msgnum.         */
/* It is assumed that RESULT will contain the number of lines  */
/* to return to the caller upon return from the subroutine.    */
/* If it doesn't then 1 line to return is assumed.             */
/* Note that by chaining the data to minimize the number of SENDs*/
/* the server will perform much more effectively. Hence minimize */
/* the array count accordingly.                               */
/* */
/* The array entries are returned one at a time thus allowing the*/
/* linecount for the write socket to drop to zero at which point */
/* the connection is closed.                                   */
/* */
IF wsock='' THEN DO
  IF linecount.wsock=Ø THEN DO
    PARSE VAR stringword.wsock '/?Command=' command .
    UPPER command
    PARSE VAR command command '+' data
    CALL data_fix /* deal with % characters and pluses */
    SELECT
      WHEN command='DISKSPACE' THEN CALL diskspace_process
      WHEN command='TSO' THEN CALL tsocmds_process
      WHEN command='TAPEINFO' THEN CALL tapeinfo_process
      WHEN command='SHUTDOWN' THEN SIGNAL shutdown
      OTHERWISE CALL front_screen
    END
    IF RESULT='' THEN linecount.wsock=1
    ELSE linecount.wsock=RESULT
  END
  linecount.wsock=linecount.wsock
  msg=msg.wsock.msgnum
  x='SOCKET'('SetSockOpt',wsock,'Sol_Socket','So_ASCII','On')
  x='SOCKET'('Send',wsock,msg)
  IF WORD(x,1)='Ø' THEN DO
    linecount.wsock = linecount.wsock - 1
    DROP msg.wsock.msgnum
  END
  IF WORD(x,1)='Ø' THEN DO /* send failure - cleanup */
    linecount.wsock=Ø      /* indicate no lines */
    DO x=1 TO msgnum
      DROP msg.wsock.x      /* release storage */
    END
    DROP stringword.wsock
DROP stringuser.wsock

END

IF linecount.wsock=Ø THEN DO
x='SOCKET'('Close',wsock)
END

END

/* */
/* Terminate the server and exit */
/* */
shutdown:
x='SOCKET'('Terminate')
SAY 'RSSERVER: Terminated'
EXIT Ø
/* */

/* ------------------- The processing subroutines -------------------*/
/* */
diskspace_process:
/* */
CALL RVOLDATA
/* */
/* first pass back the title line. */
/* */
titleline='Address Volser Free_Extents Free_Cyls Free_Trks Large_Cyl',
        'Large_Trk Index Frag'
y=2
msg.wsock.y='<HTML><HEAD><TITLE>DISKSPACE</TITLE><HEAD>',
        '<BODY BGCOLOR="#00ffff" LINK="#0000ff" VLINK="#808080">','
        '<FONT COLOR="#008000">'||,
        '<H1 ALIGN="CENTER">Result of '1parname 'Diskspace Command</H1>'','
        '<TABLE BORDER CELLSPACING=1><TR>
DO x=1 TO WORDS(titleline)
    msg.wsock.y=msg.wsock.y||'<TD VALIGN="MIDDLE"><P><FONT FACE="Arial">',
        ||WORD(titleline,x)||'</FONT></TD>'
    END
    msg.wsock.y=msg.wsock.y||'</TR>'
    y=y-1
    DO x=1 TO volser.Ø*1
        msg.wsock.y=msg.wsock.y||',
        '<TR><TD VALIGN="MIDDLE"><P>',
        address.x||'</TD>',
        'volser.x||'</TD>',
        'free_extents.x||'</TD>',
        'free_cylinders.x||'</TD>',
        (1*free_tracks.x)+(15*free_cylinders.x)||'</TD>',
        'largest_cylinder_extent.x||'</TD>',
        'largest_track_extent.x||'</TD>',
        'index_status.x||'</TD>',
        'fragmentation_index.x||'</TD></TR>'
        END
END

msg.wsock.y='</TR>:y=y-1

msg.wsock.y=''}
**tapeinfo_process:**

/* first pass back the title line. */

```
titleline='Address Device Volume Jobname'
y=2
/* */
```

