140

VM update

April 1998

In this issue

3 VM/ESA readiness for the year 2000

10 Transferring files between ICCF and CMS – part 2

20 Managing and documenting REXX procedures

26 Mike Cowlishaw's REXX Page

35 REXX tracking system re-visited

52 VM news

VM Update

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

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

Australian office
Xephon/RSM
GPO Box 6258
Halifax Street
Adelaide, SA 5000
Australia
Telephone: 088 223 1391

Contributions

Articles published in *VM Update* are paid for at the rate of £170 ($250) per 1000 words for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

VM Update on-line

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

Editor
Robert Burgess

Subscriptions and back-issues

A year's subscription to *VM Update*, comprising twelve monthly issues, costs £175.00 in the UK; $265.00 in the USA and Canada; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1990 issue, are available separately to subscribers for £14.50 ($22.50) each including postage.

Editorial panel

Articles published in *VM Update* are reviewed by our panel of experts. Members of the panel include John Illingworth (UK), Reinhard Meyer (Germany), Philippe Taymans (Belgium), Romney White (USA), Martin Wicks (UK), and Jim Vincent (USA).

VM/ESA readiness for the year 2000

A major challenge faces the information technology industry – to have systems and programs ready to operate correctly with dates in and beyond the year 2000. Thankfully, most companies and organizations have already recognized that they must be prepared for the year 2000 date rollover, and are running formally-managed projects to ensure a smooth transition. At a software level, not only must your VM/ESA operating system be year-2000-ready, but also some older IBM program products, other vendor software, subsystems, and applications with date sensitivities may require replacement or modification. Extensive testing is also essential. With fewer than 100 weekends remaining before 1 January 2000, some organizations may not have many full integrated system test slots left.

VM/ESA YEAR 2000 SUPPORT

Specific support for the year 2000 is provided in VM/ESA Version 2 Release 2 and subsequent releases. There is no year 2000 support provided in earlier releases of VM/ESA or in VM/ESA 370 Feature. Neither is there support for the year 2000 in non-ESA versions of VM, such as VM/XA, VM/SP, or VM/SP HPO.

The aim of VM/ESA's year 2000 support is twofold – to ensure that the operating system itself continues to function correctly in and after the year 2000, and to provide the APIs to allow applications to run correctly as well.

The support includes:

- APIs with four-digit year date formats.
- Commands to accept and display four-digit date formats.
- Logic to resolve two-digit years correctly into four-digit years.

VM/ESA exploits a variety of techniques to implement year 2000 support, including century indicators, full four-digit date fields, and interpretation of the century for a two-digit year using a sliding window.

IPLING IN THE YEAR 2000

VM/ESA Version 1 Release 2.1 and later releases will IPL correctly with a year of 2000 or later. For VM/ESA Version 1 Release 2.1 and Release 2.2, APAR VM57927 must be applied. On systems prior to VM/ESA Version 2 Release 2, an immediate re-IPL should be performed unless APAR VM60324 is applied. This is to cope with potential scheduler and dispatching irregularities when the TOD clock is changed by a large amount.

YEAR 2000 GUEST TESTING

VM/ESA provides an ideal environment in which to test the year 2000 readiness of other System/390 guest systems such as VSE/ESA and OS/390, without disruption to production workloads. The real system clock is never altered by a guest, which uses an offset from the real system clock stored in the VMDBK control block for that virtual machine. Adding OPTION TODENABLE to the CP directory entry of the guest allows the guest to set its own system date and time. You can have several guests, each running with a different system date and time in the future, enabling different parts of a year 2000 testing schedule to be carried out concurrently.

TESTING FOR YEAR 2000 SUPPORT

You can readily determine if year 2000 support is present in CP, CMS, and GCS. Diagnose X'00' returns a bitmap where X'7FFC' or X'7FFE' indicates the presence of year 2000 support in CP. In CMS, the presence of year 2000 support can be established either through the Callable Services Library (CSL) routine DMSERP (Extract/Replace) or the REXX CMSFLAG function. Note that these methods return ‘true’ where year 2000 support is present in *both* CMS and CP. For GCS, the FLS macro may be used to test for the presence of year 2000 support. As with CMS, GCS will only give a positive return if support for year 2000 is present in CP as well.

DATE FORMATS FOR CP AND CMS COMMANDS

For the average CMS user, perhaps the most significant enhancement for year 2000 support is the provision of default output date formats on a system-wide and user basis for relevant CP and CMS commands. In many instances, these capabilities will be completely transparent to an end user. Figure 1 summarizes the date format options.

The system-wide default date format can be set through a new configuration file statement, SYSTEM_DATEFORMAT. New CP commands, SET DATEFORMAT and QUERY DATEFORMAT, allow date formats to be set and queried for both the system and individual users. Alternatively, the CP directory statement DATEFORMAT can be used to specify the default date format for a virtual machine. Unless one or more of these new commands is used to specify a different date format, all CP and CMS commands entered by a user will produce date output in the same format as earlier releases of VM.

VISIBLE CHANGES

Several CP commands now accept date format operands to display dates with four-digit years. For example:

```
cp query time isodate
TIME IS 10:42:37 EST TUESDAY 1998-04-06
```

NAME	*FORMAT*
SHOrtdate	mm/dd/yy, mm/dd, yy/mm/dd (as in earlier releases)
FULldate	mm/dd/yyyy, yyyy/mm/dd
ISOdate	yyyy-mm-dd
SYSdefault	Use system-wide default setting for user default
VMDate	Use the user's CMS DEFAULTS setting for FILELIST/RDRLIST

Figure 1: Date format options

A number of CMS commands have been enhanced to accept output date format operands and/or to input dates with four-digit years. These are FILELIST, IDENTIFY, LISTFILE, NOTE, and RDRLIST. CMS Pipeline stage commands will also accept date format options and provide date output in four-digit format where requested. The preview announcement of VM/ESA Version 2 Release 3 mentions a new filter called DATECONVERT, which will perform date conversion and validation. In GCS, the IPL message displays four-digit years, as does the QUERY MODDATE command. Figure 2 shows an example of invoking FILELIST with the FULldate option.

```
LAKINGC  FILELIST AØ  V 169  Trunc=169 Size=19 Line=1 Col=1 Alt=Ø
Cmd Filename  Filetype  Fm   Format  Lrecl   Records  Blocks    Date      Time
    TEST      EXEC      A1   V          44        5        1    1/Ø7/1998   1Ø:51:Ø9
    OFSMAIL   OFSDATA   AØ   F         8ØØ        3        1    1/Ø7/1998   1Ø:12:59
    OFSMAIL   OFSLOGfl  AØ   F          8Ø       57        2    1/Ø7/1998   1Ø:12:13
    £OFSRDR£  £OVRix    AØ   F          79        1        1    1/Ø7/1998   1Ø:12:13
    LASTING   GLOBALV   A1   V          75       51        1    1/Ø7/1998    7:Ø1:18
    PROFILE   EXEC      A1   V          71        6        1    9/18/1997   11:36:11
    OVMAIL    OVMAILix  A1   F         122        4        1    9/1Ø/1997    7:18:44
    1= Help      2= Refresh  3= Quit   4= Sort(type)  5= Sort(date)  6=Sort(size)
    7= Backward  8= Forward  9= FL /n 1Ø=            11= XEDIT/LIST 12= Cursor
```

Figure 2: Invoking FILELIST with the FULldate option

APPLICATION PROGRAMMING INTERFACES

For system and application programmers, detailed knowledge of APIs which involve date formats is essential. New and changed interfaces support four-digit years or include century information. These APIs include CP Diagnose codes, CSL routines, macros, CMS Pipeline stages, REXX functions, and EXEC2 statements. Among other areas where changes have been made for year 2000 support are the CMS Graphical User Interface, File Status Table (FST), spool file descriptor SFBLOK, SPTAPE command, Dump Viewing Facility, and unsupported utility DMSPLU, which allows the date/time stamp on a CMS file to be changed.

REXX PROGRAMS

One area that will demand considerable attention in any VM/ESA installation is REXX programs. Although the DATE() function can return the current date in a variety of formats, many of these do not include any century information and consequently present problems. For example, ‘C’ (Century) type dates can alter expected program behaviour since 1 January 2000 is less than 31 December 1999 with this date format. Another potential cause of difficulty is that of changes to the date format from the default of SHOrtdate. A REXX program may fail to take account of the different output it receives from a CP or CMS command. Although REXX does not provide intrinsic support in its DATE() function for the new date formats FULldate and ISOdate, it is easy to translate between these and the REXX ‘S’ (Standard) date format. For example:

```
/**/
today = date('S')
say today translate('Ø123845867',today'-','Ø12345678')
exit
```

In CMS 13, one can also easily convert between the different REXX date formats. For example, this REXX program converts a supplied REXX USA date format to a REXX standard date format:

```
/**/
arg usadate
sdate = date('S',usadate,'U')
say 'USA Date' usadate 'Standard Date' sdate
exit
```

Note the resulting century when this REXX program is invoked with different date arguments:

```
test Ø1/Ø8/98
USA Date Ø1/Ø8/98 Standard Date 1998Ø1Ø8
Ready; T=Ø.Ø1/Ø.Ø1 1Ø:54:44
test Ø1/Ø8/Ø1
USA Date Ø1/Ø8/Ø1 Standard Date 2ØØ1Ø1Ø8
Ready; T=Ø.Ø1/Ø.Ø1 1Ø:55:26
```

This illustrates how the year 2000 support code in VM/ESA uses a sliding window (-50,+49) to interpret the two-digit year supplied to the REXX DATE() function.

A common difficulty in many VM/ESA installations is identifying which applications execute date-related APIs. There are a number of approaches that help, such as adding a section to the ESAMIGR SAMPLIST file (part of the REXX/EXEC Migration Tool for VM/ESA) to specify strings commonly used in date processing, using a file contents scanning program, or exploiting a sophisticated execution time monitoring tool which detects the invocation of date-related APIs. Generally speaking, a combination of methods will yield the most complete results.

HIGH-LEVEL LANGUAGES

Most installations have large numbers of programs written in popular languages such as COBOL, PL/I, FORTRAN, and C that may involve date usage or manipulation. It is important to ensure that you are running a level of the compiler and run-time library that contains year 2000 support. Sometimes problems can be avoided or overcome by using the IBM Language Environment for MVS and VM run-time libraries. Program modifications will be necessary in other cases. There is extensive documentation about year 2000 support for high-level languages on other platforms as well as VM/ESA, much of which will be of value to a programmer.

TESTING ENVIRONMENT

It is vital that you isolate your year 2000 test environment from your production environment, including all the data upon which your testing is to be carried out. Failure to do so can produce unexpected and potentially serious consequences as the TOD clock is altered and timestamps on files are changed to dates in the future.

INFORMATION SOURCES

There are several excellent sources of information giving guidance on year 2000 matters related to VM/ESA. The *VM and year 2000* page on the World Wide Web at URL http://www.vm.ibm.com/year2000 contains a detailed *VM/ESA Year 2000 Support – Technical Reference* document that will answer most technical questions. It has a hypertext link to the *IBM Licensed Products Migration Matrix for VM/ESA*, which contains notes about VM program products' year 2000 readiness. This page also lists product and/or service providers for VM, to help with year 2000 transition projects. If you get stuck with a problem there is an e-mail address provided where you can seek assistance. The IBM publication *The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation*, GC28-1251, also available from URL http://www.ibm.com/year2000, has information about VM/ESA and related program products, as well as general guidance for all IBM platforms. It should be consulted to find which releases of IBM operating systems and program products are year 2000 ready.

Probably the best cookbook for VM is the recently published ITSO Bulletin (redbook) *VM/ESA Year 2000 Migration – A Case Study*, SG24-2042. It draws upon the practical experience of individuals who have been engaged in real VM/ESA year 2000 projects and provides a wealth of hints and tips to help your own project run smoothly. Virtually every aspect of a year 2000 migration, from planning and resources to sample code and sound advice, is covered. An investment in the time to read it might just save you from an embarrassing moment in two years' time.

Finally, before embarking on your own year 2000 readiness project for VM/ESA, be sure to consult the YR2000VM PSP bucket, available through your local Software Support Centre.

29 FEBRUARY 2000

And just in case you are still debating with your colleagues about whether the year 2000 is a leap year, it will be!

