MVS update

151

April 1999

In this issue

MVS Update

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

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

Contributions

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

Editor
Jaime Kaminski

MVS Update on-line

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

Subscriptions and back-issues

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

A REXX parsing program

To Web-enable existing mainframe applications, there is a need for a parsing program to separate input values received from HTML. IBM has produced a parsing program in C, but there is no similar parsing program available in REXX. The main problem occurs when special characters that are posted in HTML format from the Web server are received as ASCII. What is required is a routine to convert the ASCII characters to the equivalent EBCDIC characters. Figure 1 contains a list of the most frequently-used characters with their ASCII and EBCDIC representations.

Special char	~	!	#	$	%	^	&	(	)	+	{	}
ASCII	7E	21	23	24	25	5E	26	29	29	2B	7B	7D
EBCDIC	A1	5A	7B	5B	6C	B0	50	4D	5D	4E	C0	D0

Special char	|	:	“	<	>	?	‘	=	\	;	’	,	/
ASCII	7C	3A	22	3C	3E	3F	60	3D	5C	3B	27	2C	2F
EBCDIC	7F	4F	7A	4C	6E	6F	79	7E	E0	5E	7D	6B	61

Figure 1: Special characters with their ASCII/EBCDIC values

A SOLUTION

The REXX parsing routine first separates the input received from the browser into variable names with their assigned values. It then replaces all ‘+’ signs with blanks (spaces). If the data was posted from a browser like MS Internet Explorer or Netscape Navigator, all blanks would have been replaced with the ‘+’ special character. All special characters (including ‘+’) would have been replaced by their corresponding ASCII values prefixed by a ‘%’ character. The parsing program converts back all ‘+’ characters to spaces and all ASCII values to their corresponding characters. The problem is that there is no standard REXX function to convert from ASCII to EBCDIC. For this purpose the REXX X2C function is used.

Note: special characters like square brackets and caret (^) are not supported in EBCDIC. Special consideration has been given to these characters in the parsing program. The sample HTML page shown below contains three input fields, and, when this HTML form is submitted, it invokes the REXX parsing program. The REXX routine receives the data, does the parsing, and then displays the result back to the browser.

OPERATIONAL ENVIRONMENT

Use of this program is dependent on the correct customization of Open Edition MVS and the Web server on the mainframe. This REXX routine must be copied into the cgi-bin directory from where you can execute your CGI (Common Gateway Interface) programs. The HTML files should be copied into the directory as specified in the pass rule of your Web server configuration file.