/* Insert <META HTTP-EQUIV="REFRESH" CONTENT=30> after the HEAD */
/* and before title if you want to get this screen to refresh */
/* itself every 30 seconds. */
/* */

```
msg.wsock.y='<HTML><HEAD><TITLE>TAPE information</TITLE><HEAD>'||,
  '<BODY BGCOLOR="#00ffff" LINK="#008000" VLINK="#800080">'||,
  '<FONT COLOR="#008000">'||,
  '<H1 ALIGN="CENTER">Result of 'lparname 'Tapeinfo Command'</H1>'||,
  '<TABLE BORDER CELLSPACING=1><TR>
```

```
DO x=1 TO WORDS(titleline)
  msg.wsock.y=msg.wsock.y||'<TD VALIGN="MIDDLE"><P><FONT FACE="Arial">',
  ||WORD(titleline,x)||'</FONT></TD>'
END
```

```
msg.wsock.y=msg.wsock.y||'</TR>';y=y-1
```

```
DO x=1 TO volume.0*1
  msg.wsock.y=msg.wsock.y||,
  '<TR><TD VALIGN="MIDDLE"><P>'||address.x||'</TD>',
  '<TD VALIGN="MIDDLE"><P>'||STRIP(unit_type.x)||'</TD>',
  '<TD VALIGN="MIDDLE"><P>'||volume.x||'</TD>',
  '<TD VALIGN="MIDDLE"><P>'||job_name.x||'</TD></TR>
```

```
msg.wsock.y=msg.wsock.y||'</TABLE></HTML>
/* */
```

**tsocmds_process:**

```
ADDRESS TSO
CALL OUTTRAP('LINE.')
'data
/* */
```

/* now build the html */
/* */

```
msg.wsock.1='<HTML><HEAD><TITLE>TSO Command</TITLE><HEAD>'||,
  '<BODY BGCOLOR="#00ffff" LINK="#008000" VLINK="#800080">'||,
  '<FONT COLOR="#008000">'||,
  '<B><FONT FACE="Arial" SIZE=4 COLOR="#ff0000"><P ALIGN="CENTER">'||,
  lparname 'TSO Command: ' data '</P></B><HR></FONT><FONT SIZE=2>''
```
DO x=1 TO line.Ø
msg.wsock.1=msg.wsock.1||line.x '<BR>'
DROP line.x
END
msg.wsock.1=msg.wsock.1||'</FONT></BODY></HTML>'
CALL OUTTRAP('OFF')
RETURN 1
/* */
/* The front screen routine sends back a basic data entry screen */
/* to enable the user to make requests of this server */
/* */
/* front_screen: */
msg.wsock.1='</HTML><HEAD><TITLE>TITLE LINE</TITLE></HEAD>'||
'<BODY BGCOLOR="#00ffff" LINK="#0000ff" VLINK="#000080">'||
'<!-- <!--<FONT COLOR="#008000">| ||
<B><FONT FACE="Arial" SIZE=4 COLOR="#ff0000"><P ALIGN="CENTER">| ||
'ENTER COMMAND FOR 'tparname 'SERVER </P></B></FONT><FONT SIZE=2>'||
'<form name="topicForm">| ||
'Please specify required command <INPUT NAME="Command" SIZE=40>'||
'</select></form></font>'||
'<H2>Available Commands</H2><HR>'||
'<FONT FACE="Courier New"><OL>'||
'<LI>DISKSPACE ......... Requests disk space information'||
'<LI>TAPEINFO .......... Requests Tape Drive Usage'||
'<LI>TSO .............. Issues a TSO command'||
'<LI>SHUTDOWN ........... Closes down the server'||
'</OL></BODY></HTML>'
RETURN 1
/* */
/* data translation routine to clear up the CGI stuff */
/* */
data_fix:
IF INDEX(data,'+')/=Ø THEN data=TRANSLATE(data,' ','+')
/* Now do left brackets */
IF INDEX(data,'%28')/=Ø THEN DO UNTIL INDEX(data,'%28')=Ø
PARSE VAR data data '%28' rem
data=data'('rem
END
IF INDEX(data,'%29')/=Ø THEN DO UNTIL INDEX(data,'%29')=Ø
PARSE VAR data data '%29' rem
data=data')'rem
END
IF INDEX(data,'%27')/=Ø THEN DO UNTIL INDEX(data,'%27')=Ø
PARSE VAR data data '%27' rem
data=data""rem
END
IF INDEX(data,'%23')/=Ø THEN DO UNTIL INDEX(data,'%23')=Ø
PARSE VAR data data '%23' rem
data=data'"rem
END
IF INDEX(data,'%3B')/=Ø THEN DO UNTIL INDEX(data,'%3B')=Ø
PARSE VAR data data '%3B' rem

