138

VM

February 1998

update

In this issue

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

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).

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*.

Editor
Robert Burgess

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

Subscriptions and back-issues
A year's subscription to *VM Update*, comprising twelve monthly issues, costs £170.00 in the UK; $255.00 in the USA and Canada; £176.00 in Europe; £182.00 in Australasia and Japan; and £180.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1990 issue, are available separately to subscribers for £14.50 ($21.50) each including postage.

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.

Printed in England.

MXEDIT – a replacement for XMENU

The MXEDIT system is an emulation of the XMENU system. It imitates the MENUEXEC call from an EXEC. Not all the command line options and the (stack) subcommands are supported but it works exactly the same as MENUEXEC does! The main part is the MXEDIT MACRO. The current menu set of XMENU can be converted to MXEDIT layout using the MXBUILD command. Use the COPY file command with the XMENU image as input.

Organize your MXEDIT menus with the MXLIB command, the usage being like the other LIB commands.

In all the EXECs where you want to use MXEDIT, the call to procedure MXEDIT and the procedure MXEDIT itself must occur in the code. Use of MXEDIT is shown in the SAMPLE EXEC file below.

SAMPL EXEC

```
/*                                                                    */
/* Sample EXEC using the MXEDIT feature  (if XMENU is unavailable)    */
/*                                                                    */
trace OFF;
address COMMAND;
'SET FULLSCREEN SUSPEND';
 .
 .
 .
if CMS_type then 'SET CMSTYPE HT';
'STATE MENUEXEC MODULE *';
XMENU? = rc = Ø;
if XMENU? then do;
   'QUERY CMSLEVEL (LIFO'
   PULL . . release ','
   if release=5 then do;
      XMENU_Messages = "MENUEXEC"
      end;
   else do;
      XMENU_Messages = "XMENU"
      end;
   end;
if CMS_type then 'SET CMSTYPE RT';
 .
 .
```

```
 .
         if Extra? then call Prepare;
         if EMSG = "" then do;
            ALARM = "";
            if queued() > Ø then pull .;
            else push "MENUEND";
            push "CHANGE EMSG DIM";
            push "IGNORE PF13 PF14";
            push "PRTNOH PF13";
            push "PRINT PF14";
            push "MENUCMDS";
            end;
         else ALARM = "ALARM";
         if XMENU? then 'MENUEXEC USR'P.MenuNr.MENU '( DISK CLEAR SKIP'
ALARM Menu_Lib Cursor_Pos;
         else call MXEDIT 'USR'P.MenuNr.MENU '( SKIP' ALARM Menu_Lib
Cursor_Pos;
         EMSG = "";
 .
 .
 .
Prepare:
procedure expose MenuNr GEN UPD Flag_Name. Flag_Nr. Level IMSG EMSG
CURSOR Cursor_Pos PFK XMENU?;
Row = 1Ø
if XMENU? then Column = 39;
else Column = 4Ø;
if queued() > Ø then pull .;
else push "MENUEND";
select;
   when MenuNr=GEN & Level = 2 then do;
      push "CHANGE REQØ.2 DIM";
      push "CHANGE QM.2 DARK";
      push "CHANGE RETØ.2 PROT DIM";
      push "ADD END.4" Row+4 Column+1 "PROT";
      push "ADD END.5" Row+5 Column+8 "PROT";
      push "CHANGE REQØ.6 DIM";
      push "CHANGE QM.6 DARK";
      push "CHANGE RETØ.6 PROT DIM";
      push "ADD END.7" Row+7 Column+1 "PROT";
      push "END.8 '    (don''t leave it blank)  '";
      push "ADD END.8" Row+8 Column+3 "PROT";
      end;
   when MenuNr=UPD & Level = 2 then do;
      push "CHANGE REQØ.1 DIM";
 ...
      end;
   otherwise nop;
   end;
push "MENUCMDS";
```

```
return;
 .
 .
 .
MXEDIT:
do queued();
   parse pull MX_Line 1 Word . ;
   queue MX_Line;
   if Word = "ADD" then do;
      MX_VName = word(MX_Line,2);
      queue "!"MX_VName":"value(MX_VName);
      end;
   end;
'MAKEBUF'
Buffer_MXEDIT = rc;
Queued = queued();
arg MX_Name . "(" MX_Options;
if find(MX_Options,"ALARM") > Ø then MXEDIT_Options = "ALARM";
else MXEDIT_Options = "";
MX_Cursor? = find(MX_Options,"CURSOR");
if MX_Cursor? > Ø then MXEDIT_Options = MXEDIT_Options
subword(MX_Options,MX_Cursor?,2);
MX_Lib = find(MX_Options,"LIB");
if MX_Lib > Ø then do;
   MX_Lib_Name = word(MX_Options,MX_Lib+1);
  'EXECIO 255 DISKR' MX_Lib_Name 'MXLIB * Ø (FIND /¬MXEDIT:'MX_Name'*/';
   if rc = Ø then do;
      pull .;
      pull . Line_Nr Nr_of_Lines .;
      'EXECIO' Nr_of_Lines 'DISKR' MX_Lib_Name 'MXLIB *' Line_Nr '(FIFO
FINIS';
      end;
   else do;
      'EXECIO Ø DISKR' MX_Lib_Name 'MXLIB * (FINIS';
      'DROPBUF' Buffer_MXEDIT;
      ERROR = MX_Lib_Name " Member:" MX_Name;
      rc = 91;
      return;
      end;
   end;
else 'EXECIO * DISKR' MX_Name 'MXEDIT * (FIFO FINIS';
Line_Nr = queued()-Queued;
do Line_Nr;
   parse pull MX_Line 1 Indicate 2 .;
   if Indicate = "&" then do;
      parse var MX_Line . 2 MX_VName "(" MX_VDescr ")";
      MX_Line = "&"MX_VName"("MX_VDescr")"value(MX_VName);
      end;
   queue MX_Line;
   end;
```

```
'XEDIT' MX_Name "CMSUT1 (WIDTH 255 PROFILE MXEDIT)" Line_Nr
MXEDIT_Options;
ERROR = "";
do queued();
   parse pull Var "=" Value;
   interpret Var "= Value";
   end;
'DROPBUF' Buffer_MXEDIT;
if ERROR = "" then rc = 0;
else rc = 110;
return;
```

MXEDIT

```
/* MACRO : MXEDIT                                                    */
/*                                                                   */
/* Use with XEDIT or as a fullscreen manager                         */
/*                                                                   */
arg Parameter "(" . ")" Count Options;
'COMMAND Set PF4 Command Quit';
'COMMAND Set RecFM V';
'COMMAND Set LRecL *';
'COMMAND Set Trunc *';
'COMMAND Set Zone 1 *';
'COMMAND Set Verify 1 *';
'COMMAND Set Arbchar OFF';
'COMMAND Set CMDLine OFF';
'COMMAND Set Image OFF';
'COMMAND Set Linend OFF';
'COMMAND Set MSGMode OFF';
'COMMAND Set MSGLINE ON 3 1 OVERLAY';
'COMMAND Set Hex ON';
'COMMAND Set Stay ON';
'COMMAND Set CtlChar ! Escape';
'COMMAND Set CtlChar ` NoProtect Invisible';
CtlChar.N.I = "!`"; In_place.N.I = "51"X; Change.1 = "/X'51'/!`/";
'COMMAND Set CtlChar ~ Protect Invisible';
CtlChar.P.I = "!~"; In_place.P.I = "52"X; Change.2 = "/X'52'/!~/";
'COMMAND Set CtlChar < NoProtect NoHigh';
CtlChar.N.N = "!<"; In_place.N.N = "53"X; Change.3 = "/X'53'/!</";
'COMMAND Set CtlChar > Protect NoHigh';
CtlChar.P.N = "!>"; In_place.P.N = "54"X; Change.4 = "/X'54'/!>/";
'COMMAND Set CtlChar { NoProtect High';
CtlChar.N.H = "!{"; In_place.N.H = "55"X; Change.5 = "/X'55'/!{/";
'COMMAND Set CtlChar } Protect High';
CtlChar.P.H = "!}"; In_place.P.H = "56"X; Change.6 = "/X'56'/!}/";
Var_Ln = 24;
do Count;
   parse pull Line 1 Indicate +1;
```

```
   select;
      when Indicate = "&" then do;
         Var_Ln = Var_Ln+1;
         parse var Line 2 XName.Var_Ln "(" Pt "," Hl ";" LNr "," CNr ","
Lgth ")" XValue 1 VLINE.Var_Ln;
         if CNr > 1 then XLINE.LNr =
overlay(In_place.Pt.Hl,XLINE.LNr,CNr-1,1);
         if XValue = "" | XValue = XName.Var_Ln then iterate;
         XLINE.LNr = overlay(XValue,XLINE.LNr,CNr,Lgth);
         if CNr+Lgth > 81 & Lnr < 24 then do;
            LNext = LNr+1;
            XLINE.LNext =
overlay(substr(XLINE.LNr,81),XLINE.LNext,1,Lgth-80);
            end;
         end;
      when Indicate = "(" then do;
         parse var Line 2 Pt "," Hl ";" LNr "," CNr ")";
         XLINE.LNr = overlay(In_place.Pt.Hl,XLINE.LNr,CNr,1);
         end;
      when Indicate = "$" then do;
         parse var Line . "." Nr ":" XLINE.Nr;
         end;
      when Indicate = "_" then do;
         Cursor? = find(Options,"CURSOR");
         if Cursor? > 0 then do;
            Cursor_Pos = word(Options,Cursor?+1);
            if datatype(Cursor_Pos,"W") then do;
               CURLINE = Cursor_Pos%80+1;
               CURCOL = Cursor_Pos//80;
               Cursor? = 0;
               end;
            else Cursor? = 1;
            end;
         else parse var Line . ":(" CURLINE "," CURCOL ")";
         end;
      when Indicate = "*" then iterate;
      end;
   end;
do I = 25 to Var_Ln;
   'COMMAND Input' VLINE.I;
   end;
if queued() > 0 then parse pull Line;
if Line = "MENUCMDS" then do forever;
   parse pull CMD String;
   select;
      when CMD = "MENUEND" then leave;
      when CMD = "CHANGE" | CMD = "ADD" then do;
         if CMD = "CHANGE" then do;
            'COMMAND Top';
            VName = word(String,1);
```

```
            'COMMAND Locate /&' || VName || "(";
            L_rc = rc;
            if L_rc = Ø then do;
               'COMMAND Extract $CurLine$';
               parse var CurLine.3 . "(" Pt "," Hl ";" LNr "," CNr ",";
               end;
            String = subword(String,2);
            end;
         else do;
            VName = word(String,1);
            if datatype(word(String,3),"W") then do;
               LNr = word(String,2);
               CNr = word(String,3);
               String = subword(String,4);
               end;
            else do;
               Offset = word(String,2);
               String = subword(String,3);
               LNr = Offset%8Ø+1;
               CNr = Offset//8Ø;
               end;
            L_rc = Ø;
            end;
         if L_rc = Ø then do;
            do I = 1 to words(String);
               Spec = translate(word(String,I));
               select;
                  when abbrev("PROTECT",Spec,4) then Pt = "P";
                  when abbrev("UNPROTECT",Spec,6) then Pt = "N";
                  when Spec = "BRIGHT" then Hl = "H";
                  when Spec = "DIM" then Hl = "N";
                  when Spec = "DARK" then Hl = "I";
                  otherwise iterate;
                  end;
               end;
            if CMD = "CHANGE" then do;
               if CNr > 1 then XLINE.LNr =
overlay(In_place.Pt.Hl,XLINE.LNr,CNr-1,1);
               end;
            else do;
               Var_Ln = Var_Ln+1;
               parse pull "!" Name ":" Value;
               if Name = VName then do;
                  Lgth = length(Value);
                  VLINE.Var_Ln =
"&"VName"("Pt","Hl";"LNr","CNr","Lgth")"Value;
                  if Value ¬= "" & Value ¬= Name then XLINE.LNr =
overlay(Value,XLINE.LNr,CNr,Lgth);
```

```
                  end;
               else VLINE.Var_Ln = "&"VName"("Pt","Hl";"LNr","CNr",ØØ)";
               'COMMAND Bottom';
               'COMMAND Input' VLINE.Var_Ln;
               if CNr > 1 then XLINE.LNr =
overlay(In_place.Pt.Hl,XLINE.LNr,CNr-1,1);
               end;
            end;
         else call ERROR  VName "not found";
         end;
      otherwise do;
         if pos("'",String) = Ø then iterate;
         'COMMAND Top';
         'COMMAND Locate /&' || CMD || "(";
         if rc = Ø then do;
            'COMMAND Extract $CurLine$';
            parse var CurLine.3 Before ";" LNr "," CNr ",";
            Lgth = length(String)-2;
            XValue = strip(String,"B","'");
            'COMMAND Replace' Before";"LNr","CNr","Lgth")"XValue;
            XLINE.LNr = overlay(XValue,XLINE.LNr,CNr,Lgth);
            end;
         else call ERROR  CMD "not found";
         end;
      end;
   end;
'COMMAND Top';
do I = 1 to 24;
   if XLINE.I = "" then XLINE.I = overlay(In_place.P.N,XLINE.I,1,1);
   'COMMAND Input' XLINE.I;
   end;
if Cursor? then do;
   'COMMAND Locate :24';
   'COMMAND Locate /&' || Cursor_Pos || "(";
   if rc = Ø then do;
      'COMMAND Extract $CurLine$';
      parse var CurLine.3 . ";" CURLINE "," CURCOL ",";
      end;
   end;
'COMMAND Cursor Screen' CURLINE CURCOL;
'COMMAND Top';
do I = 1 to 6;
   'COMMAND Change' Change.I ":25 *";
   end;
if find(Options,"ALARM") > Ø then do;
   'COMMAND Set MSGMode ON';
   'COMMAND EMSG';
   'COMMAND Set MSGMode OFF';
```

```
   end;
do I = 1 to 24;
   'COMMAND Locate :' || I;
   'COMMAND Extract $CurLine$';
   'COMMAND Set Reserved' I 'Nohigh' CurLine.3;
   end;
'COMMAND Extract $LScreen$';
do I = 25 to LScreen.1;
   'COMMAND Set Reserved' I 'Nohigh' CtlChar.P.N;
   end;
'COMMAND Read All Tag';
All = queued();
'COMMAND Extract $Cursor$';
queue "CURSOR=" || (Cursor.1-1)*8Ø+Cursor.2;
queue "CURLINE=" || Cursor.1;
'COMMAND Locate :24';
do forever;
   'COMMAND Locate /;'Cursor.1',';
   if rc = 2 then leave;
   'COMMAND Extract $CurLine$';
   parse var CurLine.3 2 Name "(" . ";" . "," CNr "," Length ")";
   if CNr <= Cursor.2 & Cursor.2 <= CNr+Length then do;
      queue "CURFNAM=" || Name;
      leave;
      end;
   end;
do All;
   parse pull Tag Nr . 1 . LNr CNr Value;
   select;
      when Tag = "ETK" then queue "PFK=ENTER";
      when Tag = "PFK" then queue "PFK=PF" || right(Nr,2,"Ø");
      when Tag = "RES" then do;
         'COMMAND Locate :24';
         'COMMAND Locate /;'LNr','CNr',';
         if rc ¬= Ø then iterate;
         'COMMAND Extract $CurLine$';
         parse var CurLine.3 2 Name "(" . ";" . "," . "," Lgth ")";
         queue Name || "=" || substr(Value,1,Lgth);
         end;
      otherwise nop;
      end;
   end;
'COMMAND Quit';
exit;
ERROR:
parse source . . Source_N .;
arg Text;
queue "ERROR=" Source_N": " Text;
'COMMAND Quit';
exit;
```