Cliff Laking
IT Technical Specialist
IBM United Kingdom Ltd (UK)

Transferring files between ICCF and CMS – part 2

This month we conclude the article outlining some REXX and ICCF procedures for moving files from ICCF to CMS, and vice versa, *initiated from the ICCF side or from CMS.*

PUNX EXEC

```
/*********************************************************************/
/*  Punch a CMS file to ICCF via a batch job                         */
/*  (/DISC DTSFILE erforderlich)                                     */
/*********************************************************************/
/* Call:   PUNX <vse> fn ft fm                                       */
/*            vse                  : VSE                             */
/*                                 :   default: global variable      */
/*            fn ft fm             : file(s) (wildcard allowed)      */
/*********************************************************************/
trace off
'GLOBALV SELECT $$GLOB$$ GET $vse1    $vse2    $vse3    $vsedef',
                            '$sysid1  $sysid2  $sysid3         '
parse upper arg vse fn ft fm .
if vse = '?' then signal fehler
if vse = substr($vse1,4,3) then vse = $vse1
if vse = substr($vse2,4,3) then vse = $vse2
if vse = substr($vse3,4,3) then vse = $vse3

address xedit 'EXTRACT /FN /FT /FM /ALT'
if rc = Ø & ¬(FTYPE.1 = 'FILELIST' & right(FMODE.1,1) = Ø) then do
   if vse = '' then vse = $vsedef
   if vse ¬= $vse1 & vse ¬= $vse2 & vse ¬= $vse3 then do
      vse = $vsedef
      parse upper arg fn ft fm .
   end
   if fn = '' then fn = FNAME.1
   if ft = '' & fn ¬= 'SEL' then ft = FTYPE.1
   if fm = '' & fn ¬= 'SEL' then fm = FMODE.1
   if alt.1 > Ø then address xedit 'SAVE'
   end
else do
   if vse = '' then signal fehler
   if vse ¬= $vse1 & vse ¬= $vse2 & vse ¬= $vse3 then do
      vse = $vsedef
      parse upper arg fn ft fm .
   end
end
```

```
'CP SET IMSG OFF'
address command 'ESTATE' fn ft fm
if rc ¬= Ø then do
   say fn ft fm 'does not exist'
   signal ende
end
cl = '4'
if vse = $vse1 then sys = $sysid1
if vse = $vse2 then sys = $sysid2
if vse = $vse3 then sys = $sysid3
do until datatype(lib,'W') = 1
   say 'To which ICCF library to you want to transfer the file(s)?'
   pull lib
end
'LISTFILE' fn ft fm '(STACK FIFO'
anz = queued()
do i = 1 to anz
   pull xfn.i xft.i xfm.i
end
queue 'MSGMODE OFF'
queue 'INPUT * $$ JOB JNM=PUNX,CLASS='cl',SYSID='sys
queue 'INPUT // JOB PUNX'
queue 'INPUT // PAUSE PLEAS /DISCONNECT DTSFILE'
queue 'INPUT // ASSGN SYSØ1Ø,DISK,VOL=SYSWK1,SHR'
queue 'INPUT // EXEC DTSUTIL'
do i = 1 to anz
   queue 'INPUT ADD MEMBER' lib xfn.i 'AAAA'
   queue 'GET' xfn.i xft.i xfm.i
   queue 'INPUT END OF MEMBER'
end
queue 'ZONE 1 5'
queue 'TOP'
queue 'N 2'
queue 'CH/* $$ /..$$ /* *'
queue 'TOP'
queue 'N 2'
queue 'CH#/*   #../* #* *'
queue 'TOP'
queue 'N 2'
queue 'CH#/&   #../& #* *'
queue 'BOT'
queue 'INPUT /*'
queue 'INPUT /&'
queue 'INPUT * $$ EOJ'
queue 'FILE'
address command 'ERASE $$PUN$$ $$PUN$$ A'
'XEDIT $$PUN$$ $$PUN$$ A'
'CP SPOOL PUNCH' vse
'PUNCH UASS JOB A (NOH'
'CP SPOOL PUNCH SYSTEM'
```

```
/*******************************************************************/
/* End                                                             */
/*******************************************************************/
ende:
'CP SET IMSG OFF'
exit
/*******************************************************************/
/* Error/Help                                                      */
/*******************************************************************/
fehler:
'VMFCLEAR'
address cms 'type punx    exec * 1 9'
```

MOVING FILES TO CMS FROM ICCF

Now to the ICCF procedures that move members from an ICCF library to the reader of a VM/CMS user. It is possible to transfer one single member with SUBVM, a series of files with SUBVMFIL, or an entire ICCF library with SUBVMALL. This is achieved by submitting a job that punches the member(s) via the ICCF batch utility program DTSUTIL. The name of the target file in CMS will become ‘fn LIBnn’, where ‘fn’ is identical to the ICCF member name and ‘nn’ is the ICCF library number. The CMS file is set by creating a ‘:READ’ card in front of the punched file.

For SUBVMFIL you have to create a file that contains a list of member names (one name per line):

```
     SUBVM member lib
     SUBVMALL * lib     or    SUBVMALL *xxx lib
     SUBVMFIL fname lib
```

where:

- ‘member’ is the ICCF member to be moved.
- ‘lib’ is the ICCF library where the member(s) reside(s).
- ‘xxx’ is the generic member name.
- ‘fname’ is the name of a file that contains a list of ICCF members.

The procedure prompts for the VM/CMS user who should receive the file in his/her reader.

Hardcoded values

The POWER reader class where the job is to run is set to 4. This should be changed to suit your requirements.

The file is punched to class V in CMS.

The job is built in a temporary file named VMRDxxxx in ICCF Library 2, where ‘xxxx’ is the ICCF user-id.

SUBVM

```
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
*  PROCEDURE TO SUBMIT A MEMBER DIRECTLY TO VM RDR (CLASS V)
*  CALL: SUBVM MEMBER LIB
*  FORMAT READ CARD FOR VM:   ':READ  FILENAME FILETYPE'
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
&&OPTIONS ØØ1ØØ11
&&IF &&PARAM1 NE ' ' &&GOTO PASS1
&&TYPE 'SUBVM' IGNORED - NO MEMBER NAME (SUBVM MEMBER LIB)
&&EXIT
&&LABEL PASS1
&&IF &&PARAM2 NE ' ' &&GOTO PASS2
&&TYPE 'SUBVM' IGNORED - NO LIB (SUBVM MEMBER LIB)
&&EXIT
&&LABEL PASS2
&&TYPE PLEASE SPECIFY THE VM/CMS USER WHO SHOULD RECEIVE THE FILE
&&READ &&VARBL1
&&IF &&VARBL1 LE ' ' &&GOTO -PASS2
/CONN OFF
/SW &&PARAM2
&&IF &&RETCOD EQ *INVALID &&GOTO L2
/LIST 1 1 &&PARAM1
&&IF &&RETCOD EQ *FILE &&GOTO L1
&&GOTO WEITER
&&LABEL L1
&&TYPE MEMBER &&PARAM1 NOT FOUND IN LIB &&PARAM2
&&EXIT
&&LABEL L2
&&TYPE LIB &&PARAM2 INVALID (SUBVM MEMBER LIB)
&&EXIT
&&LABEL WEITER
/SW 2
/PUR VMRD&&USERID
/INP NOPROMT
*********:READ
/SAVE VMRD&&USERID
/ED VMRD&&USERID
```

```
N 1
OVERLAYC8 &&PARAM1
OVERLAYC17 LIB
OVERLAYC2Ø &&PARAM2
QUIT
/INP NOPROMPT
&/LOAD DTSSUBMT
&/OPT JSDATA
/PARM &&PARAM1 DIRECT
* $$ JOB JNM=DIRECT
* $$ LST DISP=(D)
* $$ PUN CLASS=(Q)
* $$ EOJ
* $$ JOB JNM=DIRECTBG
* $$ LST DISP=D
* $$ PUN CLASS=Q
* $$ EOJ
* $$ JOB JNM=SEGMENT
* $$ LST DISP=D
* $$ PUN DISP=D,CLASS=Q
* $$ EOJ
* $$ JOB JNM=RETURN
* $$ LST DISP=K,CLASS=Q
* $$ PUN DISP=D,CLASS=Q
* $$ EOJ
* $$ JOB JNM=RETURNBG
* $$ LST DISP=K,CLASS=Q
* $$ PUN DISP=D,CLASS=Q
* $$ EOJ
XXXXXXXXXXXXXXX
* $$ JOB JNM=&&PARAM1,CLASS=4,SYSID=2,PRI=9
* $$ PUN CLASS=V,DEST=(,&&VARBL1)
* $$ LST CLASS=X,DEST=*
// JOB SUBVM
// DLBL DTSFILE,'ICCF.LIBRARY',99/365,DA
// EXTENT SYSØ1Ø,SYSWK1
// ASSGN SYSØ1Ø,DISK,TEMP,VOL=SYSWK1,SHR
// EXEC DTSUTIL
REPRO
&/INCLUDE VMRD&&USERID
PUN MEM(&&PARAM2 &&PARAM1)
&/*
&/&
* $$ EOJ
&&LABEL NONAME
/END
/PEND
/RUN
```

SUBVMALL

```
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
*  PROCEDURE TO SUBMIT MEMBERS DIRECTLY TO VM USER &&VARBL1 RDR(CLASS V)
*  CALL:   SUBVMALL * LIB   OR      SUBVMALL *XXX LIB
*  FORMAT READ CARD FOR VM:   ':READ  FILENAME FILETYPE'
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
&&OPTIONS ØØ1ØØ11
&&MAXLOOP 4Ø96
&&IF &&PARAM1 NE ' ' &&GOTO PASS1
&&TYPE 'SUBVMALL' IGNORED - NO MEMBERS (SUBVMALL *XXX LIB)
&&EXIT
&&LABEL PASS1
&&IF &&PARAM2 NE ' ' &&GOTO PASS2
&&TYPE 'SUBVMALL' IGNORED - NO LIB (SUBVMALL *XXX LIB)
&&EXIT
&&LABEL PASS2
&&TYPE PLEASE SPECIFY VM USER, WHO SHOULD RECEIVE THE FILE(S)
&&READ &&VARBL1
&&IF &&VARBL1 LE ' ' &&GOTO -PASS2
/CONN OFF
/SW &&PARAM2
&&IF &&RETCOD EQ *INVALID &&GOTO L2
&&GOTO WEITER
&&LABEL L2
&&TYPE LIB &&PARAM2 INVALID (SUBVMALL *XXX LIB)
&&EXIT
&&LABEL L5
&&TYPE &&PARAM1 INVALID (SUBVMALL *XXX LIB)
&&EXIT
&&LABEL L7
&&TYPE NO MEMBERS (&&PARAM1) FOUND
&&EXIT
&&LABEL WEITER
*
/ED
STACK OPEN
STACK /* */
STACK CLOSE
QUIT
&&OPTIONS 11X
&&IF &&PARAM1 EQ * &&GOTO W1
/LIB &&PARAM1
&&IF &&RETCOD EQ *INVALID &&GOTO -L5
&&GOTO W9
&&LABEL W1
/LIB FULL
&&LABEL W9
&&OPTIONS ØØX
*
```

```
/SW 2
/PUR VMRD&&USERID
/INP NOPROMT
DUMMY
/SAVE VMRD&&USERID
/ED VMRD&&USERID
STACK CLOSE
N 1
GET $$STACK
TOP
N 1
DEL 3
&&IF &&RETCOD EQ *EOF &&GOTO -L7
&&LABEL SCHLEIF
SHIFT RIGHT 6
OVERLAYC1 PUN MEM(
OVERLAYC9 &&PARAM2/
OVERLAYC21 )/////////////////////////////////////////
DUP 1
U 1
SHIFT LEFT 4
OVERLAYC1 :READ//
OVERLAYC17 /LIB
OVERLAYC21 &&PARAM2
U 1
&&IF &&RETCOD EQ INVALID TOP
INSERT REPRO
N 3
&&IF &&RETCOD EQ INVALID &&GOTO FERTIG
&&GOTO -SCHLEIF
&&LABEL FERTIG
QUIT
* &&EXIT
/INP NOPROMPT
&/LOAD DTSSUBMT
&/OPT JSDATA
/PARM &&PARAM1 DIRECT
* $$ JOB JNM=DIRECT
* $$ LST DISP=(D)
* $$ PUN CLASS=(Q)
* $$ EOJ
* $$ JOB JNM=DIRECTBG
* $$ LST DISP=D
* $$ PUN CLASS=Q
* $$ EOJ
* $$ JOB JNM=SEGMENT
* $$ LST DISP=D
* $$ PUN DISP=D,CLASS=Q
* $$ EOJ
* $$ JOB JNM=RETURN
* $$ LST DISP=K,CLASS=Q
* $$ PUN DISP=D,CLASS=Q
* $$ EOJ
* $$ JOB JNM=RETURNBG
* $$ LST DISP=K,CLASS=Q
* $$ PUN DISP=D,CLASS=Q
* $$ EOJ
XXXXXXXXXXXXXXX
* $$ JOB JNM=SUBVMALL,CLASS=4,SYSID=2,PRI=9
* $$ PUN CLASS=V,DEST=(,&&VARBL1)
* $$ LST CLASS=X,DEST=*
// JOB SUBVMALL
// DLBL DTSFILE,'ICCF.LIBRARY',99/365,DA
// EXTENT SYSØ1Ø,SYSWK1
// ASSGN SYSØ1Ø,DISK,TEMP,VOL=SYSWK1,SHR
// EXEC DTSUTIL
&/INCLUDE VMRD&&USERID
&/*
&/&
* $$ EOJ
&&LABEL NONAME
/END
/PEND
/RUN
```