data=data':'rem
END
IF INDEX(data,'%22')/=Ø THEN DO UNTIL INDEX(data,'%22')=Ø
PARSE VAR data data ' %22' rem
data=data"'rem
END
IF INDEX(data,'%A3')/=Ø THEN DO UNTIL INDEX(data,'%A3')=Ø
PARSE VAR data data ' %A3' rem
data=data'$'rem
END
IF INDEX(data,'%25')/=Ø THEN DO UNTIL INDEX(data,'%25')=Ø
PARSE VAR data data ' %25' rem
data=data%'rem
END
IF INDEX(data,'%2B')/=Ø THEN DO UNTIL INDEX(data,'%2B')=Ø
PARSE VAR data data ' %2B' rem
data=data'+rem
END
IF INDEX(data,'%2C')/=Ø THEN DO UNTIL INDEX(data,'%2C')=Ø
PARSE VAR data data ' %2B' rem
data=data',$rem
END
IF INDEX(data,'%21')/=Ø THEN DO UNTIL INDEX(data,'%21')=Ø
PARSE VAR data data ' %21' rem
data=data!'rem
END
IF INDEX(data,'%3F')/=Ø THEN DO UNTIL INDEX(data,'%3F')=Ø
PARSE VAR data data ' %3F' rem
data=data?'rem
END
IF INDEX(data,'%3D')/=Ø THEN DO UNTIL INDEX(data,'%3D')=Ø
PARSE VAR data data ' %3D' rem
data=data='rem
END
RETURN

SAMPLE START-UP JOB

//your job card
//A EXEC PGM=IKJEFT01,DYNAMNBR=50,REGION=6M
//STEPLIB DD DSN=library.with.ucbtape.and.rvoidata,DISP=SHR
//SYSPROC DD DSN=library.with.htmlrexx.in,DISP=SHR
//SYSTSOUT DD SYSSOUT=* 
//SYSTSPRT DD SYSSOUT=* 
//SYSUDBD DD SYSSOUT=* 
//SYSTSPRT DD * 
PROFILE NOPREFIX 
HTMLREXX

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
UCBTAPE

Note that this routine can be linked into any load library, as it does not require APF authorization. Furthermore this routine is not limited to use in just this server. It can be exploited by any REXX routine wishing to access tape details. See the list of variables that are created in the comments for ideas.