MXLIB EXEC

```
/*********************************************************************/
/*  MXLIB   :                                                        */
/*                                                                   */
/*  Create and modify MXEDIT image libraries.                        */
/*                                                                   */
/*********************************************************************/
parse source . . Source_Name .;
MX_Type = Source_Name;
arg CMD MX_Lib MX_Members;
if MX_Members = "" ,
 | (CMD ¬= "ADD" & CMD ¬= "DEL" & CMD ¬= "REP" & CMD ¬= "GEN") then do;
   say Source_Name "Format:";
   say " ";
   say Source_Name "ADD|DEL|REP|GEN library member(s)"
   say " ";
   exit 1;
   end;
CMS_type = cmsflag("CMSTYPE")
if CMS_type then 'SET CMSTYPE HT';
'MAKEBUF';
Buffer = rc;
'LISTFILE' MX_Lib MX_Type "* (NOHEADER ALLOC FIFO";
if rc = Ø then do;
   pull . . MX_Lib_Mode . . MX_Lib_Recs .;
   do queued();
      pull .;
      end;
   MX_Lib_Name = MX_Lib MX_Type MX_Lib_Mode;
   'STATEW' MX_Lib_Name;
   if CMD = "GEN" & rc ¬= Ø then MX_Lib_Name = MX_Lib MX_Type "A";
   end;
else MX_Lib_Name = MX_Lib MX_Type "A";
Lib? = rc = Ø;
if CMD ¬= "GEN" & ¬Lib? then call Error 8,"Library:" MX_Lib "not
found!";
if CMD = "GEN" & Lib? then call Error 8,"Library:" MX_Lib "already
exist!";
Date_Time = date("USA") time();
Members = words(MX_Members);
do I = 1 to Members;
   if CMD ¬= "DEL" then do;
      call Header word(MX_Members,I);
      parse var result . . . Header.I Recs.I 1 Member.I (Header.I);
      end;
   else Member.I = word(MX_Members,I);
   end;
```

```
if CMD = "GEN" then do;
   Begin = Members+1;
   do I = 1 to Members;
      'EXECIO 1 DISKW' MX_Lib_Name I ,
       "(STRING" Header.I Begin Recs.I Date_Time;
      Begin = Begin+Recs.I;
      end;
   'EXECIO Ø DISKW' MX_Lib_Name "(FINIS";
   do I = 1 to Members;
      'COPYFILE' Member.I MX_Lib_Name "(APPEND";
      end;
   end;
else do;
   Temp_Lib_Name = MX_Lib "TØ" || left(MX_Type,6) word(MX_Lib_Name,3);
   'EXECIO * DISKR' MX_Lib_Name "(FINIS LIFO AVOID /¬MXEDIT:/";
   pull Nr .;
   pull .;
   Stock = Nr-1;
   'EXECIO' Stock 'DISKR' MX_Lib_Name " 1 (FIFO FINIS";
   do I = 1 to Stock;
      pull Stock.I Start_Rec.I Nr_Rec.I DT.I 1 ":" Stock_Name.I "*";
      interpret "Flag."Stock_Name.I "=" I;
      DT.I = strip(DT.I);
      end;
   select;
      when CMD = "DEL" then do;
         New_Stock = Stock;
         do I = 1 to Members;
            Candidate = Member.I;
            if symbol('Flag.Candidate') = "VAR" then do;
               Flag.Candidate = Ø;
               New_Stock = New_Stock-1;
               end;
            else call Write "Member:" Candidate "not found!";
            end;
         if New_Stock = Stock then call Error 2;
         'ERASE' Temp_Lib_Name;
         Begin = New_Stock+1;
         J = Ø;
         do I = 1 to Stock;
            Candidate = Stock_Name.I;
            if Flag.Candidate = Ø then iterate;
            J = J+1;
            'EXECIO 1 DISKW' Temp_Lib_Name J ,
             "(STRING" Stock.I Begin Nr_Rec.I DT.I;
            Begin = Begin+Nr_Rec.I;
            end;
         'EXECIO Ø DISKW' Temp_Lib_Name "(FINIS";
```

```
      cmds = Ø;
      do I = 1 to Stock;
         Candidate = Stock_Name.I;
         if Flag.Candidate = Ø then iterate;
         prev = cmds;
         cmds = cmds+1;
         if cmds = 1 | Start_Rec.I ¬= New_Start.prev then do;
            COPY.cmds = 'COPYFILE' MX_Lib_Name Temp_Lib_Name ,
             "(FROM" Start_Rec.I "FOR" Nr_Rec.I "APPEND";
            New_Start.cmds = Start_Rec.I+Nr_Rec.I;
            New_Recs.cmds = Nr_Rec.I;
            end;
         else do;
            cmds = prev;
            New_Start.cmds = Start_Rec.I+Nr_Rec.I;
            New_Recs.cmds = New_Recs.cmds+Nr_Rec.I;
         COPY.cmds = subword(COPY.cmds,1,1Ø) New_Recs.cmds "APPEND";
            end;
         end;
      do I = 1 to cmds;
         COPY.I;
         end;
      end;
   when CMD = "ADD" then do;
      New_Stock = Stock;
      do I = 1 to Members;
         Candidate = word(Member.I,1);
         if symbol('Flag.Candidate') = "VAR" then do;
            Insert.I = Ø;
            call Write "Member:" Candidate "already exist!";
            end;
         else do;
            Insert.I = I;
            New_Stock = New_Stock+1;
            end;
         end;
      if New_Stock = Stock then call Error 2;
      'ERASE' Temp_Lib_Name;
      Begin = New_Stock+1;
      J = Ø;
      do I = 1 to New_Stock;
         if I <= Stock then do;
            'EXECIO 1 DISKW' Temp_Lib_Name I ,
             "(STRING" Stock.I Begin Nr_Rec.I DT.I;
            Begin = Begin+Nr_Rec.I;
            end;
         else do;
            do until Insert.J > Ø;
               J = J+1;
               end;
```

```
            'EXECIO 1 DISKW' Temp_Lib_Name I ,
             "(STRING" Header.J Begin Recs.J Date_Time;
            Begin = Begin+Recs.J;
            end;
         end;
      'EXECIO Ø DISKW' Temp_Lib_Name "(FINIS";
      J = Ø;
      cmds = Ø;
      do I = 1 to New_Stock;
         prev = cmds;
         cmds = cmds+1;
         if I <= Stock then
         if cmds = 1 | Start_Rec.I ¬= New_Start.prev then do;
            COPY.cmds = 'COPYFILE' MX_Lib_Name Temp_Lib_Name ,
             "(FROM" Start_Rec.I "FOR" Nr_Rec.I "APPEND";
            New_Start.cmds = Start_Rec.I+Nr_Rec.I;
            New_Recs.cmds = Nr_Rec.I;
            end;
         else do;
            cmds = prev;
            New_Recs.cmds = New_Recs.cmds+Nr_Rec.I;
         COPY.cmds = subword(COPY.cmds,1,1Ø) New_Recs.cmds "APPEND";
            New_Start.cmds = Start_Rec.I+Nr_Rec.I;
            end;
         else do;
            do until Insert.J > Ø;
               J = J+1;
               end;
            COPY.cmds = 'COPYFILE' Member.J Temp_Lib_Name "(APPEND";
            New_Start.cmds = Ø;
            end;
         end;
      do I = 1 to cmds;
         COPY.I;
         end;
      end;
   when CMD = "REP" then do;
      New_Stock = Stock;
      do I = 1 to Members;
         Candidate = word(Member.I,1);
         if symbol('Flag.Candidate') = "VAR" then Flag.Candidate = Ø;
         else do;
            call Write "Member:" Candidate "not exist, inserted!";
            New_Stock = New_Stock+1;
            end;
         end;
      Begin = New_Stock+1;
      J = Ø;
```

```
      do I = 1 to Stock;
         Candidate = Stock_Name.I;
         if Flag.Candidate = Ø then iterate I;
         J = J+1;
         'EXECIO 1 DISKW' Temp_Lib_Name J ,
          "(STRING" Stock.I Begin Nr_Rec.I DT.I;
         Begin = Begin+Nr_Rec.I;
         end;
      K = Ø;
      do I = J+1 to New_Stock;
         K = K+1;
         'EXECIO 1 DISKW' Temp_Lib_Name I ,
          "(STRING" Header.K Begin Recs.K Date_Time;
         Begin = Begin+Recs.K;
         end;
      'EXECIO Ø DISKW' Temp_Lib_Name "(FINIS";
      J = Ø;
      cmds = Ø;
      do I = 1 to Stock;
         Candidate = Stock_Name.I;
         if Flag.Candidate = Ø then iterate I;
         else J = Flag.Candidate;
         prev = cmds;
         cmds = cmds+1;
         if cmds = 1 | Start_Rec.J ¬= New_Start.prev then do;
            COPY.cmds = 'COPYFILE' MX_Lib_Name Temp_Lib_Name ,
             "(FROM" Start_Rec.J "FOR" Nr_Rec.J "APPEND";
            New_Start.cmds = Start_Rec.J+Nr_Rec.J;
            New_Recs.cmds = Nr_Rec.J;
            end;
         else do;
            cmds = prev;
            New_Start.cmds = Start_Rec.J+Nr_Rec.J;
            New_Recs.cmds = New_Recs.cmds+Nr_Rec.J;
         COPY.cmds = subword(COPY.cmds,1,1Ø) New_Recs.cmds "APPEND";
            end;
         end;
      do I = 1 to cmds;
         COPY.I;
         end;
      do I = 1 to Members;
         'COPYFILE' Member.I Temp_Lib_Name "(APPEND";
         end;
      end;
   end;
'EXECIO * DISKR' Temp_Lib_Name "(LIFO AVOID /¬MXEDIT:/";
pull Nr .;
pull .;
```

```
   Stock = Nr-1;
   'EXECIO' Stock 'DISKR' Temp_Lib_Name " 1 (FIFO FINIS";
   do I = 1 to Stock;
      pull . ":" Stock_Name "*" Start_Rec .;
      'EXECIO 1 DISKR' Temp_Lib_Name Start_Rec "(LIFO FINIS";
      pull . ":" Start_Name "*" .;
      if Start_Name = Stock_Name then iterate;
      else do;
         call Write "Error: Check of temporary" Temp_Lib_Name;
         call Write ,
          "Record:" Start_rec "does not contain" Stock_Name "header";
         call Error 16;
         end;
      end;
   'ERASE' MX_Lib_Name;
   'RENAME' Temp_Lib_Name MX_Lib_Name;
   end;
'DROPBUF' Buffer;
if CMS_type then 'SET CMSTYPE RT';
exit;
Error:
if CMS_type then 'SET CMSTYPE RT';
if arg(2,'exist') then say arg(2);
'DROPBUF' Buffer;
exit arg(1);
Write:
if CMS_type then 'SET CMSTYPE RT';
if arg(1,'exist') then say arg(1);
if CMS_type then 'SET CMSTYPE HT';
return;
Header:
'MAKEBUF';
MX_Name = arg(1);
'LISTFILE' MX_Name "MXEDIT * (NOHEADER ALLOC FIFO";
if rc = 28 then call Error 4,"File:" MX_Name "not found!";
else pull . . MX_Mode . . MX_Recs .;
'EXECIO 1 DISKR' MX_Name "MXEDIT" MX_Mode "(LIFO FINIS";
pull ":" Name "*" 1 Header;
if Name ¬= MX_Name then call Error 4,"File:" MX_Name "contain:" Name
"!";
'DROPBUF';
return MX_Name "MXEDIT" MX_Mode overlay("¬",Header) MX_Recs;
```