SUBVMFIL

```
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
*  PROCEDURE TO SUBMIT MEMBERS DIRECTLY TO VM USER &&VARBL1 RDR(CLASS V)
*  CALL:   SUBVMFIL FILE LIB (FILE MUST CONTAIN A LIST OF MEMBERS)
*  FORMAT READ CARD FOR VM:   ':READ  FILENAME FILETYPE'
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
&&OPTIONS ØØ1ØØ11
&&IF &&PARAM1 NE ' ' &&GOTO PASS1
&&TYPE 'SUBVMFIL' IGNORED - NO FILENAME (SUBVMFIL FILE LIB)
&&EXIT
&&LABEL PASS1
&&IF &&PARAM2 NE ' ' &&GOTO PASS2
&&TYPE 'SUBVMFIL' IGNORED - NO LIB (SUBVMFIL FILE LIB)
&&EXIT
&&LABEL PASS2
&&TYPE PLEASE SPECIFY VM USER, WHO SHOULD RECEIVE THE FILES
&&READ &&VARBL1
&&IF &&VARBL1 LE ' ' &&GOTO -PASS2
/CONN OFF
/SW &&PARAM2
/LIST 1 1 &&PARAM1
&&IF &&RETCOD NE *FILE &&GOTO WEITER
&&TYPE FILE &&PARAM1 DOES NOT EXIST
```

```
&&EXIT
&&LABEL WEITER
*
/SW 2
/CONN &&PARAM2
/PUR VMRD&&USERID
/INP NOPROMT
DUMMY
/SAVE VMRD&&USERID
/ED VMRD&&USERID
N 1
INSERT DUMMY
INSERT DUMMY
GET &&PARAM1
TOP
N 1
DEL 3
&&IF &&RETCOD EQ *EOF &&GOTO -L7
&&LABEL SCHLEIF
SHIFT RIGHT 12
OVERLAYC1 PUN MEM(
OVERLAYC9 &&PARAM2/
OVERLAYC21 )/////////////////////////////////////////
DUP 1
U 1
SHIFT LEFT 5
OVERLAYC1 :READ//
OVERLAYC16 /LIB
OVERLAYC2Ø &&PARAM2
U 1
&&IF &&RETCOD EQ INVALID TOP
INSERT REPRO
N 3
&&IF &&RETCOD EQ INVALID &&GOTO FERTIG
&&GOTO -SCHLEIF
&&LABEL FERTIG
QUIT
/INP NOPROMPT
&/LOAD DTSSUBMT
&/OPT JSDATA
/PARM &&PARAM1 DIRECT
* $$ JOB JNM=DIRECT
* $$ LST DISP=(D)
* $$ PUN CLASS=(Q)
* $$ EOJ
* $$ JOB JNM=DIRECTBG
* $$ LST DISP=D
* $$ PUN CLASS=Q
* $$ EOJ
* $$ JOB JNM=SEGMENT
* $$ LST DISP=D
* $$ PUN DISP=D,CLASS=Q
* $$ EOJ
* $$ JOB JNM=RETURN
* $$ LST DISP=K,CLASS=Q
* $$ PUN DISP=D,CLASS=Q
* $$ EOJ
* $$ JOB JNM=RETURNBG
* $$ LST DISP=K,CLASS=Q
* $$ PUN DISP=D,CLASS=Q
* $$ EOJ
XXXXXXXXXXXXXXX
* $$ JOB JNM=SUBVMFIL,CLASS=4,SYSID=2,PRI=9
* $$ PUN CLASS=V,DEST=(,&&VARBL1)
* $$ LST CLASS=X,DEST=*
// JOB SUBVMFIL
// DLBL DTSFILE,'ICCF.LIBRARY',99/365,DA
// EXTENT SYSØ1Ø,SYSWK1
// ASSGN SYSØ1Ø,DISK,TEMP,VOL=SYSWK1,SHR
// EXEC DTSUTIL
&/INCLUDE VMRD&&USERID
&/*
&/&
* $$ EOJ
&&LABEL NONAME
/END
/PEND
/RUN
```

Dr Reinhard Meyer (Germany) © Xephon 1998

VM Update, Issue 137, January 1998

Because of an error by our printers, certain copies of *VM Update* for January 1998 have been misnumbered as Issue 146. The correct issue number should be 137.

If you have a misnumbered copy, please contact Xephon at any of the addresses shown on page 2. We will replace the issue free of charge – our apologies for the mistake.

Managing and documenting REXX procedures

Over the years I have written several hundred REXX procedures, Pipeline stage commands, and auxiliary material such as panel definitions. Fortunately, several years ago, I set up a naming concept which facilitates the management and documentation of these objects. The key points of this concept are as follows:

- All objects reside on a single mini-disk, which is accessed automatically by all our CMS machines.
- All objects have names beginning with the same prefix 'RX', followed by a two-character application code and a 3-digit sequence number.
- Most procedures that are executed directly from the command line by a user have an abbreviation, so users don't need to know the real name of a procedure.
- All objects contain a comment box with a consistent format. In this box, a line labelled with the keyword 'Function:' describes the function of the object.

The procedure RXRD001 (called with the abbreviation RXD) provides the following services:

- It generates a listing containing name and function descriptions of every object.
- It helps update the abbreviations list.
- Given an abbreviation, it opens the corresponding procedure in XEDIT. For example 'RXD TCPTOOL1' will XEDIT the file 'RXTC002 EXEC'.
- It creates new empty objects, giving them an automatically-generated name with the next available sequence number. For example 'RXD NEW RXTC EXEC' will create the file 'RXTC004 EXEC'.

IMPLEMENTATION TIPS

The procedure uses IBM's Display Input/Output Facility for panel processing. It could easily be changed to use simple SAY/PULL processing or ISPF. Note:

- Several installation-dependent variables are set in the INIT section of the procedure.
- The abbreviations contained in the Abbreviations File have to be activated in the PROFILE EXEC of every CMS user machine.

RXRD001 EXEC

```
/*==================================================================*/
/* Name        :  RXRDØØ1 EXEC                                       */
/*==================================================================*/
/* Application :  REXX Utilities EFV                                 */
/*                                                                   */
/* Usage       :  Procedure                                          */
/*                                                                   */
/* Arguments   :  [<synonym>]                                        */
/*                                                                   */
/*                    or                                             */
/*                                                                   */
/*                NEW <prefix> <filetype>                            */
/*                                                                   */
/* Result      :  -                                                  */
/*                                                                   */
/* Function    :  REXXDOC REXX Procedure Maintenance Tool            */
/*==================================================================*/
arg opt

/*==================================================================*/
/* inits                                                             */
/*==================================================================*/
fn='RX*'
ft='*'
fm='B'
s='1'

ofn='REXXDOC'
oft='TEXT'
ofm='A'
fid_syn='efv synonym k'  /* the synonym file */
iosk=''
applname.=''
applname.rxrd='REXX Procedure Maintenance'
applname.rxtc='TCP/IP Tools'

/*==================================================================*/
/* Main                                                             */
/*==================================================================*/

if word(opt,1) = 'NEW' then call create
if opt ¬= '' then call edit_by_synonym

do forever
   call panel 'rxrdØØ4'
   if iosk = 'PFØ3' | iosk = 'PFØ4' then leave
   if iosk = 'PFØ2'
   then
      call syno
   else
      call process
end
return

/*==================================================================*/
/* panel processing                                                 */
/*==================================================================*/
panel:
arg panelid
if panelmsg = 'PANELMSG' then panelmsg=''
if cursor ¬= '' & cursor ¬= 'CURSOR'
then    do; options='(UPDATE 'CURSOR ; alarm='.A'; end
else    do; options=''               ; alarm=''  ; end
'IOS327Ø' panelid options
if rc ¬= Ø then do; say 'IOS327Ø Error:' rc; exit; end
panelmsg='';cursor='';return

/*==================================================================*/
/* create listing                                                   */
/*==================================================================*/
process:
'pipe cms listfile' fn ft fm '| sort | stem files.'

o=Ø
outrecs.=' '
call out copies('=',8Ø)
call out 'REXX Procedure Index         ' date() '   ' time()
call out copies('=',8Ø)
call out
lastname='xxxxxxxx'

do i=1 to files.Ø
   if left(files.i,4) <> left(lastname,4)
   then
```

```
      do
        key=left(files.i,4)
        call out
        call out center(' 'applname.key' ',7Ø,'+')
      end

   lastname=left(files.i,8)
   call generate_format_1 files.i
end

outrecs.Ø=o
'pipe stem outrecs. | chop 8Ø | pad 8Ø | >' ofn oft ofm 'FIXED 8Ø'
panelmsg='Index written to file' ofn oft ofm
return

/*=================================================================*/
/* format 1                                                        */
/*=================================================================*/
generate_format_1:
arg xfid
   'pipe <' xfid '| locate /Function    :/ | var fun_string'
   parse value fun_string with . 'Function    :' fun_text '*/' .
   call out xfid '  ' strip(fun_text)
   xsyn=get_synonym(xfid)
   if xsyn <> '' then call out copies(' ',23) xsyn
return

/*=================================================================*/
/* get  synonym                                                    */
/*=================================================================*/
get_synonym:
arg xname .
'pipe <' fid_syn '| locate 1-8 /'xname'/ | var syn_rec'
syn=word(syn_rec,2)
if syn <> '' then syn='Synonym: 'syn
return syn

/*=================================================================*/
/* out: write an output record                                     */
/*=================================================================*/
out:
parse arg xrec
o=o+1
outrecs.o=xrec
return
/*=================================================================*/
/* edit synonym file                                               */
/*=================================================================*/
syno:
'xedit' fid_syn
```

```
'SYNONYM' fid_syn '(CLEAR'
panelmsg='Synonyms updated'
return
/*==================================================================*/
/* edit an EXEC                                                     */
/*==================================================================*/
edit_by_synonym:
'pipe <' fid_syn '| stem syn.'
do i=1 to syn.Ø
   if word(syn.i,2) = opt
   then
      do
        'xedit' word(syn.i,1) 'EXEC' fm
        exit
      end
end
say 'Synonym' opt 'not found'
exit

/*==================================================================*/
/* create new file                                                  */
/*==================================================================*/
create:
prefix=word(opt,2)
if words(opt) < 3
then
   ftype='EXEC'
else
   ftype=word(opt,3)

'pipe cms listfile' prefix'* *' fm '| sort 1-8 | take last | var last'
if rc = 28 then last=prefix || copies('Ø',8-length(prefix))
lastname=word(last,1)
newname=prefix ||  ,
        right(substr(lastname,length(prefix)+1)+1,7-length(prefix),'Ø')
'xedit' newname ftype fm
exit
```

REXXDOC PANEL

```
;*==================================================================*/
;* Name        :  RXRDØØ4  IOS327Ø                                  */
;*==================================================================*/
;* Application :  REXXDOC                                           */
;*                                                                  */
;* Usage       :  IOS327Ø Panel                                     */
;*                                                                  */
;* Arguments   :  -                                                 */
;*                                                                  */
;* Result      :  -                                                    */
;*                                                                     */
;* Function    :  Panel for REXXDOC                                    */
;*=====================================================================*/
.n
.y
.F F1 F2 F3 F4 F5 F6 F7 F8 F9 F1Ø F11 F12
.JX SET CTL . off
.jx set  normal intensity color=white
.jx set  high   intensity color=green
.JX SET CTL [ HIG=underline col=yel TYPE=(SKIP UNPROTECTED NULLS)
.JX SET CTL % col=red
.JX SET CTL # col=green
.JX SET CTL } col=green hig=rev
.e]
.&alarm
.c
}       REXXDOC   REXX Procedure Maintenance and Documentation        ]

File Selection  [8&fn     [8&ft     [1&fm

Output File     [8&ofn    [8&oft    [1&ofm

 %&panelmsg                                                           ]
.b
  ENTER:Create Index Listing             PF2:Edit Synonyms
PF3:Exit
```

SAMPLE PROCEDURE INDEX

```
REXX Procedure Index
```

```
+++++++++++++++++++++ REXX Procedure Maintenance +++++++++++++++++++++
RXRDØØ1  EXEC     B1    REXXDOC REXX Procedure Maintenance Tool
                        Synonym: RXD
RXRDØØ2  EXEC     B1    Sample EXEC (RXRDØØ2)
RXRDØØ3  EXEC     B1    Sample EXEC (RXRDØØ3)
RXRDØØ4  IOS327Ø  B1    Panel for REXXDOC

++++++++++++++++++++++++++++ TCP/IP Tools ++++++++++++++++++++++++++++
RXTCØØ1  EXEC     B1    Sample Procedure (RXTCØØ1)
RXTCØØ2  EXEC     B1    Sample Procedure (RXTCØØ2)
                        Synonym: TCPTOOL1
RXTCØØ3  EXEC     B1    Sample Procedure (RXTCØØ3)
```

Ch Hofstetter (Switzerland) © Xephon 1998

Mike Cowlishaw’s REXX Page

In the second in the series of VM Web site reviews, we visit Mike Cowlishaw’s REXX page, which can be accessed at http://www2.hursley.ibm.com/rexx/.

It’s long been an article of VM faith that REXX is the best thing since sliced bread. But what kind of bread? This Web page reveals ‘the joy of REXXing’ in all its glory, providing many recipes, ingredient lists, pointers to cookbooks, and directories of REXX chefs and restaurants.

For the uninitiated, although it originated within IBM, REXX was a labour of love – (initially) an unsanctioned skunkworks project of one person, Mike Cowlishaw, who was unsatisfied with 1979-era VM/CMS macro languages such as EXEC 2. In the REXX history, linked from this page, he notes: ‘*This style, while adequate for simple commands, proved cumbersome for the large and complex programs and macros that were soon being written in EXEC 2. It became clear to me that a new language was needed, one based on the more classical syntax and semantics used by languages in the tradition of Algol, Pascal, and PL/I, yet including the command and string programming facilities that EXEC 2 had proven to be so effective and powerful*’. The first specification for the language is dated 29 March 1979.