//your job card
//STEA EXEC ASMFCL,PARM.LKED='NORENT,NOREUS'
//ASM.SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
// DD DSN=SYS1.MODGEN,DISP=SHR
//ASM.SYSIN DD *
***********************************************************************
* THIS ROUTINE ANALYSES THE UCBS OF THE DEVICES ON THE SYSTEM AND *
* CREATES A VARIETY OF REXX ARRAY VARIABLES FOR ON-LINE TAPE UNITS *
* THE VARIABLES CREATED ARE AS FOLLOWS: *
* * 
* CHPS ............THE NUMBER OF CHANNEL PATHS ATTACHED TO A DEVICE  *
* RCODE ..........RETURN CODE FORM THE UCBINFO MACRO  *
* UNIT_TYPE ........CONFIRMATION OF DEVICE TYPE USING THE EDTINFO  *
* MACRO.  *
* VOLUME ...........VOLUME SERIAL NUMBER.  *
* ADDRESS ..........DEVICE ADDRESS  *
* JOB_NAME ..........JOB USING THIS TAPE (IF A TAPE ON DRIVE).  *
* PATH_ID.1 ........PATH NUMBER ATTACHED TO THE DEVICE. THIS IS AN  *
* ARRAY DEPTH 8 (MAX NUM OF ATTACHABLE CHANNELS.  *
* PATH_TYPE.1 ......CHARACTER DESCRIPTION OF THE TYPE OF PATH TO WHICH  *
* THE DEVICE IS ATTACHED (ESCON, COPPER ETC.)  *
* * 
* NOTE: THE BASE VARIABLE FOR ALL THESE ITEMS IS VOLUME.Ø  *
* IF INDIVIDUAL, OR OTHER BASES ARE REQUIRED, SIMPLY ADD IN  *
* THE APPROPRIATE SHOWBASE MACROS.  
***********************************************************************
UCBTAPE TITLE 'REXX FUNCTION TO RETRIEVE UCB INFORMATION'
MACRO
REXREGS
LCLA &CNT
&CNT SETA Ø
LOOP ANOP
R&CNT EQU &CNT
&CNT SETA &CNT+1
AIF (&CNT LT 16).LOOP
MEND
MACRO
SHOWSET
AIF (D'SHOW_START').NONEED
B. BY_SHOW_START
SHOW_START DS ØH
ST R1Ø,COMRET
LA 6,COMSHVB
USING SHVBLOCK,R6
XC COMSHVB(SHVBLEN),COMSHVB
XC SHVNEXT,SHVNEXT
MVI SHVCODE,C'S'
BR 14
ABENDØØ1 DS ØH
   ABEND 1 * REQUIRED FOR THE OTHER MACROS. SAVES SOME CODING.
BY_SHOW_START DS ØH
LITLOC LOCTR
@_UNPACK DC CL16' ' *
   DC CL8' ' * FILL FIELD
   ORG @_UNPACK+8
@_UNPACKER DC CL8' '
   ORG
@_DWORD DS CL8 * USED FOR THE DEBIN FUNCTION
&SYSECT LOCTR
.MONEED ANOP
   BAL 14,SHOW_START
MEND
MACRO
SHOWARAY &LABEL,&ASNAME,&ERR=ABENDØØ1,&LEN=,&SUBARRAY=,&DEBIN=,&LINK=
   PRINT NOGEN
***********************************************************************
* MACRO TO CREATE REXX ARRAY VARIABLES
*
* NOTE RESTRICTION: THIS MACRO IS LIMITED TO CREATING UP TO 9,999,999
* ENTRIES FOR EACH ARRAY.
* MACRO FORMAT:
*   SHOWARAY &LABEL,&ASNAME,&ERR=,&LEN=,&SUBARRAY=,&DEBIN=,
* WHERE:
*   &LABEL IS THE NAME OF THE LABEL WHICH ADDRESS THE FIELD FROM
*   WHERE THE DATA TO BE DEFINED IN A REXX VARIABLE IS
*   LOCATED
*   &ASNAME IS THE NAME TO BE ASSIGNED TO THE DATA FOR USE IN REXX
*   &ERR= IS THE LABEL TO BRANCH TO SHOULD AN ERROR OCCUR WHILE
*   CREATING THE REXX VARIABLE. BY DEFAULT IT IS ABENDØØ1
*   &LEN= IF THE DATA AT &LABEL IS NOT DEFINED SUCH THAT THE LENGTH
*   OF THE DATA IS WHAT YOU WANT, SIMPLY ENTER A NUMBER HERE
*   THAT DEFINES THE LENGTH REQUIRED. CAN ALSO BE USEFUL IF
*   NECESSARY TO DUMP OUT A LARGE AREA.
*   &SUBARRAY= IF A MULTI LEVEL ARRAY IS REQUIRED EG A.1.1 THEN
*   SET THIS VALUE ACCORDINGLY.
*   &DEBIN= IF THE DATA TO BE CREATED IS BINARY, SETTING THIS TO A
*   VALUE WILL CONVERT THE SPECIFIED NUMBER OF BYTES FROM
*   BINARY TO CHARACTER. THE DEFAULT LENGTH FOR THE
*   OUTPUT DATA IS 4 BYTES. IF THIS IS INSUFFICIENT, THEN
*   SPECIFY A SUITABLE &LEN VALUE TO OVERRIDE IT.
*   &LINK= THIS IS A REXX NAME LABLE TO WHICH THE ARRAY COUNT IS
*   LINKED. THE PURPOSE OF THIS IS TO ALLOW A BRANCH OUT
*   OF ARRAY LOOPS WHILE STILL MAINTAINING NUMERIC
*   CONSISTENCY.
************************************************************************
PRINT GEN
LCLA &DEFLEN
&DEFLEN SETA 16
SHOWSET
LITLOC LOCTR
&LABCHECK SETC '@&ASNAME&SUBARRAY'
&LINKNAME SETC '@&LINK'
AIF (D'&LABCHECK).BYPASS
AIF (T'&SUBARRAY EQ 'O').NORMNAME
&LABCHECK DC C'&ASNAME..&SUBARRAY'
AGO .EOFARRAY
.NORMNAME ANOP
&LABCHECK DC C'&ASNAME'
.EOFARRAY ANOP
&LABCHECK._ARRAY DC C'.
&LABCHECK._COUNTER DC PL4'Ø' * COUNTER FIELD FOR THIS ITEM
.BYPASS ANOP
&SYSECT LOCTR
AIF (T'&LINK EQ 'O').DOADD
MVC &LABCHECK._COUNTER,&LINKNAME._COUNTER
AGO .DOUNPK
.DOADD ANOP
AP &LABCHECK._COUNTER.=P'1' * INCREMENT THE COUNTER THIS PASS
.DOUNPK ANOP
UNPK @_UNPACKER,&LABCHECK._COUNTER * UNPACK THE VALUE
OI @_UNPACKER+7,X'FØ' * REMOVE THE SIGN
* NOW NEED TO WORK OUT THE LENGTH OF THE COUNTER BIT TO ADD TO ARRAY
L R15,&LABCHECK._COUNTER * LOAD THE COUNTER VALUE TO WORK
* OUT THE LENGTH
SRL R15,4 * REMOVE THE SIGN
XR R14,R14 * CLEAR R14 FOR A COUNTER
LOOP&SYSNDX DS ØH
SRA R15,4 * MOVE DIGIT BY DIGIT
LTR R15,R15
BZ COUNT&SYSNDX
LA R14.1(R14)
B LOOP&SYSNDX
COUNT&SYSNDX DS ØH
* NOW ADD COUNT FIELD TO NAME
LA R15,@_UNPACKER+7 * POINT TO END OF FIELD
SR R15,R14 * AND COME BACK TO FIRST DIGIT.
MVC &LABCHECK._ARRAY+1(7),Ø(R15)
LA 1,&LABCHECK
ST 1,SHVNAMA
* NOW CALCULATE NEW LENGTH
LA 1,L'&LABCHECK
LA 1,2(R14,R1)
ST 1,SHVNAML
AIF (T'&DEBIN EQ 'O').NORMLAB
*** NOW ALLOW FOR A BINARY CONVERSION
*** FIRST CALCULATE THE ICM VALUE
*
SHOWBASE &LABEL,&ERR=ABEND001,&SUBARRAY=