THE REXX PARSING PROGRAM

```
  /* REXX */
 'cgiutils -status 200 -ct text/x-ssi-html'
 say '<html '
 say '<body '
 address mvs 'EXECIO 1 DISKR STDIN (STEM infile.'  /* Post method  */
 stdin = infile.1
 /*****************************************************************/
 /*   Parsing the stream of input string into stem variables      */
 /*****************************************************************/
 parm.=''
 count = 1
 do while infile.1   ''
    parse var infile.1 varname '=' value '&' infile.1
    if parm.count = '' then parm.count = value
                        else parm.count = parm.count ' ' value
    count = count + 1;
 end
 say '<h2  Output From REXX - Parse Program </h2 '
 say '<p '
 say '</b  <BR '
 /****************************************************************/
 /*   Changing the plus sign to a space                          */
 /****************************************************************/
 do i = 1 to count-2 /* to Neglect Submit and Reset input value   */
 temp = ''
 if (index(parm.i,'+')   0) then
do
do while parm.i   ''
```

```
 parse var parm.i plus '+' parm.i
  temp = temp || plus || ' '
end
parm.i = temp
 end
 end
 /***********************************************************/
 /*   Changing the hex values to actual characters          */
 /***********************************************************/
 do i = 1 to count-2
 temp = ''
  do while parm.i   ''
     parse var parm.i hexchar '%' parm.i
if parm.i <  '' then
do
asciivalue = left(parm.i,2)
ebcdicvalue = ascii2ebcdic(asciivalue)
select
     when ebcdicvalue = 'SL' then xvalue = '['     /* left square bracket */
     when ebcdicvalue = 'SR' then xvalue = ']'     /*Right square bracket */
     when ebcdicvalue = 'CR' then xvalue = '^'     /* Carret..            */
     otherwise xvalue      = x2c(ebcdicvalue)
temp = temp || hexchar || xvalue
parm.i = substr(parm.i,3,length(parm.i))
end
else temp = temp || hexchar
end
  parm.i = temp
 end
 /***************************************************************/
 /*   Displays the value posted from HTML format                */
 /***************************************************************/
 do i = 1 to count-2 /* to Neglect Submit and Reset input value    */
say '<p '
say 'Value for variable ' i ' = '
say '<b '
say parm.i
 say '</b  <br '
 end
 say '</body '
 say '</html '
 exit
 /***************************************************************/
 /*   ASCII to EBCDIC conversion function                       */
 /***************************************************************/
 ascii2ebcdic: procedure
 arg ascii
 select
       when ascii = '7E' then return 'A1'
       when ascii = '21' then return '5A'
       when ascii = '23' then return '7B'
       when ascii = '24' then return '5B'
       when ascii = '25' then return '6C'
```

```
        when ascii = '26' then return '5Ø'
        when ascii = '28' then return '4D'
        when ascii = '29' then return '5D'
        when ascii = '2B' then return '4E'
        when ascii = '7B' then return 'CØ'
        when ascii = '7D' then return 'DØ'
        when ascii = '7C' then return '4F'
        when ascii = '3A' then return '7A'
        when ascii = '22' then return '7F'
        when ascii = '3C' then return '4C'
        when ascii = '3E' then return '6E'
        when ascii = '3F' then return '6F'
        when ascii = '6Ø' then return '79'
        when ascii = '3D' then return '7E'
        when ascii = '5C' then return 'EØ'
        when ascii = '3B' then return '5E'
        when ascii = '27' then return '7D'
        when ascii = '2C' then return '6B'
        when ascii = '2F' then return '61'
        when ascii = '5B' then return 'SL'    /* Square bracket Left    */
        when ascii = '5D' then return 'SR'    /* Squere bracket Right   */
        when ascii = '5E' then return 'CR'    /* Caret                  */
        otherwise return 'ØØ'
    end
```

SAMPLE HTML PAGE

```
<html
<head
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"
<title REXX Parsing Program </title
</head
<body text="#FFØØØØ"
<form action="/cgi-bin/rexxparse" method="POST"
    <p align="center" <font color="#FFØØØØ" size="4"
    Rexx Parsing Program </font </p
    <p &nbsp;</p
    <p <font color="#FFØØØØ" Name1
    <input type="text" size="2Ø" name="name" </font </p
    <p <font color="#FFØØØØ" Name 2
    <input type="text" size="2Ø" name="name2" </font </p
    <p <font color="#FFØØØØ" Name3
    <input type="text" size="2Ø" name="name3" </font </p
    <p align="center"
    <input type="submit" name="submit" value="Submit"
    <input type="reset" name="reset" value="Reset" </p
</form
</body
</html
```

Muthukumar Kannaigan
Systems Programmer (USA)

Dynamic linklist – an ISPF dialog

THE PROBLEM

In the January 1999 issue of *MVS Update*, an article described the OS/390 dynamic linklist feature. We have been using this facility at our shop ever since we installed OS/390 and have found it to be one of the more useful new features of the operating system. However, it does have a flaw in that the main command used to control the dynamic linklist – namely the SETPROG command – is fairly lengthy and can be cumbersome when performing several linklist changes at the same time. It is also awkward to keep track of what you are doing, especially when working with more than one linkset at the same time. IBM has provided Assembler-callable services and macros to allow custom programs to be written to handle the dynamic linklist, but what I needed was a straightforward TSO-based dialog to simplify handling the linklist without the need to resort to any extensive programming effort.