MXBUILD EXEC

```
/*********************************************************************/
/*  MXBUILD :                                                        */
/*                                                                   */
/* Read an Assembler DSECT image of a (XMENU) menu and convert it to  */
/* an MXEDIT (usable by MXEDIT) menu.                                */
```

```
/*                                                                    */
/**********************************************************************/
parse source . . S_N .;
arg F_N .;
if F_N = "" | F_N = "?" then do;
   say " Format: " S_N "filename";
   exit 1;
   end;
In_place.N.I = "51"X;
In_place.P.I = "52"X;
In_place.N.N = "53"X;
In_place.P.N = "54"X;
In_place.N.H = "55"X;
In_place.P.H = "56"X;
do I = 1 to 24;
   XLine.I = copies(" ",8Ø);
   end;
Protection.PROT = "P"; Protection.UNPROT = "N";
Visibility.BRIGHT = "H"; Visibility.DIM = "N"; Visibility.DARK = "I";
Field_Index = Ø; Variable_Index = Ø; named = Ø;
do until IOend;
   'EXECIO 1 DISKR' F_N "COPY * (VAR LINE";
   IOend = rc = 2;
   if rc ¬= Ø & rc ¬= 2 then do;
      say "EXECIO: rc =" rc;
      exit rc;
      end;
   parse var LINE Name . 1Ø DC 16 C 17 Nothing "'" Field "'" Comment "("
Lnr "," Cnr ")";
   do while substr(Comment,1,1) = "'";
      Quote = pos("'",Comment,2);
      Field = Field || delstr(Comment,Quote);
      Comment = substr(Comment,Quote+1);
      end;
   Lnr = strip(Lnr,"L","Ø");
   Cnr = strip(Cnr,"L","Ø");
   if Name = "*" then do;
      parse var Nothing Prot Visy .;
      Pt.Field_Index = Protection.Prot;
      Hl.Field_Index = Visibility.Visy;
      end;
   else select;
      when IOend then if named then do;
         Length.Variable_Index = 25*8Ø -,
          V_Lnr.Variable_Index*8Ø-V_Cnr.Variable_Index;
         end;
      when DC ¬= "DC" then iterate;
      when C = "X" then do;
         if Comment = "SFA..." then do;
            if named then do;
```

```
               named = Ø;
               Length.Variable_Index = Lnr*8Ø+Cnr -,
                V_Lnr.Variable_Index*8Ø-V_Cnr.Variable_Index;
               end;
            Field_Index = Field_Index+1;
            Lnr.Field_Index = Lnr; Cnr.Field_Index = Cnr;
            end;
         else if find(translate(Comment),"MENU HEADER") > Ø &
find(translate(Comment),"CURSOR AT") > Ø then do;
            Lnr.Ø = Lnr; Cnr.Ø = Cnr;
            end;
         end;
      when C = "C" then do;
         if named then Value.Variable_Index = Value.Variable_Index ||
Field;
         if Name ¬= "" then do;
            named = 1;
            Variable_Index = Variable_Index+1;
            Name.Variable_Index = Name;
            V_Pt.Variable_Index = Pt.Field_Index;
            V_Hl.Variable_Index = Hl.Field_Index;
            Field_Index = Field_Index-1;
            V_Lnr.Variable_Index = Lnr;
            V_Cnr.Variable_Index = Cnr;
            Value.Variable_Index = Field;
            end;
         XLine.Lnr = overlay(Field,XLine.Lnr,Cnr);
         end;
      end;
   end;
do I = 1 to Field_Index;
   parse value Pt.I Hl.I Lnr.I Cnr.I with Pt Hl LNr CNr;
   XLINE.LNr = overlay(In_place.Pt.Hl,XLINE.LNr,CNr,1);
   end;
CMS_type = cmsflag("CMSTYPE")
if CMS_type then 'SET CMSTYPE HT';
'ERASE' F_N "MXEDIT A";
if CMS_type then 'SET CMSTYPE RT';
STRING = "*MXEDIT:"F_N"*";
'EXECIO 1 DISKW' F_N "MXEDIT A (VAR STRING";
STRING = "_CURSOR:("Lnr.Ø","Cnr.Ø")";
'EXECIO 1 DISKW' F_N "MXEDIT A (VAR STRING";
do I = 1 to 24;
   if I < 24 then if length(XLine.I) > 8Ø then do;
      J = I+1;
      XLine.J = overlay(substr(XLine.I,81),XLine.J);
      XLine.I = substr(XLine.I,1,8Ø);
      end;
   if I = 1 then if substr(XLine.1,1,1) = " " then XLine.1 =
substr(XLine.1,2);
```

```
   else XLine.1 = delstr(XLine.1,pos("   ",XLine.1),1);
   STRING = "$LINE."I":"XLine.I;
   'EXECIO 1 DISKW' F_N "MXEDIT A (VAR STRING";
   end;
do I = 1 to Variable_Index;
   STRING = "&"Name.I ||,
    "("V_Pt.I","V_Hl.I";"V_Lnr.I","V_Cnr.I","Length.I")"Value.I;
   'EXECIO 1 DISKW' F_N "MXEDIT A (VAR STRING";
   end;
'EXECIO Ø DISKW' F_N "MXEDIT A (FINIS";
exit;
```

Anton Altbauer (Germany)

The CMS CALL interface

GENERAL DESCRIPTION

The CMS CALL (CMSCALL) interface is useful for application programmers developing programs in Assembler, COBOL, or PL/I. The interface is required for program conversion from DOS/VSE to VM/CMS because of language processor restrictions under CMS, which must be simulated using CMS commands.

CMSCALL enables the following actions to be performed:

- Passing different commands to CMS
- Starting REXX EXECs
- Invoking XEDIT for file processing.

CMSCALL is written in Assembler and runs under CMS with VM/SP Release 5.

CMSCALL USAGE

CMSCALL processes only one parameter, which identifies an area holding a CMS command. The first two bytes of the area must contain the area length in binary code. The remaining bytes contain a CMS

command, which is executed as if it were typed in the command line of a terminal. The CMSCALL return code is the original SVC 202 return code.

Because of memory allocation conflicts, the language processors may be started dynamically only from Assembler application programs that are loaded as nucleus extensions.

Examples of CMSCALL usage are:

- File definitions from an Assembler program:

```
         CALL  CMSCALL,(AREA)
         .
         .
         .
AREA     DC    AL2(L'CMSCMD)
CMSCMD   DC    C'FI IN DISK A A A'
```

- Sorting a file in XEDIT, using XSORT XEDIT, which contains the XEDIT command SORT from a COBOL program:

```
WORKING-STORAGE SECTION.

Ø1  AREA
    49  AREA-LEN PIC S9(4) COMP VALUE +2Ø.
    49  CMS-CMD  PIC X(2Ø) VALUE 'X A A A (PROF XSORT'

PROCEDURE DIVISION.

CALL 'CMSCALL' USING AREA.
```

- Delaying a program in PL/I by starting DELAY EXEC, which contains CP command SLEEP:

```
DCL AREA CHAR (2Ø) VAR AUTO;
DCL CMSCALL ENTRY OPTIONS (ASM INTER);
 .
 .
 .
AREA = 'DELAY';
CALL CMSCALL (AREA);
```

CMSCALL ASSEMBLE

```
*************************************************************************
****                                                  ***         ****
```

```
**** CMSCALL            CMS call interface            ***         ****
****                                                  ***         ****
*************************************************************************
****   SIZE 00041  VER 1.0 MOD 00                                  ****
*************************************************************************
*                                                                       *
CMSCALL  CSECT
         SAVE  (14,12)
         BALR  2,0
         USING *,2
         ST    13,SA+4
         LA    13,SA
         B     AROUNDHP
BUILTIN  DC    C'   CMS CALL interface  - using '
HELP     DC    C'->     CALL (<var char>) /*17:08:41 13/09/97 DG"97*/ '
AROUNDHP EQU   *
         L     1,0(1)
         ICM   1,8,=X'00'
         LH    15,0(1)
         LA    1,2(1)
         ST    1,VSTART
         AR    1,15
         ST    1,VEND
         LA    1,PLIST
         LA    0,EXTPLIST
         ICM   1,8,=X'02'
         SVC   202
         DC    AL4(1)
         L     13,4(13)
         RETURN (14,12),,RC=(15)
SA       DC    18F'0'
PLIST    EQU   *
         DC    CL8'CMS'
EXTPLIST EQU   *
         DC    A(PLIST)
VSTART   DC    A(0)
VEND     DC    A(0)
         DC    A(0)
         END   CMSCALL
```

Dobrin Goranov
Information Services Co (Bulgaria)

A compressed dump/restore – part 2

This month we complete the code for a dump/restore facility for which the dump device is a CMS formatted mini-disk.