*****************************************************************************
* MACRO TO CREATE REXX BASE VARIABLES
* SHOULD BE USED IN ASSOCIATION WITH A SHOWARRAY MACRO. NOTE THAT A
* SHOWBASE MACRO IS OPTIONAL IF YOU ALREADY KNOW THE NUMBER OF
* VARIABLES BEING SET. THIS WILL CREATE THE A.Ø ENTRY
*
* MACRO FORMAT:
* SHOWBASE &LABEL,&ERR=,&SUBARRAY=
* WHERE:
* &LABEL IS THE NAME OF THE REXX ARRAY LABEL WHICH HAS BEEN
* CREATED. THIS WILL CREATE THAT LABEL.Ø ENTRY
* &ERR= IS THE LABEL TO BRANCH TO SHOULD AN ERROR OCCUR WHILE
* CREATING THE REXX VARIABLE. BY DEFAULT IT IS ABEND001
* &SUBARRAY= IF SUBARRAYS HAVE BEEN USED THIS WILL INSERT THE
* APPROPRIATE VALUE EG A.1.0
*****************************************************************************
SHOWSET
AIF (T'&SUBARRAY EQ 'O').NORMNAME
&ASNAME  SETC '&LABEL..&SUBARRAY..Ø'
AGO .CHECKER
.NORMNAME ANOP
&ASNAME  SETC '&LABEL..Ø'
.CHECKER ANOP
&LABCHECK SETC '@_&LABEL&SUBARRAY._COUNTER'
AIF (D'&LABCHECK).ITSOK
MNOTE NO ARRAY ELEMENTS DEFINED.
MEXIT
.ITSOK ANOP
LITLOC LOCTR
@_A&SYSNDX DC C'&ASNAME'
&SYSECT LOCTR
LA 1,@_A&SYSNDX
ST 1,SHVNAMA
LA 1,L'@_A&SYSNDX
ST 1,SHVNAML
UNPK @_UNPACKER,&LABCHECK
OI  @_UNPACKER+L'@_UNPACKER-1,C'Ø'
LA 1,@_UNPACKER
ST 1,SHVVALA
LA 1,L'@_UNPACKER
ST 1,SHVVALL
LR 0,10
LA 1,COMS
L 15,IRXEXCOM
BALR 14,15
LTR 15,15
BNZ &ERR
MEND
UCBTAPE AMODE 31
UCBTAPE RMODE ANY
UCBTAPE CSECT
REXREGS
BAKR  R14,R0
LR  R12,R15
LA  R11,2048(,R12)  * ESTABLISH ADDRESSABILITY FOR
   R11,2048(,R11)  * UP TO 8K
USING UCBTAPE,R12,R11
LR  R10,RO
* R10 -> A(ENVIRONMENT BLOCK)
USING ENVBLOCK,R10
L  R9,ENVBLOCK_IRXEXTE  * R9 -> A(EXTERNAL EP TABLE)
USING IRXEXTE,R9
STORAGE OBTAIN,LENGTH=GETLEN,ADDR=(8)
USING COMSDS,R8
* PREPARE THE REXX AREA FOR USE
XC  COMS(COMSLEN),COMS  * SET TO LOW VALUES
LA  R15,COMID
ST  R15,COMS
LA  R15,COMDUMMY
ST  R15,COMS+4
ST    R15,COMS+8
LA    R15,COMSHVB
ST    R15,COMS+12
LA    R15,COMRET
ST    R15,COMS+16
OI    COMS+16,X'80'     * INDICATE END OF PARMS
MVC   COMID,=C'IRXEXCOM'