The specifications, written before implementation, or even design, had begun, were widely circulated within IBM so that the language evolution, based on user feedback, began even before the first REXX program was executed. After rapid adoption throughout IBM and a few tantalizing glimpses given to customers, REXX was released in 1983 with VM/SP Release 3. Since then, it’s been introduced and enthusiastically embraced on nearly all current computing platforms, documented in many books, integrated as a macro language in countless applications, and used for immense mission-critical applications. Touring this Web page recaps REXX history, shows the breadth and diversity of the REXX community and culture, and reveals numerous information and resource sites available on the Web.

The page opens with Mike’s greeting and orientation, *Welcome to the REXX Language page at IBM Hursley*. This page links to documents relating to REXX, Object REXX, and NetREXX programming languages, gleaned from many sources. You will find the latest news, tutorials and other information about the languages, the REXX Language Association, and lots of links to other REXX-related sites.

To the left of this introduction, Mike has, at the time of writing, just added three quick-access icons. This illustrates the dynamic nature of well-managed Web pages – so be prepared to modify Web roadmaps such as this article.

The top icon (showing a stylized ‘*The REXX Language*’, as seen on the cover of Mike’s book) appears in various places and returns to the main REXX page. The middle icon links to the main Object REXX page, and the bottom icon reaches the main NetREXX page. Major link categories from the main page include:

- News.
- Information, tutorials, and documentation.
- More REXX links.
- Background.

Depending on your interests and needs, these are valuable and varied areas for browsing and targetting resources.

The *News* section includes downloadable/orderable software resources, recently introduced reference information, and information on new REXX books. The first group of news links includes three pointers to Object REXX implementations – free for Linux and OS/2 users, and priced for Windows 95/NT systems. Since the Windows 95/NT implementation is described as being upwardly-compatible with ‘classic’ REXX programs for OS/2, and the Linux and OS/2 Object REXX are compatible with Windows versions, IBM is providing a unified REXX image across these systems. In fact, the page linked from *Object REXX for Windows 95 and Windows NT* is the inclusive IBM Object REXX resource, also linked lower down under *in depth*. Showing an attractive graphic for IBM’s Software Solutions Development in Böblingen, Germany (site of IBM REXX development, relocated from Endicott, NY, USA a few years ago), the page opens with '*Object REXX – a Scripting Language, easy to learn, easy to read, and upward compatible with your 'classic' REXX programs*'.

This notes that '*Object REXX extends the system procedures language REXX with object-oriented features. It is upward compatible with previous versions of REXX and provides an easy migration path to the world of objects. So REXX programmers can continue programming in REXX with no change. For newcomers Object REXX is easy to learn. Experienced OO (object-oriented) programmers will get a powerful, state-of-the-art OO programming language*'. The page provides two sample OO applications, *Car Dealer* and *Mobile Agent Demonstration: Bid against two agents*, written by Ueli Wahli and described in IBM's *Object REXX* Redbook publication.

While OO REXX has been discussed for several years and has been available in different forms for some time, its increased availability and (especially) recent introduction as NetREXX is worth several more links under *News*. Reading that '*The reference implementation for NetREXX 1.1 is now available. This fully implements the language specification published in The NetREXX Language and extended by the NetREXX Supplement document. [10 Jan 1998]*' reveals three links to NetREXX resources:

- General information and tools.
- Mike Cowlishaw's new NetREXX book.
- A supplement to the book (although it's copyright 1997!).

A little lower, several NetREXX and OO REXX books, including Mike's recent book, are highlighted, giving both language definition and usage, and platform-specific guidance. REXX devotees and aspirants, especially those with Internet/Java interests, will benefit from exploring NetREXX. The description of Mike's book begins: '*NetREXX is a new human-oriented programming language, designed as an effective and simple alternative to the Java language. With NetREXX, you can create programs and applets for the Java environment faster and more easily than by programming in Java. Using and writing Java classes is especially easy in NetREXX, as the* *different types of numbers and strings that Java expects are handled automatically by the language'.*

Inspired by two very different programming languages, REXX and Java, NetREXX blends the easy-to-learn syntax of REXX with the robustness and portability of the Java environment. The result is a language which is tuned for both scripting and application development, and is therefore truly general-purpose.

The remaining *News* item links to information on the REXX Language Association (REXXLA), highlighting this year's REXX Symposium in North Carolina, 11-13 May. Discounted registration and hotel rates are available until 20 April.

REXXLA was established to further the understanding and use of the REXX programming language. With the release of Object REXX and NetREXX, REXXLA has expanded its mission to include these languages as well. The Association, headquartered at the Research Triangle Park in North Carolina, is international in scope with members throughout the world. Supported by a dedicated group of volunteers, the REXXLA provides many benefits for its members.

The *Information, Tutorials, and Documentation* section gathers just over a dozen world-wide links for learning, exploiting, and discussing REXX fundamentals, current and historical reference material, and REXX community activities.

The first link points to the comp.lang.rexx (CLR) newsgroup. Newsgroups, a collection of bulletin boards also known as a Usenet, a major resource for Internet discussions, reach world-wide with a relatively low-level and accessible protocol; however, Usenet basics are beyond the scope of this article. CLR is the main and authoritative REXX newsgroup, but a few others exist, such as slac.lang.rexx, z-netz.sprachen.rexx, and bit.listserv.tsorexx. At the time of writing, CLR has just over 100 posts spread over about 40 topics, ranging from *How do I access all members of a PDS on MVS?*, through *Base-64 program/algorithm*, to *REXX implementations (list of)*. CLR is a mirror image of the REXXLIST mailing list, one of about seven lists available.

REXX mailing lists are only one topic addressed in one category of the next REXX cookbook to visit, the *REXX FAQ* (Frequently Asked Questions list). This offers 13 categories of information, including *What is REXX?, REXX and the Internet, Free REXX Products, Commercial Products, REXX and ANSI, NetREXX, The REXX Language Association, The REXX Symposium, REXX Bibliography, Common REXX Coding Errors*, and *Frequently Asked Questions*. Look under *REXX and the Internet* for information on mailing lists, File Transfer Protocol (FTP) sites, and more. This FAQ, assembled and written by Dave Martin and Eric Giguere, is a good first place to consult for diverse REXX information. The next two links offer REXX tutorials – one general and introductory by Ian Collier, and the other a framework for IBM-supplied tutorials. Sadly, this page only contains one item – a year-old explanation of TCP/IP Socket Programming with REXX, an interesting but not comprehensive topic.

Links listed under *in depth* provide a wealth of reference information. It's easy to miss announcements and trade press articles about REXX evolution, so the *IBM REXX products information page* allows viewers to:

- Read about IBM's REXX products.
- Search IBM Web servers for the keyword 'REXX' (for the top ten hits).
- Go to IBM REXX Language page for information on REXX.
- Connect directly to IBM home page.
- Send mail to the REXX development organization at rexxhelp@vnet.ibm.com.

The first link is a bit misnamed because it's really a concise list of IBM REXX interpreter and compiler implementations. This page also links to various resources regarding IBM REXX such as:

- IBM Object REXX home page.
- IBM NetREXX home page.
- InterfloX Release 2 – a toolkit which provides a REXX API to Lotus Notes.

- IBM GDDM home page.
- Samples showing use of GDDM in REXX.
- Other IBM products that use REXX.

Unfortunately, this group also contains some dead links, referencing no-longer available pages. But the last link, to IBM products that use REXX, illustrates products both mainstream and obscure that use REXX as script, macro, and command language. It raises the questions, aren't we glad that all these products use REXX? Don't we wish that all IBM and non-IBM products did? A better list of REXX implementations is available back at the main REXX page *in depth* section. This implementations page lists freeware/shareware, IBM implementations, and other commercial implementations. Following the VM tradition, many implementations provide source code. (As a digression, it's entertaining to hear speculation on potential effects of Netscape's releasing its browser source code and yet not discuss several decades of VM and other mainframe operating system and application product source code. Those who have used source code for a while can certainly attest to the mutual benefits of vendors providing it.)

Two links connect to the *X3.274 ANSI Standard for REXX*, which gives information on current and future standards work, errata, etc, along with the current standard in several formats (paper, PostScript). The current standard was designed to be an American National Standard, with the scope: *'The REXX Language' described by Michael F Cowlishaw in his book, altered as necessary to promote portability, reliability, maintainability, and efficient execution of REXX programs on a variety of computing systems.* The committee is now embarking on the standardization of object-oriented REXX. The key part of the new charter is: *The scope of the standard will be that of the existing standard plus consideration of extensions that have evolved in recent years. In particular the ability to work in an 'Object-Oriented' way with both objects created by REXX programs and objects not created by REXX programs will be in the scope.*

A sense of REXX's history and evolution can be gained from two links. Firstly, several of Melinda Varian's reports on REXX Symposia give windows into what was important to the REXX community at several key moments. Secondly, the link to *Historical REXX documents and announcements* starts almost at REXX's (really, 'REX's') beginning, with *REX 1.04 news [21 Aug 1979]* describing a very early internal IBM REX release, and *REX first public paper – SHARE 56 [18 Feb 1981]* recalling the first official glimpse of REX given to the outside world. Of course, such presentations led to immense customer pressure which encouraged IBM to integrate REXX into VM/SP Release 3, which began REXX's evolution through the computer industry.

A sense of REXX's breadth and depth is given from the next section of the main page – *More REXX links* – which includes business, educational, organizational, and personal resources for understanding, using, porting, and enjoying REXX. Picking from among many such links, uni-REXX and REXXTOOLS are examples of products from companies that have embraced REXX early and vigorously, providing or strengthening its use in non-traditional environments.

Described by its vendor, The Workstation Group, uni-REXX is an ANSI standard implementation of the REXX language for Unix and Windows. By using uni-REXX with Unix or Windows, mainframe IS personnel who must now use and manage Unix can avoid the huge learning curve associated with the Unix shell languages (and their equally cryptic Unix-derived alternatives such as PERL), becoming immediately productive in the new environment. When used with Windows, uni-REXX provides the same high levels of productivity obtained when using REXX in the mainframe, Unix, or OS/2 environments, and overcomes the severe limitations of Windows .BAT command files.

Similarly ambitious, REXXTOOLS, from Open Software Technologies, supports both OS/390 and VSE/ESA. REXXTOOLS/MVS provides:

- REXX language access to VSAM, DB2 SQL (Dynamic and Static), QSAM, BPAM (PDS and PDSE), IDCAMS, and many system services.
- Web and TCP/IP support.

- A REXX interpretive compiler is included.
- Many operational samples and utilities. Version 5 of REXXTOOLS/MVS is used in numerous year 2000 projects for finding and correcting suspect data.

REXXTOOLS for VSE/ESA provides access to VSAM, SQL/DS (DB2/VSE), and system services. A REXX interpretive compiler is included with the basic component.

Browsing the *more links* section takes you through Unix, Amiga, Windows, SQL/DB2, Object REXX, NetREXX, OS/2, and many other environments. Near the bottom of the links list appears an *Information Week* article in which Mike Cowlishaw explains how the new (in March 1997) NetREXX language can benefit REXX developers. At the time of writing, this section concludes with six intriguing links marked *new*, which offer:

- Vasilis (Bill) Vlachoudis' PC-DOS, MS-DOS, and Unix BREXX interpreter (includes source code).
- Oberon Software's REXX scripting support for Java (Oberon Software).
- Manfred Schneider's Object Orientation collection.
- Les Moull's MAID – GUI front end-tools for REXX for Windows 95/NT, OS/2, etc.
- John Blumel's OREXX/SQL Object Framework (44KB zip).
- David Alcock's REXX Anywhere! portability page.

Finally, the *REXX page* is completed with background information, offering:

- A brief history of REXX.
- REXX advantages and disadvantages.
- A look ahead to the future of REXX.
- So, what's NetREXX?

REXX's history is entertaining and instructive, showing how today's elegant and pervasive language, technology, and implementations, grew from a single inspiration. It concludes, *'REXX is now available for most significant operating systems, and from several vendors, not just IBM. Freeware or shareware versions, often excellent, are also available for PC-DOS, Unix, and other systems'*.

And with Object REXX and NetREXX just recently available and increasingly popular, REXX will be discovered and adopted by new generations of programmers and users, and applied very visibly across the Internet.

The discussion of REXX's advantages and disadvantages can help evaluate REXX for various projects, or deal with questions from colleagues or managers regarding REXX's fitness for specific applications. Mike solicits requests and suggestions regarding the Web page compilation, so it is sure to grow as REXX implementers, developers, users, authors, and consultants contribute topics, resources, pointers, and software for the entire REXX community to share. Finally, at the bottom of the page, Mike's name links to a brief biography, his e-mail address is available for communication, and a small icon greets visitors with a brief audio welcome from Mike – hardly as satisfying as chatting with him in person.

Gabriel Goldberg
Computers and Publishing (USA)

Call for papers

Why not share your expertise and earn money at the same time? *VM Update* is looking for REXX EXECs, macros, program code, etc, that experienced VMers have written to make their life, or the lives of their users, easier. We will publish it (after vetting by our expert panel) and send you a cheque when the article is published. Articles can be of any length and can be sent or e-mailed to Robert Burgess at any of the addresses shown on page 2. Why not call now for a free copy of our *Notes for contributors*?