VREST ASSEMBLER

```
*        THE RESTORE PROGRAM
         LCLC  &NUMBUF
         LCLA  &I,&E
         AIF   ('&SYSPARM' EQ '').OMITTED
&NUMBUF  SETC  '&SYSPARM'
         AGO   .GOON
.OMITTED ANOP
&NUMBUF  SETC  '8'
.GOON    ANOP
&E       SETA  &NUMBUF
         AIF ((&E LT 1) OR (&E GT 64)).OMITTED
         EJECT
         PRINT NOGEN
VREST    CSECT
NUMBUF   EQU   &NUMBUF
         USING VREST,R15
         STNSM SYSMASK,X'FE' DISABLE EXTERNAL INTERRUPTS
         STCTL CØ,CØ,CTRLØ
         MVC   SAVCTRLØ,CTRLØ     SAVE CONTROL REGISTER Ø CONTENTS
         OI    CTRLØ+3,X'Ø2' TURN BIT 3Ø IN CTRL REGØ ON TO ENABLE IUCV
*                             EXTERNAL INTERRUPTS WHEN NEEDED
         LCTL  CØ,CØ,CTRLØ
         DROP  R15
         EJECT
         STM   R14,R12,12(R13)
         LR    R12,R15
         USING VREST,R12     BASE REGISTER
         USING NUCON,RØ
         USING DEVSECT,R6
         USING IPARML,R7
         USING ADTSECT,R8
         USING MULTBUFF,R11
         LA    RØ,SAVE
         ST    RØ,8(R13)
         ST    R13,SAVE+4
         LR    R13,RØ
         CLI   8(R1),X'FF'   ANY ARGUMENTS?
         BNE   PRESENT       YES
         LA    R15,1ØØØ      NO
```

```
         ST    R15,RETCODE
         MVC   ERRORTXT,=CL8Ø'DISK MODE NOT SPECIFIED'
         BAL   R3,WRTERM
         B     EXIT          RESTORE ORIGINAL CONDITIONS & REGS. EXIT
PRESENT  L     R8,IADT       ADDRESS OF THE FIRST ENTRY IN ADT
MODETEST CLC   ADTM,8(R1)    ACTIVE DISK TABLE ENTRY FOUND?
         BE    FOUND         YES
         ICM   R8,15,ADTPTR  NO. NULL ENTRY IN ACTIVE DISK TABLE?
         BNZ   MODETEST      NO. (R8)=ADDRESS OF THE NEXT ENTRY IN ADT
         LA    R15,1ØØ4      YES. DISK MODE INVALID
         ST    R15,RETCODE
         MVC   ERRORTXT,=CL8Ø'DISK MODE INVALID'
         BAL   R3,WRTERM
         B     EXIT          RESTORE ORIGINAL CONDITIONS & REGS. EXIT
FOUND    TM    ADTFLG1,ADTFRO     IS THIS DISK A R/W MINI-DISK?
         BZ    RWDISK        YES
         LA    R15,1ØØ8      NO. R/O MINI-DISK. THIS IS AN ERROR
         ST    R15,RETCODE
         MVC   ERRORTXT,=CL8Ø'DISK TO BE RESTORED MUST BE R/W MINI-DISK'
         BAL   R3,WRTERM
         B     EXIT          RESTORE ORIGINAL CONDITIONS & REGS. EXIT
RWDISK   L     R6,ADTDTA     (R6)=RELATED DEVICE TABLE ENTRY ADDRESS
         LH    RØ,DEVADDR    VIRTUAL DISK ADDRESS
         STH   RØ,VDEVADDR
         L     RØ,ADTDBSIZ   (RØ)=VIRTUAL DISK BLOCKSIZE
         ST    RØ,BLKSIZE
         DROP  R6
         DROP  R8
         XC    PARMLIST,PARMLIST  CLEAR THE IUCV PARAMETER LIST
         LA    R7,PARMLIST   NOW R7 IS IUCV PARAMETER LIST BASE REG
DECLARE  IUCV  DCLBFR,PRMLIST=PARMLIST,BUFFER=INTPLIST,CONTROL=NO
         BZ    DCLBFROK      DECLARE BUFFER COMPLETED SUCCESSFULLY?
         BC    1,DCLBFR3     NO. CC=3 MEANS ERROR READING DIRECTORY
         BC    2,NEVER       CC=2. SHOULD NEVER OCCUR
         CLI   IPRCODE,X'13' IUCV EXT INTERRUPT BUFFER ALREADY DEFINED?
         BE    ABENDED       YES. MAY BE THE PROGRAM HAS ABENDED BEFORE
*                            THIS RUN
NEVER    MVC   ERRORTXT,=CL8Ø'DECLARE BUFFER FAILED'  CC=2
*                            *BLOCKIO ERROR RET CODE STORED IN IPRCODE
         BAL   R3,WRTERM
         XR    R15,R15
         IC    R15,IPRCODE   PUT *BLOCKIO ERROR RETURN CODE IN R15
         ST    R15,RETCODE
         B     EXIT
DCLBFR3  MVC   ERRORTXT,=CL8Ø'ERROR READING DIRECTORY'
         BAL   R3,WRTERM
         LA    R15,1Ø12
         ST    R15,RETCODE
         B     EXIT
ABENDED  IUCV  RTRVBFR
```

```
         B     DECLARE
DCLBFROK XC    PARMLIST,PARMLIST       CLEAR THE PARAMETER LIST
         MVC   IPUSER(4),BLKSIZE       * FILL THE PARAMETER LIST WITH
         MVC   IPUSER+4(4),OFFSET      * PARMS NEEDED TO CONNECT TO
         MVC   IPUSER+8(2),VDEVADDR    * *BLOCKIO SYSTEM SERVICE
         MVC   IPVMID,=CL8'*BLOCKIO'
         MVC   IPMSGLIM,=X'ØØFF'  MAXIMUM ØØ255 OUTSTANDING MESSAGES TO
*                                 *BLOCKIO ALLOWED
         IUCV  CONNECT,CONTROL=NO,PRMDATA=YES,PRTY=NO,QUIESCE=NO,      *
               PRMLIST=PARMLIST
         BZ    CONNWAIT      CONNECT FUNCTION SUCCESSFULLY STARTED?
         MVC   ERRORTXT,=CL8Ø'CONNECTION NOT ESTABLISHED'  NO. CC ¬= Ø.
*                            *BLOCKIO ERROR RET CODE STORED IN IPRCODE
         BAL   R3,WRTERM
         XR    R15,R15
         IC    R15,IPRCODE   PUT *BLOCKIO ERROR RETURN CODE IN R15
         ST    R15,RETCODE
         B     RTRVBFR       TERMINATE USE OF IUCV
CONNWAIT HNDEXT SET,IUCVEXIT ACTIVATE IUCV EXTERNAL INTERRUPT HANDLER
         MVC   NEXTADDR,=AL3(CONNCMPL)
         LPSW  WAITPSW       ENABLE EXTERNAL INTERRUPTS AND WAIT FOR
*                            CONNECTION COMPLETE INTERRUPTION
CONNCMPL LA    R7,INTPLIST   NOW R7 IS IUCV INTERRUPT BUFFER BASE REG
         CLI   IPTYPE,X'Ø2'  CONNECTION COMPLETED?
         BE    CONNOK        YES
         CLI   IPUSER,X'Ø4'  NO, PATH SEVERED BY *BLOCKIO. CHECK IF THE
*                            CONNECTION WAS ESTABLISHED BEFORE THIS RUN
         BNE   CONNERR       IT WAS NOT ESTABLISHED. THIS IS AN ERROR
         XC    PARMLIST,PARMLIST  CONNECTION ALREADY ESTABLISHED. CLEAR
*                            IUCV PARAMETER LIST AND SEVER THE PATH
         LA    R7,PARMLIST   NOW R7 IS IUCV PARAMETER LIST BASE REG
         IUCV  SEVER,PRMLIST=PARMLIST,ALL=NO,PATHID=PATHID
         BZ    DECLARE       IF SEVER SUCCESSFUL TRY TO CONNECT AGAIN
         MVC   ERRORTXT,=CL8Ø'SEVER FAILED' CC=1
*                            IUCV ERROR RETURN CODE STORED IN IPRCODE
         BAL   R3,WRTERM
         XR    R15,R15
         IC    R15,IPRCODE   PUT IUCV ERROR RETURN CODE IN R15
         ST    R15,RETCODE
         B     RTRVBFR       TERMINATE IUCV USE AND EXIT
CONNERR  MVC   ERRORTXT,=CL8Ø'PATH SEVERED BY *BLOCKIO'
         BAL   R3,WRTERM
         XR    R15,R15
         IC    R15,IPUSER          PUT *BLOCKIO ERROR CODE IN R15
         ST    R15,RETCODE
         B     RTRVBFR             TERMINATE USE OF IUCV
CONNOK   MVC   PATHID,IPPATHID     SAVE THE PATH ID
         MVC   STRBLOCK,IPUSER     START BLOCK RETURNED FROM *BLOCKIO
         MVC   ENDBLOCK,IPUSER+4   END BLOCK RETURNED FROM *BLOCKIO
         OPEN  (INPDSK,(INPUT))    OPEN THE DUMP FILE
```

```
         L     R6,STRBLOCK
         L     R5,ENDBLOCK
         SR    R5,R6
         A     R5,=F'1'      (R5)=NUMBER OF BLOCKS ON MINI-DISK
         L     R2,=A(RDW)    READ THE DUMP HEADER RECORD TO CHECK IF
*                            THE CURRENT VALUES OF THE DISK BLOCK SIZE
*                            AND THE DISK SIZE IN BLOCKS ARE VALID
         GET   INPDSK,(R2)
         L     RØ,BLKSIZE    DISK BLOCK SIZE IN RØ
         C     RØ,4(R2)      IS IT VALID?
         BNE   BLKERR        NO. TYPE ERROR MESSAGE AND EXIT
         C     R5,8(R2)      IS THE DISK SIZE IN BLOCKS VALID?
         BNE   SIZEERR       NO. TYPE ERROR MESSAGE AND EXIT
         LA    R4,Ø
         D     R4,=A(NUMBUF) DIVIDE THE NUMBER OF BLOCKS THE MINIDISK
*                            CONTAINS BY THE NUMBER OF DATA BUFFERS
*                            (R4)=REMAINDER,(R5)=QUOTIENT
         LTR   R5,R5         IS THE QUOTIENT ZERO?
         BZ    RESTTEST      YES
QUOTLOOP LA    R7,PARMLIST   NOW R7 IS IUCV PARAMETER LIST BASE REG
         XC    PARMLIST,PARMLIST  CLEAR IUCV PARAMETER LIST
         LA    RØ,3
         ST    RØ,IPTRGCLS   MULTIPLE REQUEST TO *BLOCKIO
         LA    R1Ø,NUMBUF    NUMBER OF BUFFERS TO BE USED
         ST    R1Ø,IPRMMSG1  STORE IT IN PARMLIST
         LA    R11,MBUFLST   ADDRESS OF PLIST DEFINING THE BLOCKS TO BE
*                            PROCESSED AND THE BUFFERS TO BE USED
         ST    R11,IPRMMSG2  STORE IT IN PARMLIST
         L     R9,=A(DATABUFS)
FILLQUOT L     R2,=A(INPBUFF)     INPUT BUFFER ADDRESS USED BY GET
         GET   INPDSK,(R2)   READ A RECORD INTO THE BUFFER
         ST    R9,OUTBUFAD   ADDRESS TO WHICH DMKDDC IS TO MOVE THE
*                            THE DECOMPACTED STRING
         LA    R1,DPLIST     ADDRESS OF DECOMPACTION ROUTINE'S PLIST
         L     R15,DMKDDCAD  DECOMPACTION ROUTINE'S ENTRY ADDRESS
         BALR  R14,R15       CALL THE DECOMPACTION ROUTINE
         ICM   R15,15,RETCODE     DECOMPACTION ROUTINE'S RETCODE IN R15
         BZ    DCOMPQOK      DECOMPACTION O.K.
         C     R15,=F'2'     DECOMPACTION RETURN CODE=2?
         BNE   DCOMPERR      NO. DECOMPACTION ERROR. SHOULD NOT OCCUR
         XC    RETCODE,RETCODE    YES. THIS IS NOT AN ERROR
         LR    RØ,R9         THE RECORD JUST READ IS A COPY OF A BLOCK
*                            WHICH WAS NOT COMPACTED BEFORE. PREPARE
*                            FOR MVCL
         L     R14,=A(IBUFFER)
         LA    R14,1(R14)    RØ AND R14 ARE TARGET AND SOURCE ADDRESSES
*                            FOR MVCL RESPECTIVELY
         L     R1,BLKSIZE    NUMBER OF BYTES TO BE MOVED TO TARGET
         LR    R15,R1        SAME FOR SOURCE
         MVCL  RØ,R14        MOVE THE UNCOMPACTED DATA
```

```
DCOMPQOK EQU   *
         ST    R9,MBPBUFAD   STORE BUFFER ADDRESS INTO MBUFLST
         ST    R6,MBPBKNUM   NUMBER OF THE BLOCK TO BE PROCESSED
         A     R9,BLKSIZE    (R9)=ADDRESS OF THE NEXT BUFFER
         A     R6,=F'1'      (R6)=NUMBER OF THE NEXT BLOCK
         LA    R11,MBUFLEN(R11)    (R11)=ADDRESS OF THE NEXT ENTRY IN
*                                   MBUFLST
         BCT   R1Ø,FILLQUOT   FILL ALL ENTRIES IN MBUFLST
         IUCV  SEND,PRMLIST=PARMLIST,PATHID=PATHID,DATA=PRMMSG,PRTY=NO,*
               TYPE=2WAY
         BZ    SENDWAIT      NORMAL COMPLETION OF SEND REQUEST
NONZERO  MVC   ERRORTXT,=CL8Ø'NONZERO VALUE IN IPRCODE ON SEND REQUEST'
         BAL   R3,WRTERM
         XR    R15,R15       NON-ZERO VALUE STORED IN IPRCODE
         IC    R15,IPRCODE   PUT SEND ERROR RETURN CODE IN R15
         ST    R15,RETCODE
         B     RTRVBFR       TERMINATE USE OF IUCV
SENDWAIT MVC   NEXTADDR,=AL3(SENDCMPL)
         LPSW  WAITPSW       ENABLE EXTERNAL INTERRUPTS AND WAIT FOR
*                            IUCV MESSAGE COMPLETE INTERRUPTION
SENDCMPL LA    R7,INTPLIST   NOW R7 IS IUCV INTERRUPT BUFFER BASE REG
         CLI   IPTYPE,X'Ø7'  IUCV MESSAGE COMPLETE INTERRUPTION?
         BE    MESSCMPL      YES
         CLI   IPTYPE,X'Ø4'  PATH QUIESCED BY *BLOCKIO?
         BE    QUIESCED      YES
SEVERED  MVC   ERRORTXT,=CL8Ø'PATH SEVERED BY *BLOCKIO'    NO. SEVERED
         BAL   R3,WRTERM
         XR    R15,R15
         IC    R15,IPUSER    PUT *BLOCKIO ERROR CODE IN R15
         ST    R15,RETCODE
         B     RTRVBFR
QUIESCED MVC   ERRORTXT,=CL8Ø'PATH QUIESCED BY *BLOCKIO'
         BAL   R3,WRTERM
         XR    R15,R15
         IC    R15,IPUSER    PUT *BLOCKIO ERROR CODE IN R15
         ST    R15,RETCODE
         B     RTRVBFR
MESSCMPL XR    R15,15
         ICM   R15,15,IPRMMSG1    ANY ERRORS BEFORE I/O INITIATION?
         ST    R15,RETCODE
         BNZ   MESSERR       YES. ISSUE ERROR MESSAGE AND EXIT
         LA    R11,MBUFLST
         LA    R1Ø,NUMBUF
TSTQLOOP XR    R15,R15
         ICM   R15,1,MBPSTAT ANY ERRORS DETECTED AFTER I/O INITIATION?
         ST    R15,RETCODE
         BNZ   MULTERR       YES. ISSUE ERROR MESSAGE AND EXIT
         LA    R11,MBUFLEN(R11)
         BCT   R1Ø,TSTQLOOP
         BCT   R5,QUOTLOOP
```

```
         B     RESTTEST
MESSERR  MVC   ERRORTXT,=CL8Ø'*BLOCKIO ERROR BEFORE ANY I/O INITIATION'
         BAL   R3,WRTERM
         B     CLOSE
MULTERR  MVC   ERRORTXT,=CL8Ø'ERROR READING ONE OR MORE DISK BLOCKS'
         BAL   R3,WRTERM
         B     CLOSE
DCOMPERR MVC   ERRORTXT,=CL8Ø'ERROR IN DECOMPACTION ROUTINE'
         BAL   R3,WRTERM
         B     CLOSE
BLKERR   MVC   ERRORTXT,=CL8Ø'DISK BLOCK SIZES DO NOT MATCH'
         BAL   R3,WRTERM
         LA    R15,1Ø16
         ST    R15,RETCODE
         B     CLOSE
SIZEERR  MVC   ERRORTXT,=CL8Ø'DISK SIZES DO NOT MATCH'
         BAL   R3,WRTERM
         LA    R15,1Ø2Ø
         ST    R15,RETCODE
         B     CLOSE
RESTTEST LTR   R4,R4          IS THE REMAINDER ZERO?
         BZ    CLOSE          YES
         LA    R7,PARMLIST    NOW R7 IS IUCV PARAMETER LIST BASE REG
         XC    PARMLIST,PARMLIST  CLEAR IUCV PARAMETER LIST
         LA    RØ,3
         ST    RØ,IPTRGCLS    MULTIPLE REQUEST TO *BLOCKIO
         LR    R1Ø,R4         (R1Ø)=NUMBER OF DISK BLOCKS REMAINED
         ST    R1Ø,IPRMMSG1   STORE IT IN PARMLIST
         LA    R11,MBUFLST    ADDRESS OF PLIST DEFINING THE BLOCKS TO BE
*                             PROCESSED AND THE BUFFERS TO BE USED
         ST    R11,IPRMMSG2   STORE IT IN PARMLIST
         L     R9,=A(DATABUFS)
FILLREST L     R2,=A(INPBUFF)     INPUT BUFFER ADDRESS USED BY GET
         GET   INPDSK,(R2)    READ A RECORD INTO THE BUFFER
         ST    R9,OUTBUFAD    ADDRESS TO WHICH DMKDDC IS TO MOVE THE
*                             THE DECOMPACTED STRING
         LA    R1,DPLIST      ADDRESS OF DECOMPACTION ROUTINE'S PLIST
         L     R15,DMKDDCAD   DECOMPACTION ROUTINE'S ENTRY ADDRESS
         BALR  R14,R15        CALL THE DECOMPACTION ROUTINE
         ICM   R15,15,RETCODE     DECOMPACTION ROUTINE'S RETCODE IN R15
         BZ    DCOMPROK       DECOMPACTION OK
         C     R15,=F'2'      DECOMPACTION RETURN CODE=2?
         BNE   DCOMPERR       NO. DECOMPACTION ERROR. SHOULD NOT OCCUR
         XC    RETCODE,RETCODE    YES. THIS IS NOT AN ERROR
         LR    RØ,R9          THE RECORD JUST READ IS A COPY OF A BLOCK
*                             WHICH WAS NOT COMPACTED BEFORE. PREPARE
*                             FOR MVCL
         L     R14,=A(IBUFFER)
         LA    R14,1(R14)     RØ AND R14 ARE TARGET AND SOURCE ADDRESSES
*                             FOR MVCL RESPECTEVELY
```

```
         L     R1,BLKSIZE    NUMBER OF BYTES TO BE MOVED TO TARGET
         LR    R15,R1        SAME FROM SOURCE
         MVCL  RØ,R14        MOVE THE UNCOMPACTED DATA
DCOMPROK EQU   *
         ST    R9,MBPBUFAD   STORE THE BUFFER ADDRESS IN MBUFLST
         ST    R6,MBPBKNUM   NUMBER OF THE BLOCK TO BE PROCESSED
         A     R9,BLKSIZE    (R9)=ADDRESS OF THE NEXT BUFFER
         A     R6,=F'1'      (R6)=NUMBER OF THE NEXT BLOCK
         LA    R11,MBUFLEN(R11)    (R11)=ADDRESS OF THE NEXT ENTRY IN
*                                   MBUFLST
         BCT   R1Ø,FILLREST   FILL FIRST (R4) ENTRIES IN MBUFLST
         IUCV  SEND,PRMLIST=PARMLIST,PATHID=PATHID,DATA=PRMMSG,PRTY=NO,*
               TYPE=2WAY
         BNZ   NONZERO       NONZERO VALUE STORED IN IPRCODE
         MVC   NEXTADDR,=AL3(CMPLSEND)
         LPSW  WAITPSW       ENABLE EXTERNAL INTERRUPTS AND WAIT FOR
*                            IUCV MESSAGE COMPLETE INTERRUPTION
CMPLSEND LA    R7,INTPLIST   NOW R7 IS IUCV INTERRUPT BUFFER BASE REG
         CLI   IPTYPE,X'Ø7'  IUCV MESSAGE COMPLETE INTERRUPTION?
         BE    CMPLMESS      YES
         CLI   IPTYPE,X'Ø4'  PATH QUIESCED BY *BLOCKIO?
         BE    QUIESCED      YES
         B     SEVERED       NO. SEVERED
CMPLMESS XR    R15,15
         ICM   R15,15,IPRMMSG1    ANY ERRORS BEFORE I/O INITIATION?
         ST    R15,RETCODE
         BNZ   MESSERR       YES. ISSUE ERROR MESSAGE AND EXIT
         LA    R11,MBUFLST
         LR    R1Ø,R4
TSTRLOOP XR    R15,R15
         ICM   R15,1,MBPSTAT ANY ERRORS DETECTED AFTER I/O INITIATION?
         ST    R15,RETCODE
         BNZ   MULTERR       YES. ISSUE ERROR MESSAGE AND EXIT
         LA    R11,MBUFLEN(R11)
         BCT   R1Ø,TSTRLOOP
CLOSE    CLOSE INPDSK        CLOSE THE DUMP FILE
RTRVBFR  IUCV  RTRVBFR       TERMINATE ALL USE OF IUCV
EXIT     HNDEXT CLR          DEACTIVATE IUCV EXTERNAL INTERRUPT HANDLER
         LCTL  CØ,CØ,SAVCTRLØ     RESTORE CONTROL REGISTER Ø CONTENTS
         STNSM WORK,X'ØØ'
         XR    R15,R15
         IC    R15,SYSMASK
         EX    R15,STOSM           RESTORE ORIGINAL SYSTEM MASK
         L     R15,RETCODE
         L     R13,SAVE+4
         L     R14,12(R13)
         LM    RØ,R12,2Ø(R13)
         BR    R14
         SPACE
IUCVEXIT STM   R14,R12,12(R13)    GENERAL EXTERNAL INTERRUPT EXIT
```

```
         LR    R15,R12
         USING IUCVEXIT,R15
         USING EXTUAREA,R1
         CLC   EXTUCODE,=X'4000'  IUCV EXTERNAL INTERRUPT?
         BNE   EXTRET             NO. IGNORE IT AND WAIT
         NI    EXTUPSW+1,255-X'02'     TURN WAIT BIT IN PSW OFF
         NI    EXTUPSW,255-X'01'  DISABLE EXTERNAL INTERRUPTS
EXTRET   LM    R14,R12,12(R13)    RESTORE REGISTERS
         BR    R14                EXIT
         DROP  R15
         DROP  R1
         SPACE
WRTERM   LINEWRT DATA=(ERRORTXT,L'ERRORTXT)
         BR    R3
         SPACE
STOSM    STOSM WORK,X'00'
         SPACE
SAVE     DS    9D
PARMLIST DS    CL48     IUCV PARAMETER LIST   (DOUBLE WORD BOUNDARY!)
INTPLIST DS    CL48     IUCV INTERRUPT BUFFER (DOUBLE WORD BOUNDARY!)
WAITPSW  DS    D
         ORG   WAITPSW
         DC    X'FF06000000'
NEXTADDR DS    AL3
&I       SETA  1
MBUFLST  DS    0D
.REPEAT  ANOP
         DC    X'01'
         DS    X
         DC    H'0'
         DS    A
         DS    F
         DC    F'0'
&I       SETA  &I+1
         AIF   (&I LE &E).REPEAT
RETCODE  DS    F
SAVCTRL0 DS    F
CTRL0    DS    F
BLKSIZE  DS    F
OFFSET   DC    F'0'
STRBLOCK DS    F
ENDBLOCK DS    F
VDEVADDR DS    H
PATHID   DS    H
WORK     DS    X
SYSMASK  DS    X
ERRORTXT DS    CL80
*
OUTSTRLN DS    F
DMKDDCAD DC    V(DMKDDC)
```

```
DMKDDTAD DC    V(DMKDDT)
*
*              DMKDDC (DECOMPACTION ROUTINE) PARAMETER LIST FORMAT
*
*              R1 -> A(A(DECODING TABLES))
*                    A(INPUT STRING)
*                    A(LENGTH OF OUTPUT STRING) - DECOMPACTED STRING
*                                                 LENGTH RETURNED BY
*                                                 DMKDDC
*                    A(OUTPUT BUFFER)
*                    A(LENGTH OF OUTPUT BUFFER)
*                    A(RETURN CODE)
*
DPLIST   DC    A(DMKDDTAD)
         DC    A(IBUFFER)
         DC    A(OUTSTRLN)
OUTBUFAD DS    A
         DC    A(BLKSIZE)
         DC    A(RETCODE)
         SPACE
INPDSK   DCB   DDNAME=INPDD,DSORG=PS,MACRF=(GM)
         LTORG
         ORG   VREST+4Ø96
&I       SETA  1
DATABUFS EQU   *
.LOOP    ANOP
         DS    CL4Ø96
&I       SETA  &I+1
         AIF   (&I LE &E).LOOP
         SPACE
INPBUFF  DS    CL41Ø1
         ORG   INPBUFF
RDW      DS    H
         DS    H'Ø'
IBUFFER  DS    CL4Ø97
         SPACE
         NUCON
         DEVSECT
         ADT
         COPY  IPARML
         EXTUAREA
         SPACE
MULTBUFF DSECT ,             MUST BE ON A DOUBLEWORD BOUNDARY
MBPRCOD  DS    X'Ø1'         REQUEST CODE: X'Ø2' - READ, X'Ø1' - WRITE
MBPSTAT  DS    X             BLOCK STATUS RETURNED BY *BLOCKIO
         DS    H'Ø'          NOT USED. MUST BE ZERO
MBPBUFAD DS    A             DATA BLOCK ADDRESS
MBPBKNUM DS    F             NUMBER OF THE BLOCK TO BE READ OR WRITTEN
         DS    F'Ø'          NOT USED. MUST BE ZERO
MBUFLEN  EQU   *-MULTBUFF
```

```
         SPACE
         REGEQU
         END   VREST
```