* COMMENCE THE LOOP OF UCBs LOOKING FOR A RELEVANT DISKS.
* HAVING FIRST SET ADDRESSABILITY TO THE UCB AREA TO R3
LA R3,UCBWORK
USING UCBOB,R3
LA R4,UCBEXTN
USING UCBCMEXT,R4
LA R5,PATHINFO
USING PATH,R5
XC HUNDRED,HUNDRED

UCBLOOP DS ØH
  UCSSCAN COPY,WORKAREA=HUNDRED,RANGE=ALL,UCBAREA=UCBWORK,
    DYNAMIC=YES,DEVCLASS=TAPE,DEVNCHAR=MYDEV,DEVN=Ø,
    PLISTVER=MAX,CMXTAREA=UCBEXTN
* IF R15 CONTAINS Ø4, THEN LAST UCB RETRIEVED SO SET END FLAG
* FOR DEFENSIVE CODE, ASSUME ALL NON-ZERO RETURN CODES ARE THE
* EQUIVALENT OF END OF UCBs
LTR R15,R15          * END OF UCBs?
BNZ RETURNV          * YES SO GO SET THE BASE ARRAY VALUES

*** IF THE DEVICE IS OFFLINE, THEN WE ARE NOT INTERESTED
TM UCBSTAT,X'8Ø'     * IS THIS VOLUME ONLINE
BNO UCBLOOP          * NO, SO GO BACK AROUND

*** THE FOLLOWING SECTION RELATES TO CREATING THE VARIABLES ASSOCIATED
*** WITH THE UCB
MVI ATTRIBS,X'ØA' * NEEDS TO BE DONE PRIOR TO EDTINFO CALL
EDTINFO RTNUNIT,DEVTYPE=UCBTYP,OUTUNIT=UNIT,EXTENDED=YES
SHOWARAY UNIT,UNIT_TYPE
SHOWARAY UCBVOLI,VOLUME
  CLC UCBVOLI,=XL6'ØØ'      * IS THE VOLUME NULL
  BE NULL_JOB              * YES SO SKIP THIS BIT
LOCASCB ASID=UCBASID          * TRY TO GET JOBNAME
LTR 15,15
BNZ NULL_JOB
LM 2,1          * MOVE ADDRESS TO USABLE REGISTER
IAZXJSAB READ,ASCB=(2),JOBNAME=THISJOB
LTR 15,15     * NO DETAILS FOUND?
BNZ NULL_JOB
SHOWARAY THISJOB,JOB_NAME
    B SET_DEVICE

NULL_JOB DS ØH
SHOWARAY BLANK,JOB_NAME

SET_DEVICE DS ØH
SHOWARAY MYDEV,ADDRESS

*** NOW GET THE PATH DETAILS FOR DISPLAY
*** NOTE THAT A NON-ZERO RETURN CODE FORM THIS MACRO INDICATES A
*** NON-ATTACHED DEVICE. PRIOR TO CALLING THIS MACRO IT IS NECESSARY
*** TO ZEROIZE THE WORKAREA, OTHERWISE IT MAY END UP CONTAINING OLD
*** INFORMATION FROM A PREVIOUS CALL.