REXX tracking system re-visited

A Problem Tracking Facility (PTF) was first introduced in *REXX tracking facility* in *VSE Update,* Issue 23, September 1996, and Issue 24, December 1996. This has now been re-written to be year 2000 compliant.

PTF was designed to track support centre calls, but quickly evolved into a common tracking system to fit several needs, such as tracking night calls, data centre problems, inventory control, and Help Desk calls. PTF is a series of REXX EXECS and ISPF panels running in the VM environment to perform the following functions:

- Quick opening of incidents to begin immediate tracking.
- Listing of incidents based on various search criteria, including text search and wildcards.
- Use of a PF key to modify panel names, field names, printer-id, column headers, etc, on-the-fly for the current session or permanently.
- Use of a PF key to restore original configuration values.
- Automatic updating of control information, such as last update and updating user-id.
- Use of the XEDIT editor with pop-up window to display control information for the member being worked on.
- Automatic rollover from one year to the next, including year 2000.

Once the base modules are loaded to a common shared disk, and the subset modules are loaded to a read/write disk, PTF is invoked by typing PTF, followed by parameters which are identical to the CPLINK command minus the TO, AS, and password parameters. By loading the subset onto multiple disks, each disk becomes a separate database and the configuration defaults and screens can be tailored specifically for that database. The user determines which database to use based on the disk address used in the PTF command. For example, the following command invokes PTF using a database on Jackson’s 400 disk, provided the subset modules were previously loaded to that disk:

```
    PTF jackson 4ØØ 4ØØ mr
```

The user is prompted for the password if the disk is password protected. If it’s already linked read/write to someone else, PTF goes ahead and links it read-only, in which case all functions remain available except permanent updating and deleting. The type of link is displayed on each panel.

Figure 1 shows the modules that comprise the PTF system. Some members have mixed-case filetypes to prevent accidental deletes. These must remain mixed-case unless the PTF2 EXEC references are changed as well.

Configuration members contain two lines for each data item that can be altered during configuration. The first line is the installed value and the second line is the user-altered value. Each successive group of two lines corresponds to the next alterable field on the panel. Source for

```
BASE MODULES                    SUBSET MODULES

PTF       EXEC                   PTF       CfgØØ
PTF2      EXEC                   PTF       CfgØ1
PTFØØ     PANEL                  PTF       CfgØ2
PTFØ1     PANEL                  PTF       CfgØ3
PTFØ2     PANEL                  PTF       CfgØ6
PTFØ3     PANEL                  PTF       Cfg99
PTFØ4     PANEL                  PTFSKEL   DATA
PTFØ6     PANEL
PTFØ7     PANEL
PTF1Ø     PANEL
PTF11     PANEL
PTF12     PANEL
PTF13     PANEL
PTF17     PANEL
PTF99     PANEL
PTF       XEDIT
PTF2      XEDIT
PTFCTRL   XEDIT
PTFD      XEDIT
PTFQUIT   XEDIT
```

Figure 1: PTF system modules

configuration members appears later. Member CFG02 contains trailing blank lines that do not appear in this article but which must be present in the member. CFG02 must have a size of 394 lines and be arranged as documented later. Configuration members should be manually edited to set the installation defaults as desired. These are the values that will be restored if you choose to use the PF2 RESTORE DEFAULTS key after permanent configuration has been done. Again, configuration members must have mixed-case filetypes, which can be set with the PTF3 maintenance EXEC. PTF3 also converts filetypes to uppercase for maintenance if necessary. Configuration members must also have a fixed record format with a record length of 80.

EXPLANATION OF FIGURES

Example screens are shown in Figures 2 to 12. Figure 2 shows the primary panel displayed when PTF is invoked and Figure 3 shows the

```
PTF00            Problem Tracking Facility                 01/08/
MODE=UPDATE      Incident Select Panel                     14:30

   Enter Incident Number, OPEN or specific search criteria...

           <== Select Incident Number or OPEN

         A <== Select OPEN, CLOSED, or ALL Incident Number

           <== Select OPEN/CLOSED/ALL on or after this date (yyyymmdd)

           <== Select author who originally opened the Incident Number

           <== Select Component Code

           <== Select Vendor Name

           <== Select Vendor Refid

           <== Find words in description or text


PF2 =Refresh   PF3,4 =Exit   ENTER =Fullist   PF6 =Password
PF7 =List Component Code     PF8 =List Vendor Name  PF9 =Configure
```

Figure 2: Primary panel displayed when PTF is invoked

```
PTF03           Problem Tracking Facility                01/08/
MODE=UPDATE     Control Panel                            14:33

Enter or update control information as necessary...

Incident Number     TS980005

Component Code  ==>

Description     ==>
                ==>

Vendor Name     ==>      Created     19980108  14:33:32  by  CM240

Vendor Refid    ==>      Last update 19980108  14:33:32  by  CM240

Severity Code   ==>      Status   ==> O


PF3 =Return                 PF4 =Exit
PF7 =List Component Code    PF8 =List Vendor Name    PF10 =Print
```

Figure 3: Control panel display

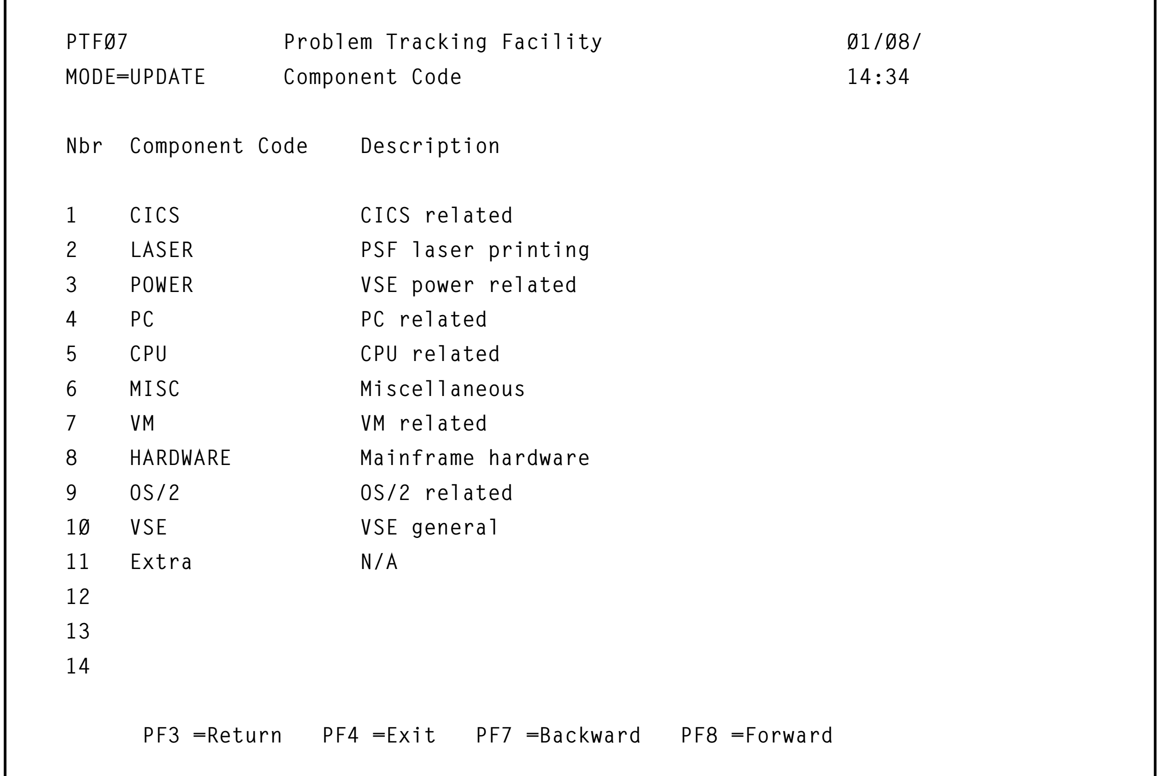
control panel displayed when OPEN is selected on the PTF00 panel.

Figure 4 shows the component code panel, which is displayed when PF7 is pressed from the PTF00 or PTF07 panels and Figure 5 shows the vendor name panel, displayed when PF8 is pressed from these panels.

Figure 6 shows the FULLIST panel, which is displayed when ENTER is pressed from the PTF00 panel.

Figure 7 shows a sample XEDIT session, displayed when ENTER is pressed from the PTF01 panel. Figure 8 shows a sample control-information display which appears when PF9 is pressed from the XEDIT session.

Figure 9 shows the password panel, which is displayed when PF6 is pressed from the PTF00 panel or on the first occasion an update or delete is attempted from the PTF01 panel (this only appears when MODE=UPDATE).

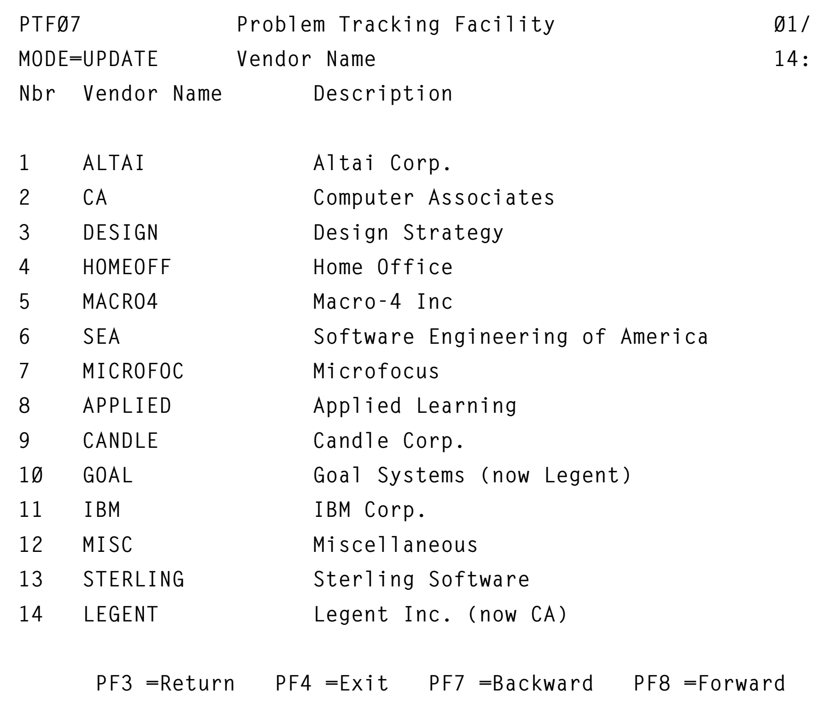


```
PTFØ7             Problem Tracking Facility                  Ø1/Ø8/
MODE=UPDATE       Component Code                             14:34

Nbr  Component Code    Description

1    CICS              CICS related
2    LASER             PSF laser printing
3    POWER             VSE power related
4    PC                PC related
5    CPU               CPU related
6    MISC              Miscellaneous
7    VM                VM related
8    HARDWARE          Mainframe hardware
9    OS/2              OS/2 related
1Ø   VSE               VSE general
11   Extra             N/A
12
13
14

      PF3 =Return   PF4 =Exit   PF7 =Backward   PF8 =Forward
```

Figure 4: Component code panel

Figure 10 shows the configuration panel, displayed when PF9 is pressed from the PTF00 panel, and Figure 11 shows a sample of a panel being displayed in configuration mode. Other panels are basically the same except for the FULLIST panel. Although they may not show in this article, entries that can be changed are displayed in reversed or highlighted fashion. In this panel for example, the following fields/ field names are highlighted:

- Control panel
- Incident number
- Component code
- Description
- Vendor name
- Severity code



```
PTFØ7            Problem Tracking Facility               Ø1/Ø8/
MODE=UPDATE      Vendor Name                             14:41
Nbr  Vendor Name       Description

1    ALTAI             Altai Corp.
2    CA                Computer Associates
3    DESIGN            Design Strategy
4    HOMEOFF           Home Office
5    MACRO4            Macro-4 Inc
6    SEA               Software Engineering of America
7    MICROFOC          Microfocus
8    APPLIED           Applied Learning
9    CANDLE            Candle Corp.
1Ø   GOAL              Goal Systems (now Legent)
11   IBM               IBM Corp.
12   MISC              Miscellaneous
13   STERLING          Sterling Software
14   LEGENT            Legent Inc. (now CA)

      PF3 =Return   PF4 =Exit   PF7 =Backward   PF8 =Forward
```