VDUMP EXEC

```
/* VDUMP EXEC */
     TRACE;
     SAY "ENTER THE ADDRESS OF THE DISK TO BE DUMPED (INPUT DISK)";
     SAY "OR PRESS <ENTER> TO EXIT:";
     PARSE UPPER PULL IDSK . ;
     IF IDSK = "" THEN EXIT;
  SAY "ENTER THE ADDRESS OF THE DISK TO RECEIVE THE DUMP (OUTPUT DISK)";
     SAY "OR PRESS <ENTER> TO EXIT:";
     PARSE UPPER PULL ODSK . ;
     IF ODSK = "" THEN EXIT;
     IDSK = RIGHT(IDSK,4,'Ø');
     ODSK = RIGHT(ODSK,4,'Ø');
     IF IDSK = ODSK THEN DO;
                         SAY "INPUT AND OUTPUT DISKS MUST BE DIFFERENT";
                         EXIT;
                         END;
     ALPHABET = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
     DO J = 1 TO 26 BY 1;
      MODE.J = SUBSTR(ALPHABET,J,1);
     END;
     BUFFI = RIGHT(' ', 153,' ');
     BUFFO = BUFFI;
     ACCI = Ø;
     ACCO = Ø;
     DO J = 1 TO 26 BY 1 WHILE IDSK ¬= LEFT(BUFFI,4);
      BUFFI = RIGHT(' ', 153,' ');
      QI.J = CSL('DMSQFMOD RTNC.J RSNC.J MODE.J BUFFI FLAGI');
     END;
     IF IDSK = LEFT(BUFFI,4) THEN DO;
                                  ACCI = 1;
                                  J = J - 1;
                                  IDSKMODE = MODE.J;
                                 END;
     DO J = 1 TO 26 BY 1 WHILE ODSK ¬= LEFT(BUFFO,4);
      BUFFO = RIGHT(' ', 153,' ');
      QI.J = CSL('DMSQFMOD RTNC.J RSNC.J MODE.J BUFFO FLAGO');
     END;
     IF ODSK = LEFT(BUFFO,4) THEN DO;
                                  ACCO = 1;
                                  J = J - 1;
                                  ODSKMODE = MODE.J;
                                 END;
     MODEV = 'V';
     MODEW = 'W';
```

```
     BUFFV = RIGHT(' ', 153,' ');
     BUFFW = RIGHT(' ', 153,' ');
     QV = CSL('DMSQFMOD RTNCV RSNCV MODEV BUFFV FLAGV');
     QW = CSL('DMSQFMOD RTNCW RSNCW MODEW BUFFW FLAGW');
    'ACCESS '||IDSK||' V';
     IF RC ¬= Ø THEN EXIT RC;
    'ACCESS '||ODSK||' W';
     IF RC ¬= Ø THEN EXIT RC;
     SAY "ENTER THE FILENAME OF THE DUMP FILE";
     SAY "OR PRESS <ENTER> TO EXIT:";
     PARSE UPPER PULL FNAME . ;
     IF FNAME = "" THEN EXIT;
     SAY "ENTER THE FILETYPE OF THE DUMP FILE";
     SAY "OR PRESS <ENTER> TO EXIT:";
     PARSE UPPER PULL FTYPE . ;
     IF FTYPE = "" THEN EXIT;
    'FILEDEF OUTDD DISK '||FNAME||' '||FTYPE||' '||'W4';
     IF RC ¬= Ø THEN EXIT RC;
    'VDUMP V';
     RCDUMP = RC;
    'SET CMSTYPE HT';
    'FILEDEF OUTDD CLEAR';
    'RELEASE V';
    'RELEASE W';
     IF RSNCV = Ø THEN 'ACCESS '||BUFFV||' V';
     IF RSNCW = Ø THEN 'ACCESS '||BUFFW||' W';
     IF ACCI = 1 THEN 'ACCESS '||IDSK||' '||IDSKMODE;
     IF ACCO = 1 THEN 'ACCESS '||ODSK||' '||ODSKMODE;
     EXIT RCDUMP;
```