XC PATHINFO,PATHINFO * ZEROIZE ALL THE AREA
UCBINFO PATHINFO,PATHAREA=PATHINFO,DEVN=UCBCHAN,RETCODE=RC
*
SHOWARRAY RC,RCODE,DEBIN=4
L 15,RC
LTR 15,15
BNZ UCBLOOP
SHOWARRAY PATH#CHPIDS,CHPS,LEN=2,DEBIN=4,LINK=RCODE
*   * SHOW NUMBER OF ATTACHED PATHS
***
*** SECTION PATHLOOP CREATES THE VARIABLES FOR EACH OF THE ATTACHED
*** PATHS. NOTE THAT UP TO 8 CAN BE ATTACHED.
***
PATHLOOP DS ØH
LA R7,PATPATHIDARRAY
 USING PATH_ARRAY_MAP,R7
SHOWARRAY IDPATH,PATH_ID,SUBARRAY=1,DEBIN=2,LINK=RCODE
BAL R2,PATHSET
SHOWARRAY DESCRIPTION,PATH_TYPE,SUBARRAY=1,LINK=RCODE
LA R7,28(,R7) * SHIFT PATH MAP ALONG TO NEXT 28 BYTE SET
SHOWARRAY IDPATH,PATH_ID,SUBARRAY=2,DEBIN=2,LINK=RCODE
BAL R2,PATHSET
SHOWARRAY DESCRIPTION,PATH_TYPE,SUBARRAY=2,LINK=RCODE
LA R7,28(,R7) * SHIFT PATH MAP ALONG TO NEXT 28 BYTE SET
SHOWARRAY IDPATH,PATH_ID,SUBARRAY=3,DEBIN=2,LINK=RCODE
BAL R2,PATHSET
SHOWARRAY DESCRIPTION,PATH_TYPE,SUBARRAY=3,LINK=RCODE
LA R7,28(,R7) * SHIFT PATH MAP ALONG TO NEXT 28 BYTE SET
SHOWARRAY IDPATH,PATH_ID,SUBARRAY=4,DEBIN=2,LINK=RCODE
BAL R2,PATHSET
SHOWARRAY DESCRIPTION,PATH_TYPE,SUBARRAY=4,LINK=RCODE
LA R7,28(,R7) * SHIFT PATH MAP ALONG TO NEXT 28 BYTE SET
SHOWARRAY IDPATH,PATH_ID,SUBARRAY=5,DEBIN=2,LINK=RCODE
BAL R2,PATHSET
SHOWARRAY DESCRIPTION,PATH_TYPE,SUBARRAY=5,LINK=RCODE
LA R7,28(,R7) * SHIFT PATH MAP ALONG TO NEXT 28 BYTE SET
SHOWARRAY IDPATH,PATH_ID,SUBARRAY=6,DEBIN=2,LINK=RCODE
BAL R2,PATHSET
SHOWARRAY DESCRIPTION,PATH_TYPE,SUBARRAY=6,LINK=RCODE
LA R7,28(,R7) * SHIFT PATH MAP ALONG TO NEXT 28 BYTE SET
SHOWARRAY IDPATH,PATH_ID,SUBARRAY=7,DEBIN=2,LINK=RCODE
BAL R2,PATHSET
SHOWARRAY DESCRIPTION,PATH_TYPE,SUBARRAY=7,LINK=RCODE
LA R7,28(,R7) * SHIFT PATH MAP ALONG TO NEXT 28 BYTE SET
SHOWARRAY IDPATH,PATH_ID,SUBARRAY=8,DEBIN=2,LINK=RCODE
BAL R2,PATHSET
SHOWARRAY DESCRIPTION,PATH_TYPE,SUBARRAY=8,LINK=RCODE
B UCBLOOP   * NOW GO AROUND AGAIN
RETURNV DS ØH
SHOWBASE VOLUME
ENDREXX DS ØH
    STORAGE RELEASE,LENGTH=GETLEN,ADDR=(8)
    PR
PATHSET DS ØH
    IOSCHPD CHP_TYPE=TYPEPATH,DESC=DESCRIPTION
    BR R2                    * AND RETURN FOR DISPLAY
    LTORG
BLANK DC C ' '

*  *
PATH_ARRAY_MAP DSECT
IDPATH        DS CL2
    DS C
MASKPATH      DS C
TYPEPATH      DS C
TDESCT DSECT
    IEFUCBOB DEVCLASS=DA
******************************************************************************
*  
******************************************************************************
***      IRXEXCOM PARAMETER AREA                                    ***
******************************************************************************
*  
COMSDS     DSECT
COMS       DS 5AL4
COMID      DS CL8          * IRXEXCOM ID - C'IRXEXCOM'
COMDUMMY   DS AL4          * NOT USED
COMSHVB    DS (SHVBLEN)X   * IRXEXCOM SHVBLOCK (LENGTH FROM DSECT)
COMRET     DS AL4          * IRXECOM RC
COMSLEN    EQU *-COMS
HUNDRED    DS CL1ØØ
UCBWORK    DS CL25Ø
UCBEXTN    DS CL5Ø
MYDEV      DS CL4
PATHINFO   DS CL256
UNIT       DS CL8
ATTRIBS    DS CL1Ø
RC         DS F
THISJOB    DS CL8
STATUS     DS CL7
DESCRIPTION DS CL32
GETLEN     EQU *-COMS
    IOSDPATH
    IAZJSAB
    IHAASCB
    IHAASSB
    DS ØD
    IRXEFPPL
    IRXARGTB
    IRXEVLB
    IRXENVB
ADDITIONAL FUNCTIONALITY