Figure 5: Vendor name panel

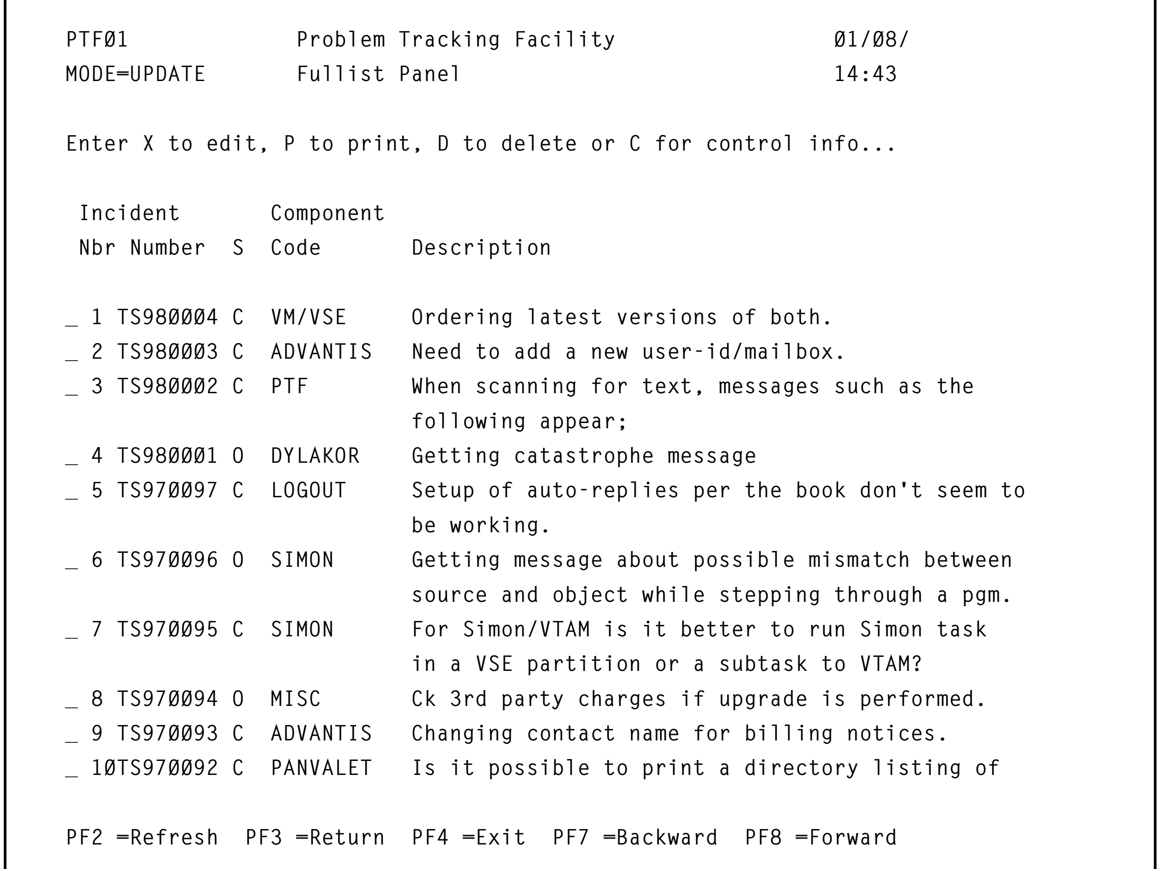
- Status.

When one of these is updated, the change is propagated to all occurrences within the system, ie fields and field names are stored as variables which are changed immediately.

In the FULLIST panel (Figure 12), you can select which two columns are displayed by selecting one of the numbers shown in the list. You can even have the same column twice if you wish. 'Fullist Panel' is the only other field/field name on this particular panel that can be changed.

PTF

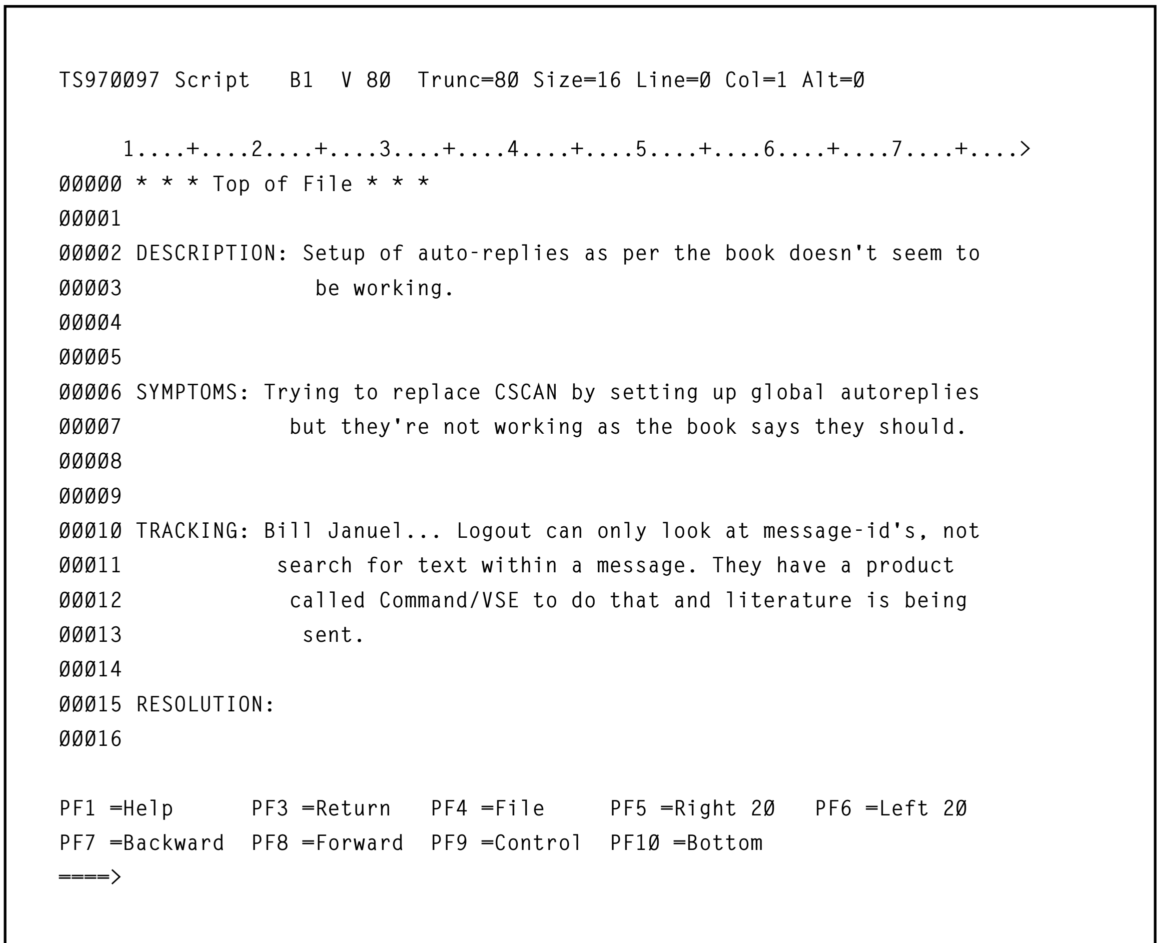
```
/*****************************************************************/
/*****************************************************************/
/***                                                           ***/
/***  PROGRAM NAME  - PTF  Problem Tracking Facility           ***/
/***                                                           ***/
```



```
PTFØ1             Problem Tracking Facility                 Ø1/Ø8/
MODE=UPDATE       Fullist Panel                             14:43

Enter X to edit, P to print, D to delete or C for control info...

 Incident       Component
 Nbr Number  S  Code       Description

_ 1 TS98ØØØ4 C  VM/VSE     Ordering latest versions of both.
_ 2 TS98ØØØ3 C  ADVANTIS   Need to add a new user-id/mailbox.
_ 3 TS98ØØØ2 C  PTF        When scanning for text, messages such as the
                           following appear;
_ 4 TS98ØØØ1 O  DYLAKOR    Getting catastrophe message
_ 5 TS97ØØ97 C  LOGOUT     Setup of auto-replies per the book don't seem to
                           be working.
_ 6 TS97ØØ96 O  SIMON      Getting message about possible mismatch between
                           source and object while stepping through a pgm.
_ 7 TS97ØØ95 C  SIMON      For Simon/VTAM is it better to run Simon task
                           in a VSE partition or a subtask to VTAM?
_ 8 TS97ØØ94 O  MISC       Ck 3rd party charges if upgrade is performed.
_ 9 TS97ØØ93 C  ADVANTIS   Changing contact name for billing notices.
_ 1ØTS97ØØ92 C  PANVALET   Is it possible to print a directory listing of

PF2 =Refresh  PF3 =Return  PF4 =Exit  PF7 =Backward  PF8 =Forward
```

Figure 6: FULLIST panel display

```
/***  Function      - This EXEC is executed when PTF is entered.  ***/
/***                  It sets up initial defaults and invokes      ***/
/***                  the PTF2 EXEC which drives the ISPF panels   ***/
/***                  and does the main processing.                ***/
/***                                                               ***/
/*********************************************************************/
/*********************************************************************/
'VMFCLEAR'
PARSE UPPER ARG VADDR INCADDR INCFM INCRW .

IF VADDR = '' THEN DO
   SAY 'Invalid or missing disk address'
   EXIT ; END

IF INCADDR = '' | INCADDR = '=' THEN INCADDR = VADDR

IF INCFM = '' THEN DO
   GETFMADR                          /* FIND NEXT-AVAIL FILEMODE  */
   PARSE UPPER PULL ARG X
   PARSE VAR X INCFM .
   END

IF INCRW = '' | INCRW = '=' THEN INCRW = 'MR'

'VMLIB' VADDR INCADDR INCFM INCRW       /* VMLIB can be replaced by   */
                                        /* the standard CP LINK       */

MAKEBUF                                 /* SEE IF R/W OR R/O         */
'Q V' INCADDR '(STACK'
PARSE UPPER PULL ARG X
PARSE VAR X . . . ACCSW .
DESBUF

IF ACCSW = 'R/W' THEN DO
   ACCTYP = 'Mode=UPDATE'
   END
ELSE
```



```
TS970097 Script   B1  V 80  Trunc=80 Size=16 Line=0 Col=1 Alt=0

     1....+....2....+....3....+....4....+....5....+....6....+....7....+....>
00000 * * * Top of File * * *
00001
00002 DESCRIPTION: Setup of auto-replies as per the book doesn't seem to
00003              be working.
00004
00005
00006 SYMPTOMS: Trying to replace CSCAN by setting up global autoreplies
00007            but they're not working as the book says they should.
00008
00009
00010 TRACKING: Bill Januel... Logout can only look at message-id's, not
00011           search for text within a message. They have a product
00012            called Command/VSE to do that and literature is being
00013             sent.
00014
00015 RESOLUTION:
00016

PF1 =Help      PF3 =Return   PF4 =File     PF5 =Right 20   PF6 =Left 20
PF7 =Backward  PF8 =Forward  PF9 =Control  PF10 =Bottom
====>
```

Figure 7: Sample XEDIT session

```
Incident Number      TS970097

Component Code   ==> LOGOUT

Description      ==> Setup of auto-replies as per the book doesn't seem to
                 ==> be working.

Vendor Name      ==> MACRO4 Created 19971222  13:46:04 by CM240009

Vendor Refid     ==> LGDU781 Last update 19971222 14:45:38 by CM240009

Severity Code    ==> 2       Status   ==> C

               PF3 =Return
```

Figure 8: Sample control-information display

```
IF ACCSW = 'R/O' THEN DO
   ACCTYP = 'Mode=READ/ONLY'
   PULL
   END
ELSE DO
   SAY 'Invalid or missing disk address'
   EXIT
   END

'FILEDEF ISPPLIB DISK ISPNULL PANEL * (PERM CONCAT)'
'FILEDEF ISPMLIB DISK ISPNULL MESSAGE * (PERM CONCAT)'
'EXEC ISPSTART CMD(PTF2' INCFM ACCSW ACCTYP INCADDR') NEWAPPL(PTF)'
'FILEDEF ISPPLIB CLEAR'
'FILEDEF ISPMLIB CLEAR'
```

PTF2

```
/* */
/* TRACE L */
/*********************************************************************/
/*********************************************************************/
/***                                                               ***/
/***  PROGRAM NAME  - PTF2  Main Module                            ***/
/***                                                               ***/
/***  Function      - This EXEC does the main processing. It      ***/
```