A SOLUTION

Th result was the following ISPF-based dialog. Instead of writing custom programs to interface to the linklist, I opted instead for the TSO CONSOLE interface. I did this for several reasons:

- I already had TSO CONSOLE authority.
- It would allow me to simply embed straight into the routine the SETPROG LNKLST commands I was already familiar with.
- REXX (my language of choice for ISPF dialogs) has built-in support for TSO CONSOLE using the GETMSG function to retrieve messages.

The main REXX routine issues the various SETPROG LNKLST commands via TSO CONSOLE and displays the results via ISPF table displays. The only exception to this is the initial display of the currently defined and/or active linksets. The SETPROG LNKLST,DISPLAY command returns only a very brief display containing the names of the currently defined linksets. I required a more detailed display with information such as the number of jobs (address spaces) using a particular linkset, and whether the linkset was the current LLA-managed one.

In order to do this I had to resort to examining the control blocks that govern dynamic linklist. IBM supplies two macros for programming the dynamic linklist. The first is CSVDYNL, which is the callable version of the SETPROG command. Using this I could have written some Assembler REXX functions to extract the information I needed, but, as I stated earlier, I was trying for a simpler solution. The second macro is CSVDLCB, which maps the DLCB (Dynamic Linklist Control Block) used internally by the CSVDYNL and SETPROG interfaces. The Assembler section of the macro does not show any useful values other than linkset name. However, if you read further on into the PL/I mapping section, you see much more detailed information on what is stored in the control block.

There are actually two different versions of the DLCB in existence. The first is pointed to from field ECVTDLCB off the ECVT (itself pointed to from the CVT). This contains a list of the currently defined linksets, and status flags that indicate whether it is an active linkset and, if so, whether it is the ‘current’ linkset (ie the most recently activated one). The bit values for these status flags are listed in the PL/I section of the CSVDLCB macro.

The second flavour of the DLCB is contained in the ASSB control block (field ASSBDLCB) of each address space. This control block contains the DLCB address of the linkset being used by this address space. I used this information to build the storage calls in the REXX routine to:

- Read the ECVTDLCB to extract the current linkset names and their status.
- Scan the ASVT chain, and examine each address space’s ASSBDLCB to check which linkset it was using (and in the process generate a job count for that linkset).
- Identify which linkset was being managed by LLA by simply checking the LLA address space’s ASSBDLCB value.

All of this information is built into an ISPF table and displayed on the first panel. The functions available to you are:

- Activate a linkset.
- Copy a linkset to a new name (you are prompted on the panel for the new name).
- Delete a linkset.
- Select the linkset, which takes you to a display of the datasets contained in the linkset, from where you can delete and/or insert new datasets.
- Display a list of the jobs (address spaces) using a linkset.
- Issue an ‘UPDATE’ against a linkset to update one or all address spaces to use that linkset (issued after an activate).
- Generate a PROGxx format member from a particular linkset. The generated member is presented in EDIT mode to allow you to copy it to SYS1.PARMLIB should you wish.

The dataset and jobs display also supports a FIND primary command to allow you to locate a particular job or dataset. The command does not require generic strings (eg JOB*), but will simply locate the first line containing the string you specify. (I have found this to be a more useful function than an ISPF member list style ‘LOCATE’ command.)

All messages issued by the generated SETPROG commands are returned as an ISPF message back to the dialog. Because the SETPROG commands self-check (eg you cannot delete an active linkset, and you cannot activate an already active linkset), I had no need to build in error checking to the REXX routine, merely to pass back the CSV* message.

Extending the routine to other dynamic routines

One final point to note is that OS/390 also uses the SETPROG command to control the dynamic APF and EXITS lists, and, with OS/390 Release 4 and higher, the dynamic LPA list. It is therefore quite feasible to either extend this routine to support these facilities, or quickly build new routines for each one based on this example dialog.

This dialog was written and tested under OS/390 1.3.0, TSO/E 2.5, and ISPF 4.4.