VREST EXEC

```
/* VREST EXEC */
     TRACE;
   SAY "ENTER THE ADDRESS OF THE DISK CONTAINING THE DUMP (INPUT DISK)";
     SAY "OR PRESS <ENTER> TO EXIT:";
     PARSE UPPER PULL IDSK . ;
     IF IDSK = "" THEN EXIT;
     SAY "ENTER THE ADDRESS OF THE DISK TO BE RESTORED (OUTPUT DISK)";
     SAY "OR PRESS <ENTER> TO EXIT:";
     PARSE UPPER PULL ODSK . ;
     IF ODSK = "" THEN EXIT;
     IDSK = RIGHT(IDSK,4,'Ø');
     ODSK = RIGHT(ODSK,4,'Ø');
     IF IDSK = ODSK THEN DO;
                         SAY "INPUT AND OUTPUT DISKS MUST BE DIFFERENT";
                         EXIT;
                         END;
     ALPHABET = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
```

```
 DO J = 1 TO 26 BY 1;
  MODE.J = SUBSTR(ALPHABET,J,1);
 END;
 BUFFI = RIGHT(' ', 153,' ');
 BUFFO = BUFFI;
 ACCI = Ø;
 ACCO = Ø;
 DO J = 1 TO 26 BY 1 WHILE IDSK ¬= LEFT(BUFFI,4);
  BUFFI = RIGHT(' ', 153,' ');
  QI.J = CSL('DMSQFMOD RTNC.J RSNC.J MODE.J BUFFI FLAGI');
 END;
 IF IDSK = LEFT(BUFFI,4) THEN DO;
                               ACCI = 1;
                               J = J - 1;
                               IDSKMODE = MODE.J;
                              END;
 DO J = 1 TO 26 BY 1 WHILE ODSK ¬= LEFT(BUFFO,4);
  BUFFO = RIGHT(' ', 153,' ');
  QI.J = CSL('DMSQFMOD RTNC.J RSNC.J MODE.J BUFFO FLAGO');
 END;
 IF ODSK = LEFT(BUFFO,4) THEN DO;
                               ACCO = 1;
                               J = J - 1;
                               ODSKMODE = MODE.J;
                              END;
 MODEV = 'V';
 MODEW = 'W';
 BUFFV = RIGHT(' ', 153,' ');
 BUFFW = RIGHT(' ', 153,' ');
 QV = CSL('DMSQFMOD RTNCV RSNCV MODEV BUFFV FLAGV');
 QW = CSL('DMSQFMOD RTNCW RSNCW MODEW BUFFW FLAGW');
'ACCESS '||IDSK||' V';
 IF RC ¬= Ø THEN EXIT RC;
'ACCESS '||ODSK||' W';
 IF RC ¬= Ø THEN EXIT RC;
 SAY "ENTER THE FILENAME OF THE FILE CONTAINING THE DUMP";
 SAY "OR PRESS <ENTER> TO EXIT:";
 PARSE UPPER PULL FNAME . ;
 IF FNAME = "" THEN EXIT;
 SAY "ENTER THE FILETYPE OF THE FILE CONTAINING THE DUMP";
 SAY "OR PRESS <ENTER> TO EXIT:";
 PARSE UPPER PULL FTYPE . ;
 IF FTYPE = "" THEN EXIT;
'FILEDEF INPDD DISK '||FNAME||' '||FTYPE||' '||'V';
 IF RC ¬= Ø THEN EXIT RC;
'VREST W';
 RCREST = RC;
'SET CMSTYPE HT';
'FILEDEF INPDD CLEAR';
'RELEASE V';
```

```
    'ACCESS '||ODSK||' W';
    'RELEASE W';
     IF RSNCV = Ø THEN 'ACCESS '||BUFFV||' V';
     IF RSNCW = Ø THEN 'ACCESS '||BUFFW||' W';
     IF ACCI = 1 THEN 'ACCESS '||IDSK||' '||IDSKMODE;
     IF ACCO = 1 THEN 'ACCESS '||ODSK||' '||ODSKMODE;
     EXIT RCREST;
```

DUMPV EXEC

```
/* LOAD THE VDUMP EXEC INTO VIRTUAL STORAGE AND CALL IT */
    'EXECLOAD VDUMP EXEC *';
    'EXEC VDUMP';
     RETCODE = RC;
    'EXECDROP VDUMP EXEC';
     EXIT RETCODE;
```

RESTV EXEC

```
/* LOAD THE VREST EXEC INTO VIRTUAL STORAGE AND CALL IT */
    'EXECLOAD VREST EXEC *';
    'EXEC VREST';
     RETCODE = RC;
    'EXECDROP VREST EXEC';
     EXIT RETCODE;
```

Vladimir Ivanov Valov
Chief Expert in System Programming
Information Centre of Ministry of Finance (Bulgaria)

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*?

Printing transparencies for your presentations

GENERAL DESCRIPTION

Have you ever wanted to print transparencies for your presentations without using PC programs, instead going from plain CMS via VSE or MVS to a CICS printer? Some years ago, I was faced with this situation and built this REXX procedure. It uses CMS as the editor of the pages, and VSE as the medium to print to a CICS-attached Kyocera laser printer. The printer can be connected via a PC with 3270 emulation or an ASCII adapter on an ES/9000. It has PRESCRIBE as a prerequisite, which is a printer control language for Kyocera printers.

I decided to use PRESCRIBE, because it uses controls that are plain text and doesn't need escape sequences, which can lead to problems when trying to transport them from a host system to an ASCII printer.

The following tasks are performed:

- A text file in CMS is created with the layout of your transparency. This file contains some control tags that are interpreted by the procedure.
- You invoke PRFOLIE, which creates an intermediate file with printer controls and your text. This file is submitted, wrapped in a VSE job that 'prints' it to the VSE list queue.
- From the list queue, the file can be sent to the printer by any utility that can print from VSE batch to a CICS printer (for example the report controller feature from IBM or RAPS from Computer Associates). My sample job, which is included in this article, uses RAPS.

The transparency is built up using a template that specifies a rough layout of the page. Within this page there can be the following types of text:

- A header line (in bold face).

- The topic of your presentation (in bold face, centered).
- Alternatively, you can use four tab stops together with left-aligned topics (in bold face).
- Plain text between the topics (in standard font).
- A bottom line (with date, foil name, etc).

To submit the job to VSE, using variable parameters, some utility procedures were developed, which could also be used in other contexts:

- SUB EXEC submits a job to VSE.
- EXXX EXEC uses skeleton jobs with variable parts and produces a job stream that is finally submitted by calling SUB EXEC. Variables '*xxxxxx*' within the job can be replaced by real values.
- INCLUDE XEDIT is used to implement the feature of 'including' files within the submitted job (this is similar to the /INCLUDE command in VSE ICCF).

All procedures use the construct of global variables that are set by a procedure SETGLOBL EXEC, published in *Back-up of CMS mini-disks and SFS directories, VM Update,* Issue 99, November 1994. SUB, EXXX, and INCLUDE were published in *Saving all relevant VM/VSE data – part 2,VSE Update*, Issue 26, June 1997.

The transparency text file must have the following format and contents:

- First line – header text with a colon in column one, and a maximum of 28 characters.
- Second line – n, where n is the foil number; or n TABS t1 t2 t3 t4, where t1, t2, t3, t4 are tab positions in centimetres from left border (with default values of 3, 6, 9, and 12).
- Next lines – your presentation topics.
- These lines can be interrupted by 'includes' of normal text by specifying:

```
:INC
...... any text ......
...... any text ......
...... any text ......
:INC
```

You can code a maximum of 14 topics (included-text is not counted). All topics are printed in bold style, the included text in normal style. If you use the above mentioned tab positions, you code the tabulator as a circumflex accent (^).

SYNTAX PRFOLIE EXEC

The procedure is called with a series of parameters:

```
PRFOLIE <vse> fn <ft <fm>> <(skel>
```

or

```
PRFOLIE <vse> FILE <(skel>
```

where:

- vse – the VSE machine where the batch job is to be submitted.
- fn – the filename with the contents of the transparency.
- ft – the filetype, with a default is FOLIE.
- fm – the filemode, with a default of A.

  Note that fn, ft, and fm may also be generic, for example TEST*.
- skel – the filetype of the skeleton, with a default of BEISP$$$. The filename is always PRFOLIE.
- FILE – the filenames of all transparency files are read from a CMS file for which the user is prompted. In this case the transparencies become sequentially numbered.

INTERNALS AND PREREQUISITES

Temporary files ‘fn $PRFOLIE$’ on the A mini-disk are used. An XEDIT macro PRFOLIE1 is used for composing the print file. A job PRFOLIE JOB is used as a skeleton for the submitted VSE job.

Several assumptions are made on average character widths and characters per line, using specific fonts.