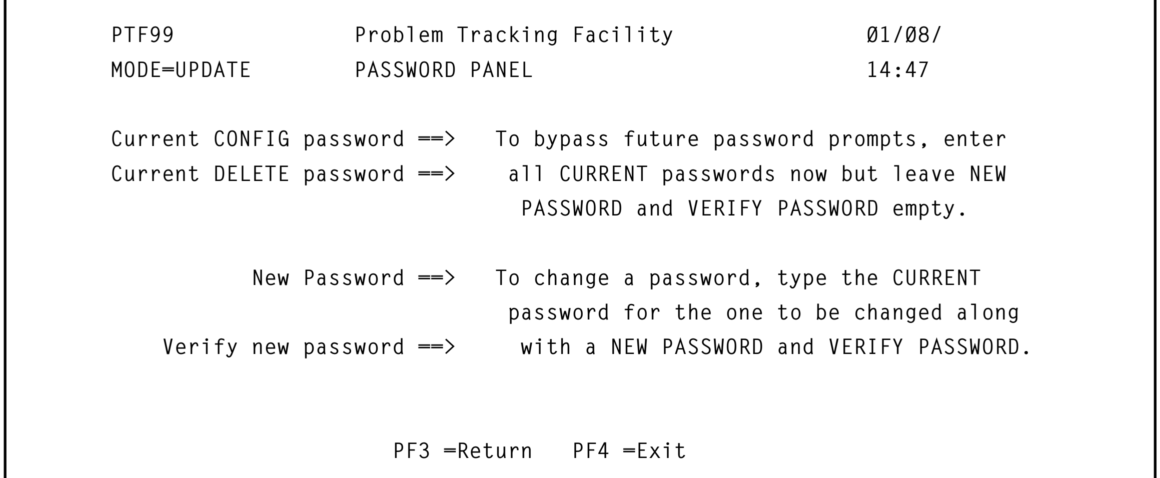
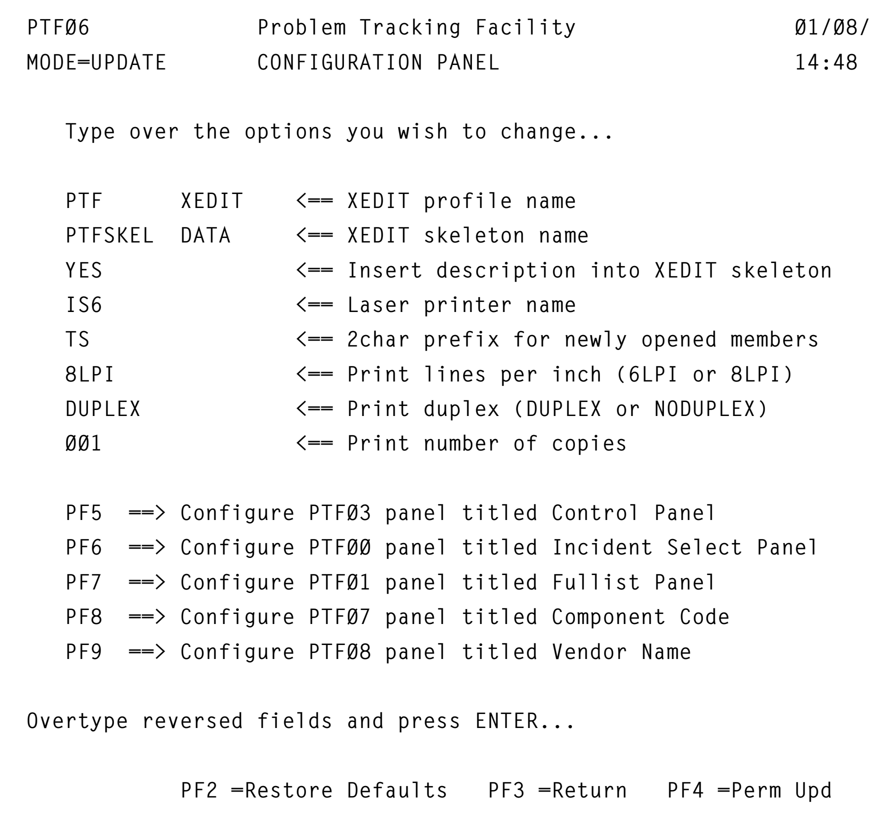

```
PTF99             Problem Tracking Facility              Ø1/Ø8/
MODE=UPDATE       PASSWORD PANEL                         14:47

Current CONFIG password ==>   To bypass future password prompts, enter
Current DELETE password ==>    all CURRENT passwords now but leave NEW
                                PASSWORD and VERIFY PASSWORD empty.

           New Password ==>   To change a password, type the CURRENT
                               password for the one to be changed along
    Verify new password ==>     with a NEW PASSWORD and VERIFY PASSWORD.


                      PF3 =Return   PF4 =Exit
```

Figure 9: Password panel

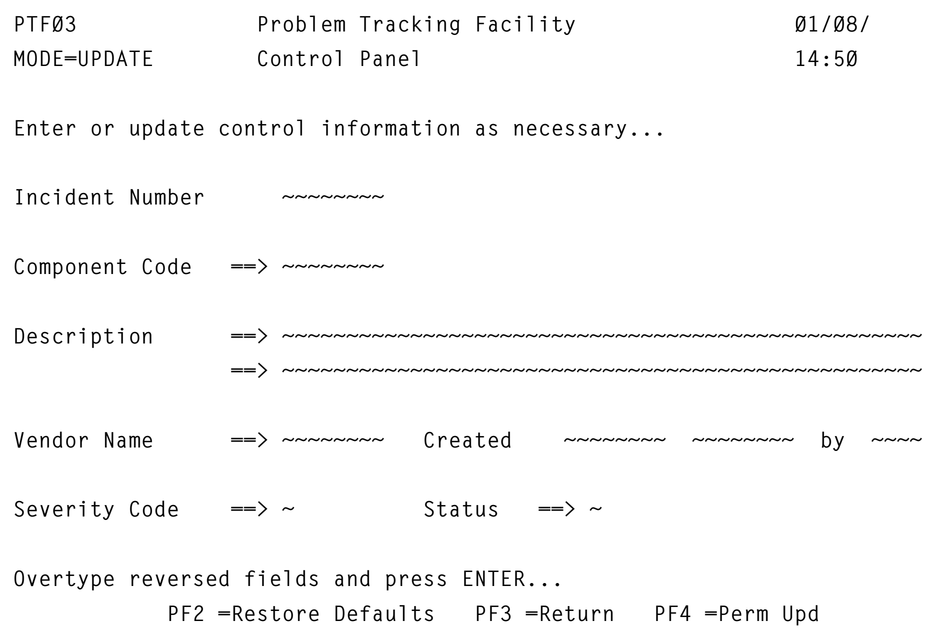
```
/***                  displays ISPF panels, performs calcula-     ***/
/***                  tions, updates incident members, and         ***/
/***                  controls the entire PTF session.            ***/
/***                                                              ***/
/*******************************************************************/
/*******************************************************************/

PARSE UPPER ARG INCFM ACCSW ACCTYP INCADDR .

CFGPFX  = X2C(C38687)                /* CONFIG FILE PREFIX        */
CALL CONFIGØØ                        /* GET DEFAULTS              */
CALL CONFIGØ1
CALL CONFIGØ2
CALL CONFIGØ3
CALL CONFIGØ6
CALL CONFIG99

ZPFCTL   = 'OFF'                     /* DON'T SHOW ISPF PF KEYS   */
SESSTYP  = 'INCIDENT'                /* DEFAULT SESSION TYPE      */

ADDRESS ISPEXEC 'VPUT (ZPFØ1 ZPFØ2 ZPFØ3 ZPFØ4 ZPFØ5 ZPFØ6 ZPFØ7 ZPFØ8
ZPFØ9 ZPF1Ø ZPF11 ZPF12) PROFILE'
ADDRESS ISPEXEC 'VPUT (ZPF13 ZPF14 ZPF15 ZPF16 ZPF17 ZPF18 ZPF19 ZPF2Ø
ZPF21 ZPF22 ZPF23 ZPF24) PROFILE'
ADDRESS ISPEXEC 'VPUT (ZPFCTL) PROFILE'

/*******************************************************************/
/***  INITIALIZE & DISPLAY PRIMARY PANEL                          ***/
```



```
PTF06           Problem Tracking Facility                01/08/
MODE=UPDATE     CONFIGURATION PANEL                      14:48

   Type over the options you wish to change...

   PTF      XEDIT    <== XEDIT profile name
   PTFSKEL  DATA     <== XEDIT skeleton name
   YES               <== Insert description into XEDIT skeleton
   IS6               <== Laser printer name
   TS                <== 2char prefix for newly opened members
   8LPI              <== Print lines per inch (6LPI or 8LPI)
   DUPLEX            <== Print duplex (DUPLEX or NODUPLEX)
   001               <== Print number of copies

   PF5  ==> Configure PTF03 panel titled Control Panel
   PF6  ==> Configure PTF00 panel titled Incident Select Panel
   PF7  ==> Configure PTF01 panel titled Fullist Panel
   PF8  ==> Configure PTF07 panel titled Component Code
   PF9  ==> Configure PTF08 panel titled Vendor Name

Overtype reversed fields and press ENTER...

            PF2 =Restore Defaults   PF3 =Return   PF4 =Perm Upd
```

Figure 10: Configuration panel via PF9 from PTF00 panel

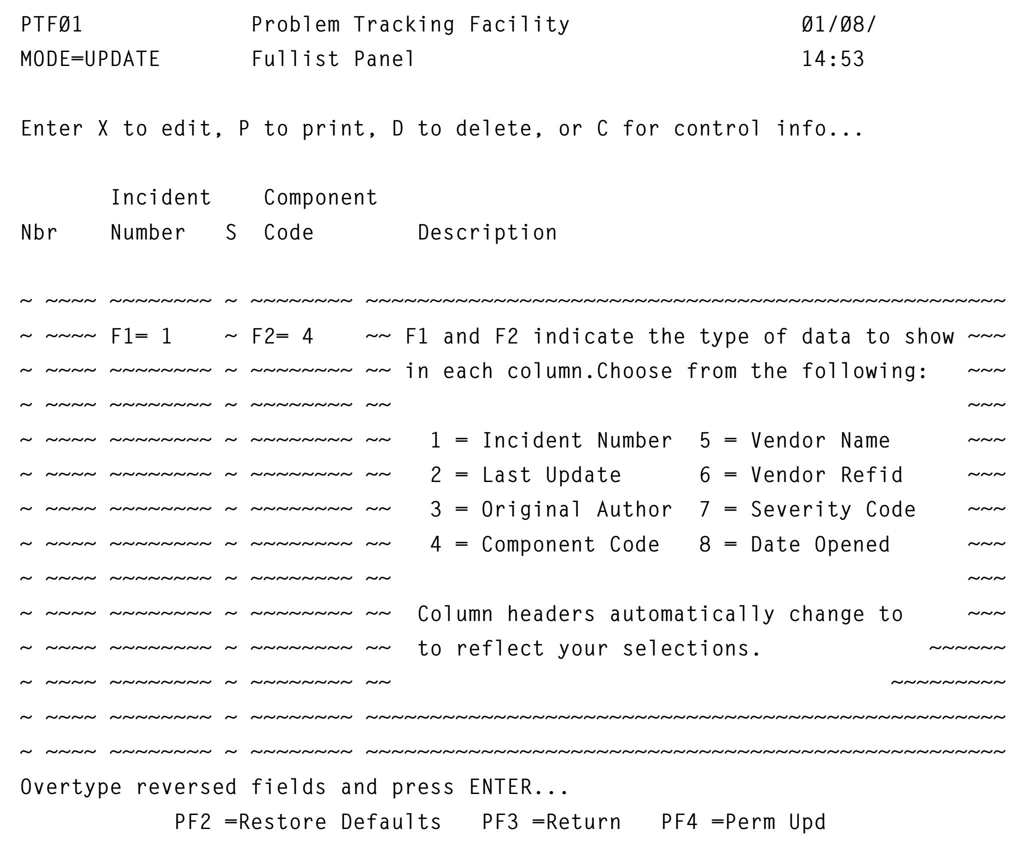
```
/******************************************************************/
INIT:

CALL SETHDR
DIRFN   = 'PTF'
DIRFT   = X2C(C489998583A3) || SUBSTR(DATE(J),1,2)
INCFT   = X2C(E283998997A3)
EDPRTFT = 'CONTROL'

ACCT     = ''
INCID    = ''  ; AUTHOR  = ''   ; COMP   = ''  ; SRCH  = ''
DATE     = ''  ; S       = 'A'  ; VEND   = ''  ; VREF  = ''

WDATE    = SUBSTR(DATE(U),1,6)
WDATE    = WDATE || WORD(DATE(N),3)

CALL DISP00

/******************************************************************/
/***  EDIT USER RESPONSE                                        ***/
```



```
PTFØ3              Problem Tracking Facility                  Ø1/Ø8/
MODE=UPDATE        Control Panel                              14:5Ø

Enter or update control information as necessary...

Incident Number      ~~~~~~~~

Component Code   ==> ~~~~~~~~

Description      ==> ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                 ==> ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Vendor Name      ==> ~~~~~~~~   Created    ~~~~~~~~  ~~~~~~~~  by  ~~~~

Severity Code    ==> ~          Status   ==> ~

Overtype reversed fields and press ENTER...
            PF2 =Restore Defaults   PF3 =Return   PF4 =Perm Upd
```