LINKR00

```
/*REXX
  Member: LINKRØØ
  Function: Dynamic Linklist control Dialog main routine
  Called by: TSO %LINKRØØ
  Panels: LINKPØØ, LINKPØ1, LINKPØ2, and LINKPØ3
  Skeleton: LINKSØØ
*/
call init
do while linkset()
end
exit
/**/
init:
"CONSPROF SOLDISPLAY(NO) UNSOLDISPLAY(NO)"
lcurs='';lrow=1
dcurs='';drow=1
linit = 1
dinit = 1
return
/**/
linkset:
"ISPEXEC TBCREATE LINKTØØ KEYS(SET) NAMES(STATUS LLAVAL JCNT)",
"NOWRITE REPLACE"
if rc ¬= Ø then do
  "ISPEXEC TBEND LINKTØØ"
  signal linkset
end
if lcurs = '' then lcurs = 'ZCMD'
if ¬linit then
 "ISPEXEC TBSKIP LINKTØØ NUMBER("lcrp")"
linit = Ø
call get_current_linksets
'ISPEXEC TBDISPL LINKTØØ PANEL(LINKPØØ) ',
'CURSOR('lcurs') CSRROW('lrow')' ,
'AUTOSEL(NO) POSITION(LCRP)'
retcode=rc;rc=Ø
lcurs='';lrow=1
select
  when retcode  > 4 then return Ø
  when ztdsels = Ø then lcrp = ztdtop
  otherwise
  do
    select
      when lcmd='C' then
       call copy_lnkset
      when lcmd='D' then
       call delete_lnkset
      when lcmd='G' then
       call generate_lnkset
      when lcmd='J' then
```

```
        call display_users set
       when lcmd='S' then
        do while display_lnkset();end
       when lcmd='A' then
        call activate_lnkset
       when lcmd='U' then
        call update_lnkset
       otherwise nop
     end
   end
end
"ISPEXEC TBEND LINKTØØ"
return 1
/**/
copy_lnkset:
zedlmsg = 'No response from system to SETPROG command - check',
'SYSLOG for messages'
lcurs = 'ZCMD';lrow=lcrp
"CONSOLE ACTIVATE"
"CONSOLE SYSCMD(SETPROG LNKLST,DEFINE,NAME="name",COPYFROM="set")",
"CART(LINKLIST)"
do forever
   getrc = getmsg('LNK.','SOL','LINKLIST',,'3Ø')
   if getrc ¬= Ø then leave
   if left(word(lnk.1,1),3) = 'CSV' then
    do
     zedlmsg = ''
     do num = 1 to lnk.Ø
       zedlmsg = zedlmsg lnk.num
     end
     lcurs = 'LCMD';lrow=lcrp
     leave
   end
end
"CONSOLE DEACTIVATE"
"ISPEXEC SETMSG MSG(ISRZØØ1)"
return
/**/
activate_lnkset:
zedlmsg = 'No response from system to SETPROG command - check',
'SYSLOG for messages'
lcurs = 'ZCMD';lrow=lcrp
"CONSOLE ACTIVATE"
"CONSOLE SYSCMD(SETPROG LNKLST,ACTIVATE,NAME="set")",
"CART(LINKLIST)"
do forever
   getrc = getmsg('LNK.','SOL','LINKLIST',,'3Ø')
   if getrc ¬= Ø then leave
   if left(word(lnk.1,1),3) = 'CSV' then
    do
     zedlmsg = ''
     do num = 1 to lnk.Ø
```

```
      zedlmsg = zedlmsg lnk.num
    end
    lcrp = lcrp-1;if lcrp < 1 then lcrp=1
    lcurs = 'LCMD';lrow=lcrp
    leave
  end
end
"CONSOLE DEACTIVATE"
"ISPEXEC SETMSG MSG(ISRZ001)"
return
/**/
update_lnkset:
zedlmsg = 'No response from system to SETPROG command - check',
'SYSLOG for messages'
lcurs = 'ZCMD';lrow=lcrp
"CONSOLE ACTIVATE"