A dynamic PRESCRIBE font 1001 is used for the bold style. This is generated by the PRESCRIBE GENF command which is coded in the skeleton file. This is done because the standard Kyocera fonts are not large enough or bold enough for transparencies, which should be readable from a distance.

The VSE/ICCF utility program DTSRELST is used to print the transparency text, which is submitted in the job as SYSIPT data, to the list queue. The RAPS utility CAPRUTL0 is used to separate the print output, with the job control output (which is 'discarded' to class X), and the 'real' text output for the CICS printer. This is necessary, because DTSRELST prints to SYSLST, as does the job control.

INSTALLATION–SPECIFIC CONFIGURATION

To use this procedure at different installations, with different environments, there are several variables, which are described below.

Permanent GLOBALV variables

Permanent GLOBALV variables are set by procedure SETGLOBL EXEC, which creates a file INITIAL GLOBALV that should be copied to the A disk of all users who want to work with PRFOLIE EXEC.

Some of the variables of SETGLOBL, which are used by PRFOLIE and its child procedures are:

- $vse1 – first VSE system
- $vse2 – second VSE system
- $vse3 – third VSE system
- $vsedef – default VSE system
- $sysid1 – sysid of first VSE system
- $sysid2 – sysid of second VSE system

- $sysid3 – sysid of third VSE system.

Hardcoded values

The only hardcoded value is ‘class’. This allows you to specify the printer class which is intercepted by RAPS for printing to a CICS printer (it becomes the POWER list queue class). In the following procedure, class is set to ‘D’.

LIMITATIONS

The procedure is able to produce transparencies with more than one page, automatically repeating the header and footer on subsequent pages. This, however, requires a special program for printing in VSE from SYSIPT to the list queue, which accepts a ‘.pa’ command creating a page eject. This program is beyond the scope of this article and therefore I used DTSRELST, instead. If you have a program that accepts some form of page eject command while printing from ‘card’ to ‘list’, you can use it instead of DTSRELST and change the ‘.pa’ in PRFOLIE1 XEDIT accordingly.

PRFOLIE EXEC

```
/**********************************************************************/
/* Printing transparencies                                            */
/**********************************************************************/
/* Call:   PRFOLIE <vse> fn <ft <fm>> <(skel>                         */
/*         PRFOLIE <vse> FILE         <(skel>                         */
/*                                                                    */
/* parameters: see description above                                  */
/**********************************************************************/
trace off
'GLOBALV SELECT $$GLOB$$ GET $vse1    $vse2    $vse3    $vsedef',
                           '$sysid1  $sysid2  $sysid3         '
parse upper arg vse fn ft fm . '(' beisp test
if vse = '?' then signal hilfe

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 vse = $vsedef
   parse upper arg fn ft fm . '(' beisp .
   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 ¬= $vse1 & vse ¬= $vse2 & vse ¬= $vse3 then vse = $vsedef
   parse upper arg fn ft fm . '(' beisp .
   end

if ft = '' then ft = 'FOLIE'
if fm = '' then fm = 'A'
if vse = $vse1 then sysid = $sysid1
   else if vse = $vse2 then sysid = $sysid2
        else sysid = $sysid3
class = 'D'                          /* <======== printer class */
if beisp = '' then beisp = 'BEISP$$$'

if verify('*%',fn,'M') ¬= Ø | verify('*%',ft,'M') ¬= Ø | ,
   index(ft,'*') ¬= Ø then name = 'PRFOLI'
                      else name = fn


'SET CMSTYPE HT'
'ERASE * $PRFOLI$ A'
'SET CMSTYPE RT'

push fn ft fm beisp
'XEDIT' name '$PRFOLI$ A (PROFILE PRFOLIE1'


/* A job PRFOLIE JOB must exist containing a skeleton of the VSE job */

'EXEC EXXX' vse name '$PRFOLI$ A PRFOLIE' ,
                     'CLASS'   class   'DEST'    'ANY'

exit
hilfe:
'VMFCLEAR'
address cms 'type prfolie exec * 1 8'
PRFOLIE1 XEDIT
/*********************************************************************/
/* Subroutine for PRFOLIE EXEC  (XEDIT macro)                        */
/* Takes the transparency skeleton an builds up the actual contents  */
/* using:   character width: Ø.25 inch (4 cpi) with capitals         */
/*          character width: Ø,2Ø inch (5 cpi) on average            */
/*          space width:     Ø,14 inch (7 cpi)                       */
```

```
/*          characters per line: approx. 25    with capitals          */
/*          characters per line: approx. 32    on average             */
/*********************************************************************/
/* Input from PRFOLIE EXEC via stack:                                */
/* fn ft fm beisp                                                    */
/*********************************************************************/

trace off

'SET MSGMODE OFF'
'SET WRAP OFF'
'EXTRACT /LINEND/'
'SET LINEND OFF'

parse upper pull fn ft fm beisp

?file = Ø
if fn = 'FILE' then do
   ?file = 1
   say 'Please specify fn ft fm'
   pull xfn xft xfm
   'EXECIO * DISKR' xfn xft xfm '(FINIS'
   end
else 'LISTFILE' fn ft fm '(STACK'

foliennr = 1
zeilenmax = 14
zeile. = ''

anzahl = queued()
do i = 1 to anzahl
   if i ¬= 1 then 'INPUT .pa;'
   pull fn ft fm
   'GET PRFOLIE' beisp '*'
   'TOP'
   'EXECIO 1 DISKR' fn ft fm '1 (VAR UEB'
   'EXECIO 1 DISKR' fn ft fm '2 (VAR XXX'
   'EXECIO * DISKR' fn ft fm '3 (FINIS STEM ZEILE.'
   ueb = strip(substr(ueb,2))
   maxchar = 25
   indent = trunc((maxchar - length(ueb)) % 2 * 7 / 4)
   if indent < Ø then indent = Ø
   ueb = copies(' ',indent) || strip(ueb)
   parse var xxx ':' nr 'TABS' tabpos1 tabpos2 tabpos3 tabpos4 .
   nr  = strip(nr)
   if ?file then nr = foliennr
   if tabpos1 = '' then tabpos1 = 3
   if tabpos2 = '' then tabpos2 = 6
   if tabpos3 = '' then tabpos3 = 9
   if tabpos4 = '' then tabpos4 = 12
```

```
   maxchar = 32
   inc = Ø
   maxlen = Ø
   do j = 1 to zeile.Ø
                    /* If tab positions are specified then no    */
                    /* centering in performed                    */
      if pos('¬',zeile.j) > Ø then do
         maxlen = maxchar
         leave j
      end
      zeile.j = strip(zeile.j,'T')
      if pos(':INC',zeile.j) > Ø & ¬inc then inc = 1
         else if pos(':INC',zeile.j) > Ø &  inc then inc = Ø
      if inc = Ø then maxlen = max(maxlen,length(zeile.j))
   end


/* indent = trunc(((maxchar - maxlen) % 2) * 7 / 5) * Ø.14 * 2.54 */
                         /* indent in centimeters */
   if maxlen > maxchar then maxlen = maxchar
   indent = (maxchar - maxlen) / 2 / 5 * 2.54
   indent = indent + 1     /* + 1 centimeter border */

   ':Ø CH|*UEB*|'ueb'|*'
   ':Ø CH/*NR*/-' nr '-/*'
   folname = left('*'||fn||'*',1Ø)
   ':Ø CH/*NAME*/'folname'/*'
   ':Ø CH+*DT*+'date(E)'+*'
   inc = Ø
   k = Ø
   einheiten = Ø  /* for page eject within a foil */
   teilseite = 1
   ':Ø /*LINE1*/'
   'UP 1'
   do j = 1 to zeile.Ø
      if pos(':INC',zeile.j) > Ø & ¬inc then do
      /* 'INPUT |R| SLM 1.5;FONT 6; SCPI 12; EXIT;' */
      /* 'INPUT |R| SLM 1.5;FONT 9; SCPI 12; SLPI 6; EXIT;' */
         'INPUT |R| SLM 1.5;FONT 45; SCPI 12; SLPI 6; EXIT;'
         einheiten = einheiten + 4
         inc = 1
         iterate j
      end
      if pos(':INC',zeile.j) > Ø & inc then do
         'INPUT |R| SLM' indent';FONT 1ØØ1; SCPI 7; SCPI Ø; EXIT;'
         einheiten = einheiten + 1
         inc = Ø
         iterate j
      end
```

```
      if inc = 1 then do
         if einheiten > 47 then call seitenwechsel
         if left(zeile.j,4) = '.pa;' | left(zeile.j,4) = 'PA;' then do
            call seitenwechsel
            iterate j
         end
         'INPUT' left(zeile.j || ' ',8Ø)
         einheiten = einheiten + 1
         end
      else do
         k = k + 1

         tab = Ø
         parse var zeile.j links '¬' tab1 '¬' tab2 '¬' tab3 '¬' tab4
         if tab1 ¬= '' | tab2 ¬= '' | tab3 ¬= Ø | tab4 ¬= Ø then do
            tab = 1
            zeile.j = '|R| SCP;MRP 'tabpos1',Ø;EXIT;'strip(tab1,'T')
            z2      = '|R| RPP;SCP;MRP 'tabpos2',Ø;EXIT;'strip(tab2,'T')
            z3      = '|R| RPP;SCP;MRP 'tabpos3',Ø;EXIT;'strip(tab3,'T')
            z4      = '|R| RPP;SCP;MRP 'tabpos4',Ø;EXIT;'strip(tab4,'T')
            z5      = '|R| RPP;EXIT;'strip(links,'T')
         end

         ':Ø /*LINE'k'*/'
         if k = 1 then do
            'UP 1'
            'INPUT |R| SLM' indent';EXIT;'
            einheiten = einheiten + 4
            'N 1'
         end
         'CH|*ZEILE'k'*|'zeile.j'|'
         einheiten = einheiten + 4
         if tab then do
            if tab2 ¬= '' then 'INPUT' z2
            if tab3 ¬= '' then 'INPUT' z3
            if tab4 ¬= '' then 'INPUT' z4
            'INPUT' z5
         end
      end
   end
   do l = (k+1) to zeilenmax
      ':Ø /*LINE'l'*/'
      'DELETE 1'
   end
   'BOTTOM'
   foliennr = foliennr + 1
end
'BOTTOM'
'INPUT |R| RES; STM Ø.5; SLM Ø.75; FTMD 13; FONT 6; EXIT,E;'
'SET LINEND ON' LINEND.2
```

```
'FILE'
exit

/*********************************************************************/
 seitenwechsel:
/*********************************************************************/

/* bottom */
'INPUT |R| SLM' indent';FONT 1Ø01; SCPI 7; SCPI Ø; EXIT;'
'INPUT |R| SLM 1; MZP 1, 26.2; EXIT,E;'
'INPUT ____________________|R| FONT 56; EXIT;'
'INPUT  '
'INPUT             Your company  'folname'      -' nr '-' ,
       '  'date(E)' ... /'teilseite+1
'INPUT .pa;'      /* TEXTMAN new page */

teilseite = teilseite + 1

/* header */
"INPUT |R| UNIT C; GENF 1Ø01,'DYNAMIC2',1,1,254,1,Ø.8,Ø,Ø,Ø,3,Ø;EXIT;"
'INPUT |R| SLM 1; SRM 2Ø; FTMD 15; EXIT;'
'INPUT |R| SCF; FONT 1Ø01; SCPI 7; SCPI Ø; EXIT,E;'
'INPUT 'ueb
'INPUT ___________________'teilseite'_'
 if inc then 'INPUT |R| SLM 1.5;FONT 45; SCPI 12; SLPI 6; EXIT;'

einheiten = Ø

return
```

PRFOLIE JOB

```
* $$ LST CLASS=X,PRI=7,FNO=PRVS,DISP=L,SYSID=*SYSID*
// JOB PRFOLIE
// OPTION NOLOG
// UPSI 1Ø
// EXEC DTSRELST
/INCLUDE XXXXXXXX XXXXXXXX X
/*
* $$ LST CLASS=X,PRI=3,FNO=PRVS,DISP=L
/*       v————————— location of power queue and data files
// ASSGN SYSØØ1,DISK,VOL=DOSRES,SHR
// ASSGN SYSØØ2,DISK,VOL=SYSWK1,SHR
// EXEC CAPRUTLØ,SIZE=8ØK
INPUT  *JOBNAME*,CPRI=7,CFNO=PRVS,ENDISP='PRI=3',FCB=$$BFCB22
REPORT *JOBNAME*,SELECT=(1,ALL,EQ,'// JOB PRFOLIE '), -
               LST='CLASS=X,DISP=L,FNO=PRVS,SYSID=*SYSID*'
REPORT *JOBNAME*,SELECT=NOMATCH, -
               LST='CLASS=*CLASS*,DISP=L,COPY=1,SYSID=*SYSID*'
/*
```

```
/&
```