While developing this system I was curious about the possibility of using the Web to draw together the information from the servers on each machine so that the data could be viewed on one screen. The following code uses frames to generate the above screen. In order to exploit this, you will need to modify the HTTP code to point to your IP:PORT address for each of the machines/LPARs where the respective servers reside. Note that if you simply specify the IP:PORT without the tapeinfo command, you will be presented with multiple data entry screens, each of which can be used independently to allow all the servers to be contacted for differing information.

```html
<HTML><HEAD>
<TITLE>This Document Demonstrates Frames</TITLE></HEAD>
<FRAMESET COLS="30%,*">
<FRAME SRC="http://yourip:port/?Command=tapeinfo" SCROLLING="yes" NAME="frame1" FRAMEBORDER="yes">
<FRAMESET COLS="30%,*">
<FRAME SRC="http://yourip:port/?Command=tapeinfo" SCROLLING="yes" NAME="frame2" FRAMEBORDER="yes">
<FRAMESET ROWS="30%,*">
<FRAME SRC="http://yourip:port/?Command=tapeinfo" SCROLLING="yes" NAME="frame3" FRAMEBORDER="yes">
<FRAMESET ROWS="30%,*">
<FRAME SRC="http://yourip:port/?Command=tapeinfo" SCROLLING="yes" [ccc] NAME="frame4">
FRAMEBORDER="yes">
<FRAME SRC="http://yourip:port/?Command=tapeinfo" SCROLLING="yes" [ccc] NAME="frame5">
FRAMEBORDER="yes">
</FRAMESET>
</FRAMESET>
</FRAMESET>
</FRAMESET>
</FRAMESET>
</HTML>
```

Systems Programmer (UK) © Xephon 2001
Sites running TCP/IP with MVS will be interested in Landmark Systems’ TMON for TCP/IP. The product is used to monitor and manage the availability of network devices and overall network performance. It provides a centralized monitor for both OS/390 and non-OS/390 stacks. TMON for TCP/IP complements Landmarks’ TMON for IMS and TMON for Unix Systems Services product family.

The product promises to provide a detailed insight into resources impacting TCP/IP network performance, extending beyond the OS/390 environment to monitor and manage non-OS/390 TCP/IP stacks and Cisco routers, while also providing visibility into OS/390 subsystems, applications, and resources that impact the OS/390 TCP/IP stack.

Specifically, TMON for TCP/IP pinpoints both the problems and root causes throughout the entire network using conditional monitoring and alarming across SNMP and non-SNMP entities. It identifies the performance problems, bottlenecks, and availability issues that can impact business operations.

For further information contact:
Landmark Systems Corporation, 12700 Sunrise Valley Drive, Reston, Virginia 20191-5804, USA.
Tel: (703) 464 1300
Fax: (703) 464 4918
Landmark Systems (UK), The Boardroom, Morton Gardens, Lower Hillmorton Road, Rugby, Warwickshire, CV21 3SX, UK.
Tel: (08700) 744 755
Fax: (08700) 755 766
http://www.landmark.com

Software AG has announced support for SuSE Linux AG’s Linux Enterprise Server for System/390 by its Tamino XML database, which means XML can be used for existing mainframe applications and, therefore, extended to the Web more effectively. The System/390 can host up to 30,000 virtual Linux servers concurrently. Apparently, Software AG ported Tamino in just a few weeks, supported by SuSE and IBM. The Tamino XML Database is the core of Software AG’s Tamino XML Platform, and is an XML-based architecture for developing, processing, and storing documents in XML format.

For further information contact:
Software AG, Uhlandstr. 12, 64297 Darmstadt, Germany.
Phone: +49 6151 92 0
Fax: +49 6151 92 1191
http://www.softwareag.com

IBM has extended its WebSphere Application Server V3.5 Standard Edition to OS/390 with a run time version. It includes support for host Web components such as servlets and Java Server Pages developed with the WebSphere Family V3.5 programming model and it supports servlets and Java Server Pages, developed and tested using WebSphere V3.5 run times and V3.5 tooling on distributed platforms, that can be redeployed unchanged into the new software’s run time.

Contact your local IBM representative for further information.
http://www.ibm.com/software