Figure 11: Panel displayed in configuration mode

```
/*******************************************************************/
CKRESP:

IF LSCREEN = 'PTFØØ' | LSCREEN = 'PTF1Ø' THEN SIGNAL CKRPTFØØ
IF LSCREEN = 'PTFØ1' THEN SIGNAL CKRPTFØ1

IF LSCREEN = 'PTFØ2' THEN DO
   CALL CKRPTFØ2
   SIGNAL CKRESP ; END

IF LSCREEN = 'PTFØ7' | LSCREEN = 'PTFØ8' THEN DO
   CALL CKRPTFØ7
   SIGNAL CKRESP ; END

MESSAGE = 'Internal mismatch on panelid; notify Tech support'
CALL DISPØØ ; SIGNAL CKRESP

/*******************************************************************/
/***  EDIT RESPONSES FROM PTFØØ PANEL                            ***/
/*******************************************************************/
CKRPTFØØ:

IF CPFKEY = 'PFØ2' THEN SIGNAL INIT

IF CPFKEY = 'PFØ3' | CPFKEY = 'PFØ4' THEN SIGNAL EXIT
```



```
PTFØ1             Problem Tracking Facility                  Ø1/Ø8/
MODE=UPDATE       Fullist Panel                              14:53

Enter X to edit, P to print, D to delete, or C for control info...

       Incident    Component
Nbr    Number   S  Code        Description

~ ~~~~ ~~~~~~~~ ~ ~~~~~~~~ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~ ~~~~ F1= 1    ~ F2= 4    ~~ F1 and F2 indicate the type of data to show ~~~
~ ~~~~ ~~~~~~~~ ~ ~~~~~~~~ ~~ in each column.Choose from the following:   ~~~
~ ~~~~ ~~~~~~~~ ~ ~~~~~~~~ ~~                                             ~~~
~ ~~~~ ~~~~~~~~ ~ ~~~~~~~~ ~~   1 = Incident Number  5 = Vendor Name      ~~~
~ ~~~~ ~~~~~~~~ ~ ~~~~~~~~ ~~   2 = Last Update      6 = Vendor Refid     ~~~
~ ~~~~ ~~~~~~~~ ~ ~~~~~~~~ ~~   3 = Original Author  7 = Severity Code    ~~~
~ ~~~~ ~~~~~~~~ ~ ~~~~~~~~ ~~   4 = Component Code   8 = Date Opened      ~~~
~ ~~~~ ~~~~~~~~ ~ ~~~~~~~~ ~~                                             ~~~
~ ~~~~ ~~~~~~~~ ~ ~~~~~~~~ ~~  Column headers automatically change to     ~~~
~ ~~~~ ~~~~~~~~ ~ ~~~~~~~~ ~~  to reflect your selections.            ~~~~~~
~ ~~~~ ~~~~~~~~ ~ ~~~~~~~~ ~~                                      ~~~~~~~~~
~ ~~~~ ~~~~~~~~ ~ ~~~~~~~~ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~ ~~~~ ~~~~~~~~ ~ ~~~~~~~~ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Overtype reversed fields and press ENTER...
            PF2 =Restore Defaults   PF3 =Return   PF4 =Perm Upd
```

Figure 12: FULLIST panel display

```
IF CPFKEY = 'PFØ6' & ACCSW = 'R/O' THEN DO
   MESSAGE = 'Password functions allowed only in READ/WRITE mode.'
   SIGNAL INIT ; END

IF CPFKEY = 'PFØ6' & ACCSW = 'R/W' THEN DO
   PSCREEN = 'PTFØØ'
   PWDC3 = 'New Password'
   PWDC4 = 'Verify new password'
   PC1     = '==>' ; PC2 = '==>' ; PC3 = '==>' ; pc4 = '==>'
   PWDNULL = 'X' ; CALL SETPWD
   SIGNAL INIT ; END

IF CPFKEY = 'PFØ7' THEN DO
   INCNBR = 1
   PAGENBR = 1
   LSCREEN = 'PTFØ7'
   P7TYPE = '17'
   CALL FULLIST3
   SIGNAL CKRESP ; END
```

```
IF CPFKEY = 'PFØ8' THEN DO
   INCNBR = 1
   PAGENBR = 1
   LSCREEN = 'PTFØ7'
   P7TYPE = '18'
   CALL FULLIST3
   SIGNAL CKRESP ; END

IF CPFKEY = 'PFØ9' THEN DO
   PSCREEN = 'PTFØØ'
   CALL DISPØ6
   CALL DISPØØ
   SIGNAL CKRESP ; END

/* SET EMSG OFF                                      */
/* ADDRESS COMMAND 'STATE' P62 'XEDIT *'             */
/* IF RC = Ø THEN DO                                 */
/*    SET EMSG ON ; END                              */
/* ELSE DO                                           */
/*    SET EMSG ON                                    */
/*    MESSAGE = 'Editor profile 'P62' not found.'    */
/*    SIGNAL INIT ; END                              */

IF CPFKEY = ' ' & INCID = ' ' THEN DO
   CALL CLRLST2
   FULLIST = 'Y'
/* P14     = 'Incident'      */
   PSCREEN = 'PTFØØ'
   CALL FULLIST
   SIGNAL CKRESP ; END

IF CPFKEY ¬= ' ' THEN SIGNAL INIT

IF INCID = 'OPEN' & ACCSW = 'R/W' THEN DO
   CALL BUILDNEW
   CALL DISPØØ
   SIGNAL CKRESP; END
ELSE
  IF INCID = 'OPEN' & ACCSW = 'R/O' THEN DO
     MESSAGE = 'Access is Read/Only; OPEN not allowed.'
     CALL DISPØØ
     SIGNAL CKRESP ; END
  ELSE DO
     PSCREEN='PTFØØ'
     SVWLNS=WLNS ; CALL WORK ; WLNS=SVWLNS
     CALL DISPØØ
     SIGNAL CKRESP ; END

/*****************************************************************/
/***  EDIT RESPONSES FROM PTFØ1 PANEL                          ***/
```

```
/*********************************************************************/
CKRPTFØ1:

SET EMSG OFF
'ACC' INCADDR INCFM                    /* RE-ACCESS INCIDENT DISK   */
SET EMSG ON

IF CPFKEY = 'PFØ2' THEN DO
   CALL CLRLST2
   DIRFT   = X2C(C489998583A3) || SUBSTR(DATE(J),1,2)
   CALL FULLIST
   SIGNAL CKRESP ; END

IF CPFKEY = 'PFØ3' THEN DO
   SIGNAL INIT ; END

IF CPFKEY = 'PFØ4' THEN SIGNAL EXIT

IF CPFKEY = 'PFØ8' & WLNS > Ø THEN DO
   CALL CLRLST2
   WLNS=WLNS+1
   INCNBR=INCNBR-1
   PAGENBR=PAGENBR+1
   CALL FULLIST2
   SIGNAL CKRESP ; END

/* temporary */
IF CPFKEY = 'PFØ7' THEN DO
   MESSAGE = 'PF7 temporarily disabled.'
   ZCSR = 'Z1' ; CALL DISPØ1 ; SIGNAL CKRESP ; END

/* IF CPFKEY = 'PFØ7' & PAGENBR > 1 THEN DO      */
/*    CALL CLRLST2                               */
/*    PAGENBR=PAGENBR-1                          */
/*    WLNS=PLNS.PAGENBR                          */
/*    INCNBR=PINCNBR.PAGENBR                     */
/*    CALL FULLIST2                              */
/*    SIGNAL CKRESP ; END                        */

IF CPFKEY = 'PFØ7' THEN DO
   MESSAGE = 'Already at the beginning...'
   CALL DISPØ1
   SIGNAL CKRESP ; END

IF CPFKEY = 'PFØ8' THEN
   MESSAGE = 'No more entries to display...'

IF CPFKEY ¬= ' ' THEN DO
   CALL DISPØ1
   SIGNAL CKRESP ; END
```

```
PSCREEN='PTFØ1'

/*****************************************************************/
/* CHECK FOR XEDIT, PRINT OR DELETE REQUEST OF LINE-ITEMS...      */
/*****************************************************************/

Z.1  = Z1    ;   I.1  = I1    ;   D.1  = D1
Z.3  = Z3    ;   I.2  = I2    ;   D.2  = D2
Z.5  = Z5    ;   I.3  = I3    ;   D.3  = D3
Z.7  = Z7    ;   I.4  = I4    ;   D.4  = D4
Z.9  = Z9    ;   I.5  = I5    ;   D.5  = D5
Z.11 = Z11   ;   I.6  = I6    ;   D.6  = D6
Z.13 = Z13   ;   I.7  = I7    ;   D.7  = D7
Z.15 = Z15   ;   I.8  = I8    ;   D.8  = D8
Z.17 = Z17   ;   I.9  = I9    ;   D.9  = D9
Z.19 = Z19   ;   I.1Ø = I1Ø   ;   D.1Ø = D1Ø
Z.21 = Z21   ;   I.11 = I11   ;   D.11 = D11
Z.23 = Z23   ;   I.12 = I12   ;   D.12 = D12
Z.25 = Z25   ;   I.13 = I13   ;   D.13 = D13
Z.27 = Z27   ;   I.14 = I14   ;   D.14 = D14

ZCTR = 1  ;  ICTR = 1
ZCSR = 'Z1'                          /*  DEFAULT CURSOR POSITION  */
DO 14
  IF Z.ZCTR = 'X' | Z.ZCTR = 'C' THEN DO
     ZCSR = 'Z' || ZCTR ; SVZCTR = Z.ZCTR
     Z.ZCTR = '*' ; SVINCID=INCID ; INCID=I.ICTR
     SVLNS=WLNS ; CALL WORK ; PSCREEN= 'PTFØ1' ; WLNS=SVLNS
     INCID=SVINCID ; SVZCTR = ' ' ; END

  IF Z.ZCTR = 'P' & P62 ¬= '' THEN DO
     ZCSR = 'Z' || ZCTR ; SVZCTR = Z.ZCTR
     MESSAGE = 'Printing...'
     Z.ZCTR = '*' ; SVINCID=INCID ; INCID=I.ICTR
     SVDATE   = DATE   ; SVCOMP = COMP  ; SVS    = S  ; SVVREF = VREF
     SVAUTHOR = AUTHOR ; SVVEND = VEND  ; SVSRCH = SRCH
     CALL GETPARSE ; CALL EDPRINT
     DATE   = SVDATE   ; COMP = SVCOMP  ; S    = SVS  ; VREF = SVVREF
     AUTHOR = SVAUTHOR ; VEND = SVVEND  ; SRCH = SVSRCH
     INCID=SVINCID ; SVZCTR = ' ' ; END

  IF Z.ZCTR = 'D' & ACCSW ¬= 'R/O' THEN DO
     ZCSR = 'Z' || ZCTR ; SVZCTR = Z.ZCTR
     Z.ZCTR = '*' ; SVINCID=INCID ; SVZCTR=ZCTR ; SVICTR=ICTR
     SVWLNS=WLNS ; INCID=I.ICTR ; CALL DELETE ; WLNS=SVWLNS
     INCID=SVINCID ; ZCTR=SVZCTR ; ICTR=SVICTR
     SVZCTR = ' ' ; END

  ZCTR = ZCTR+2   ;   ICTR = ICTR+1
```

```
END

Z1  = Z.1    ;   I1  = I.1
Z3  = Z.3    ;   I2  = I.2
Z5  = Z.5    ;   I3  = I.3
Z7  = Z.7    ;   I4  = I.4
Z9  = Z.9    ;   I5  = I.5
Z11 = Z.11   ;   I6  = I.6
Z13 = Z.13   ;   I7  = I.7
Z15 = Z.15   ;   I8  = I.8
Z17 = Z.17   ;   I9  = I.9
Z19 = Z.19   ;   I1Ø = I.1Ø
Z21 = Z.21   ;   I11 = I.11
Z23 = Z.23   ;   I12 = I.12
Z25 = Z.25   ;   I13 = I.13
Z27 = Z.27   ;   I14 = I.14

CALL DISPØ1
SIGNAL CKRESP

/*******************************************************************/
/***  EDIT RESPONSES FROM PTFØ2 PANEL                            ***/
/*******************************************************************/
CKRPTFØ2:
IF CPFKEY = 'PFØ4' THEN SIGNAL EXIT

IF CPFKEY ¬= 'PFØ3' THEN DO
   CALL DISPØ2 ; END
ELSE
   IF PSCREEN = 'PTFØØ' THEN DO
      CALL DISPØØ ; END

RETURN

/*******************************************************************/
/***  EDIT RESPONSES FROM PTFØ7 PANEL                            ***/
/*******************************************************************/
CKRPTFØ7:

IF CPFKEY = 'PFØ4' THEN SIGNAL EXIT

IF CPFKEY = 'PFØ7' & PAGENBR > 1 THEN DO
   PAGENBR = PAGENBR-1
   INCNBR = (PAGENBR * 14) - 13
   CALL FULLIST3
   SIGNAL CKRESP ; END
```

Editor's note: this article will be continued next month.

Steve Bernard
Senior Systems Programmer (USA)

VM news

IBM has announced the release of Lotus mailFax Version 2.1 which provides fax applications for OfficeVision, Lotus Notes, Domino, and other mail systems using VM, MVS, OS/400, and AIX. The mailFax program achieves fidelity by incorporating the Windows print-output image in the fax. All or selected messages can be archived on CD-ROM.

For further information contact your local IBM representative.

* * *

For VM/VSE and native VSE installations, Macro 4 has launched Version 4.3 of its VTAMPRINT/VSE, with TCP/IP support. With the latest version, any VSE output can be printed directly to any printer in the TCP/IP network, including printers attached to other platforms. The software has the capability to pass reports to UniQPrint, Macro 4's print management product for Unix and NT.

The VTAMPRINT/IPDS feature, which enables AFP and non-AFP printing on 3270 IPDS printers, has been improved to include support for PSF/2.

For further information contact:
Macro 4, The Orangery, Turners Hill Road, Worth, Crawley, W Sussex, RH10 4SS, UK.
Tel: (01293) 886060.
Macro 4, 35 Waterview Blvd, PO Box 292, Parsippany, NJ 07054-0292, USA.
Tel: (201) 402 8000.

* * *

IBM has announced the withdrawal of its systems management product, NetView File Transfer Program for VM Version 1, as well as numerous other VM-related System/390 products. Licensed programs for System/390 that have been withdrawn include VM/ESA Version 1 DUA, VM PWSCS, and VM/ESA 370 Feature Version 1.1.5, which has been replaced by VM/ESA Version 2.2.0 (ESA only).

For further information contact your local IBM representative.

* * *