PRFOLIE BEISP$$$

```
|R| UNIT C; GENF 1ØØ1,'DYNAMIC2',1,1,254,1,Ø.8,Ø,Ø,Ø,3,Ø;EXIT;
|R| SLM 1; SRM 2Ø; FTMD 15; EXIT;
|R| SCF; FONT 1ØØ1; SCPI 7; SCPI Ø; EXIT,E;
*UEB*

_____
*ZEILE1*
*ZEILE2*
*ZEILE3*
*ZEILE4*
*ZEILE5*
*ZEILE6*
*ZEILE7*
*ZEILE8*
*ZEILE9*
*ZEILE1Ø*
*ZEILE11*
*ZEILE12*
*ZEILE13*
*ZEILE14*
|R| SLM 1; MZP 1, 26.2; EXIT,E;
_____|R| FONT 56; EXIT;

            Your company   *NAME*       *NR*   *DT*
|R| RPF; EXIT,E;
```

SAMPLE FOR A TRANSPARENCY FILE: TEST PRFOLIE

```
:My presentation
:1 TABS 2 6 9 13
^o First Topic
^^= Subtopic 1
^^= Subtopic 2
^^= Subtopic 3
^o Second Topic
:INC
This is a detailed description of my second topic. It gives you some
more information on this point. Here are the details:
             - first point
             - second point
             - third point
:INC
^o Third Topic (Table)
^100^200^300^400
^134^245^398^405
^o Fourth Topic
```

Year 2000 count-down machine and REXX

Many articles have been written discussing the year 2000 (Y2K) problem and all, without exception, have presented the time when computer clocks reach the last second of 31 December 1999 as a potentially catastrophic event. I fully agree with this. Even if thousands of analysts and programmers work hard to avoid the catastrophe, it is likely to happen anyway. The situation involves so many systems, programs, lines of code, and so on, that, even if only a very small percentage of these are forgotten, a large number of problems will arise.

I have been considering the Y2K situation for several months. Given the speed that my company is investigating the problem, the year 2000 will have arrived before any reasonable decision has been taken about what to do, and how to do it.

To focus attention on the Y2K situation, I decided to write a count-down program, showing the time remaining until 00.00am on 1 January 2000, and to put the program on a spare PC in an eye-catching position in the IT department. The program has been running since I did this and, by the time of publication, it will be showing that there are fewer than 70 million seconds remaining until the year 2000.

Second by second, the count-down program calculates the number of seconds remaining, and displays that figure in red, in the middle of the screen. There are also lines of text above and below, explaining what the figure represents. Because looking at the display soon becomes somewhat boring, I decided to upgrade it a little to make it more noticeable.

To achieve a more interesting effect, I decided to have the program issue a beep whenever the figure displayed for the time remaining in seconds did not contain the same decimal digit twice. For example 93,124,760 is a candidate, whereas 93,124,759 is not (because there are two number 9s).

After running this new version for a few days, I found that it was much more attractive, and that observers were surprised to hear a beep from

time to time, which appeared to be randomly generated, without being able to guess the cause. There are certain periods when the beep occurs frequently and other occasions when there is silence for several hours. On analysis, I found that the average at present is about 1 beep every 55 seconds, and that the average will decrease by a major step each time the count-down figure has one fewer digit. For example, the average is about one beep every 16 seconds when the remaining time is below 10 million seconds, one beep every seven seconds below one million seconds, one beep every three seconds below 100 thousand seconds, and increasing to almost one beep per second during the last one and a half minutes of the year 1999! So the count-down machine will become more and more noisy as it approaches the fatal moment! Although this increase in noise was not planned for in the upgrade, it is quite a successful outcome.

REXXING THE COUNT-DOWN MACHINE

The original count-down program was written in Turbo Pascal, not in REXX; however I have the 'Personal REXX' from Mansfield, but no graphic library. Because prototyping and debugging is so easy in REXX, I decided to use this to prototype many parts.

A simple count-down machine in REXX will look like this:

```
/* Count down machine 1 */
do forever
 say remaining() 'seconds remaining till the Year 2000'
 call pause
end
```

Here, the 'remaining' function has to calculate the time in seconds from the present time until 00.00am on 1 January in the year 2000. The 'pause' routine is not essential, but it is desirable for two reasons. Firstly, a routine can be introduced which will pause until the current time decreases by 1 second, so your program does not display the same counter several times. Secondly, if the language you use can issue a real wait, for around a second, your environment will not suffer from a CPU-grabbing program!

To develop further, the 'remaining' function can be coded as follows:

```
remaining: procedure
```

```
return (julian('20001231')  - julian(date('S')) * 86400 - seconds('S')
```

with Julian defined as follows:

```
julian: procedure
parse arg with year +4 month +2 day
if (month <= 2) then do
 year = year - 1
 month = month + 12
end
a = trunc(year / 100)
b = 2 - a + trunc(a / 4)
return trunc(365.25 * year) + trunc(30.6 * (month + 1)) + day + 1720996
+ b
```

Note how simply the Julian function has been coded, to calculate the Julian date for any given date. The formula is taken from a book of astronomy and is valid for any date represented as yyyymmdd.

Because the Julian date is the number of calendar days elapsed since a fixed date in the past, the ‘remaining’ function simply calculates the number of seconds to the year 2000. It does this by calculating the number of days still to go to the year 2000, transforms this into seconds, and subtracts from this the number of elapsed seconds since 00:00 am on the current day.

In VM, we would code the ‘pause’ routine something like:

```
pause: routine
'CP SLEEP 1 SEC'
return
```

At first sight, this may appear to be OK. However, if the required result is to have the count-down machine showing the time remaining until the year 2000, second by second, there will be a small problem here. Let’s look at the main code once more:

```
/* Count-down machine 1 */
do forever                                                  /* 1 */
 say remaining() 'seconds remaining till the Year 2000'     /* 2 */
 call pause                                                 /* 3 */
end                                                         /* 4 */
```

Assuming that statement 2 starts at time ‘t’, the next execution of statement 2 will take place at ‘t + 1 + e + d’ , where ‘1’ is the sleep time of routine ‘pause’, and ‘e’ the total execution time used by REXX to interpret statement 3, statement 4, statement 1, and statement 2

again. The time ‘d’ is the period of time when your virtual machine didn’t get CPU cycles (outside the ‘pause’ routine). That value, of course, is more or less random.

So there is a small shift, probably of a few milliseconds, for each display done at statement 2.

One could try eliminating this by coding the following main routine:

```
/* Count-down machine 1 */
last = 0
do forever
 current = remaining()
 if current = last then iterate
 say current 'seconds remaining till the Year 2000'
 last = current
end
```

Now we are virtually certain that, at every second, the new figure will be displayed. However, this would use up all free CPU resources on our system!

How you handle this depends mainly on the environment in which you are running REXX. Because I run it on a multitasking PC, I tried the Personal REXX ‘delay(duration)’ function (similar to the VM CP SLEEP command). The main code is:

```
/* Count-down machine 1 */
do forever
 say remaining() 'seconds remaining till the Year 2000'
 delay(1)
end
```

Here again, we are faced with the shift problem explained earlier. I decided to change the code again to fit with the following conditions:

- No shift problem
- As little CPU used as possible.

The main code now is:

```
/* Count-down machine 1 */
last = 0
d    = 0.9
do forever
 current = remaining()
 if current = last then iterate
```

```
 say current 'seconds remaining till the Year 2000'
 last = current
 delay(d)
end
```

This is an improvement, isn't it? The program frees 90% of CPU resources (d=0.9). The rest of the time it is calculating 'remaining', and looping if it is the same as before. Value 0.9 was empiric. I rounded it to a tenth of a second because, according to the manual, the 'delay()' function accuracy was 1/10th of a second, and I calculated that the rest of the coding took much less than 1/10th of a second to execute.

All these considerations show that an apparently simple program may, after analysis, not be simple at all.

MATHEMATICAL CONSIDERATIONS

The actual count-down machine that I have written (in Turbo Pascal) also beeps on each occasion that the decimal representation of the number of remaining seconds ('remaining' function) does not contain two digits the same. What is interesting here is that, as the count down comes closer to the year 2000, the beeps come with a higher 'density'. Above 9,999,999,999 seconds (+/- 317 years before year 2000) there

digits	No of beeps	Range	Frequency (%)
10	3,265,920	9,999,999,999 to 1,000,000,000	0.04
9	3,265,920	999,999,999 to 100,000,000	0.36
8	1,632,960	99,999,999 to 10,000,000	1.81
7	544,320	9,999,999 to 1,000,000	6.05
6	136,080	999,999 to 100,000	15.12
5	22,680	99,999 to 10,000	25.20
4	4,536	9,999 to 1,000	50.41
3	648	999 to 100	72.08
2	81	99 to 10	91.01
1	9	9 to 1	100.00

And the final [disastrous?] beep at time 0

Figure 1: Frequency of beeps

is no beep at all. In fact the first one is at 9,876,543,210 (313 years).

The average frequency of beeps depends on the number of decimal digits in the count-down number. Figure 1 shows the behaviour.

To finish, here is the function which returns 1 if the number fulfils this condition:

```
beep: procedure
arg n
d. = 0
do i = 1 to length(n)
 c = substr(n,i,1)
 if d.c then return 0
 d.c = 1
end i
return 1
```

COUNT-DOWN MACHINE FOR A PC

For those who are interested, I can send them the Turbo Pascal version of this count-down machine. Because it is written for a PC, it displays a graphic count-down engine. The syntax is as follows:

```
 Year 2000 Count Down Machine.
         WY2K   Version 1.8 - 20 October 1997

     WY2K   [/L:lang] [/S[:[times][-millisecs]]] [/C:secs]

     /L:    Use EN for English, FR for French or NE for German.
     /S     Sounds will come up according to a special secret rule |
      Use times and duration to overwrite the default music which is 2
sounds of 50 milliseconds.
     /C:    Every secs seconds, display cycles on another language.

     Animates the 'Year 2000' count-down-machine. Take a minute to read
     the notes at the bottom when it runs...
```

My e-mail address is Taymans-P@Notes.Westinghouse.com, or my private address is Taymans@ibm.net.

Philippe Taymans (Belgium)

Subscribers who want copies of the code from this issue can call our Web site – http:www.xephon.com – and ask for the article they require. The article will then be e-mailed to them. This service is free to subscribers.

Sterling Software has announced VM:Webserver Secure for VM/ESA, said to provide mainframe-class authentication and access control, and privacy and integrity of transmitted data through the use of Secure Sockets Layers.

The product uses algorithm technology licensed from RSA Data Security, which has been re-engineered as a VM-specific implementation, rather than being simply ported. This means, says Sterling, that the implementation exploits the strengths of VM/ESA and the System/390 architecture to deliver optimal performance.

To streamline the process of obtaining digital certificates via SSL handshakes, Sterling has gone to certification specialist, VeriSign, as a result of which the product is expected to provide facilities that make it easy to obtain a certificate from VeriSign, and also enables customers to obtain certificates from other standards-based certifying authorities.

For further information contact:
Sterling Software, 1800 Alexander Bell Drive, Reston, VA 22091, USA.
Tel: (703) 264 8000.
Sterling Software, 1 Longwalk Road, Stockley Park, Uxbridge, Middlesex, UB11 1DB, UK.
Tel: (0181) 867 8000.

* * *

Compuware has announced that Release 3.1 of its CICS Abend-AID/FX fault management tool now includes custom support for Language Environment for VM Release 1.5 and above. The product is geared towards resolving transaction and region problems. It provides programmers with on-line access to information about faults, identifying problems, capturing key fault information, listing all concurrent problems, and analysing and diagnosing captured information to pinpoint the cause of the problem.

Now claimed to be year 2000 ready, it's said to work with all relevant year 2000-ready releases of IBM software. Its programs have been modified to fully support year 2000 date compliance.

For further information contact:
Compuware, 31440 Northwestern Highway, PO Box 9080, Farmington Hills, MI 48334-2564.
Tel: (800) 737 7300.
Compuware,163 Bath Road, Slough, Berks, SL1 4AA, UK.
Tel: (01753) 774000.

* * *

IBM has announced software to support a Lotus Notes Version 4.5 Windows 3.1 client of OfficeVision/VM for mail and calendar services, aiding the transition to Lotus Notes. There is support for discontinued mail, PC-host calendar synchronization, and download of distribution lists and nicknames.

IBM has also announced AIX software which enables the exchange of messages between messaging systems, including OfficeVision/VM, Notes, Domino, and MS Mail. There are access rules, directory synchronization, alternative routing, multi-level security, and traffic statistics.

For further information contact your local IBM representative.

* * *