"CONSOLE SYSCMD(SETPROG LNKLST,UPDATE,JOB="name")",
"CART(LINKLIST)"
do forever
  getrc = getmsg('LNK.','SOL','LINKLIST',,'30')
  if getrc ¬= 0 then leave
  if left(word(lnk.1,1),3) = 'CSV' then
   do
    zedlmsg = ''
    do num = 1 to lnk.0
      zedlmsg = zedlmsg lnk.num
    end
    lcrp = lcrp-1;if lcrp < 1 then lcrp=1
    lcurs = 'LCMD';lrow=lcrp
    leave
  end
end
"CONSOLE DEACTIVATE"
"ISPEXEC SETMSG MSG(ISRZ001)"
return
/**/
delete_lnkset:
if status = 'CURRENT' | STATUS = 'ACTIVE' then
do
 zedlmsg = 'You cannot delete an active Linklist Set!'
 "ISPEXEC SETMSG MSG(ISRZ001)"
 lcurs='LCMD';lrow=lcrp
 return
end
zedlmsg = 'No response from system to SETPROG command - check',
'SYSLOG for messages'
lcurs = 'ZCMD';lrow=lcrp
"CONSOLE ACTIVATE"
"CONSOLE SYSCMD(SETPROG LNKLST,UNDEFINE,NAME="set")",
"CART(LINKLIST)"
do forever
```

```
  getrc = getmsg('LNK.','SOL','LINKLIST',,'3Ø')
  if getrc ¬= Ø then leave
  if left(word(lnk.1,1),3) = 'CSV' then
   do
    zedlmsg = ''
    do num = 1 to lnk.Ø
      zedlmsg = zedlmsg lnk.num
    end
    lcrp = lcrp-1;if lcrp < 1 then lcrp=1
    lcurs = 'LCMD';lrow=lcrp
    leave
  end
end
"CONSOLE DEACTIVATE"
"ISPEXEC SETMSG MSG(ISRZØØ1)"
return
/**/
generate_lnkset:
call get_linklist set
"ISPEXEC FTOPEN TEMP"
"ISPEXEC FTINCL LINKSØØ"
"ISPEXEC FTCLOSE"
"ISPEXEC VGET (ZTEMPF) ASIS"
"ISPEXEC EDIT DATASET('"ztempf"')"
"ISPEXEC TBEND LINKTØ1"
return
/**/
display_lnkset:
call get_linklist set
if dcurs = '' then dcurs = 'ZCMD'
if ¬dinit then
  "ISPEXEC TBSKIP LINKTØ1 NUMBER("dcrp")"
dinit = Ø
zcmd = ''
'ISPEXEC TBDISPL LINKTØ1 PANEL(LINKPØ1) ',
'CURSOR('dcurs') CSRROW('drow')' ,
'AUTOSEL(NO) POSITION(DCRP)'
retcode=rc;rc=Ø
dcurs='';drow=1
"ISPEXEC VGET (ZVERB) ASIS"
if zverb = '' then parse var zcmd zverb zcmd .
select
  when retcode  > 4 then return Ø
  when abbrev('FIND',zverb,1) then
  do
    dcrp = find_entry(zcmd,dcrp,'LINKTØ1','dsname')
    return 1
  end
  when ztdsels = Ø then dcrp = ztdtop
  otherwise
  do
```

```
   select
     when lcmd='D' then
      call delete_lnklib
     when lcmd='I' then
      call insert_lnklib
     otherwise nop
   end
  end
end
"ISPEXEC TBEND LINKTØ1"
return 1
/**/
display_users:
zcmd = ''
"ISPEXEC TBCREATE LINKTØ2 KEYS(USR)",
"NOWRITE REPLACE"
if rc ¬= Ø then do
  "ISPEXEC TBEND LINKTØ2"
  signal display_users
end
"CONSOLE ACTIVATE"
"CONSOLE SYSCMD(D PROG,LNKLST,USERS,NAME="set") CART(LINKLIST)"
drop usr.
do until getrc ¬= Ø
getrc = getmsg('USR.','SOL','LINKLIST',,'3Ø')
if getrc ¬= Ø then iterate
if word(usr.1,1) = 'CSV481I' then
 do
  zedlmsg = usr.1
  "ISPEXEC SETMSG MSG(ISRZØØ1)"
  "CONSOLE DEACTIVATE"
  "ISPEXEC TBEND LINKTØ2"
  return
 end
if word(usr.1,1) = 'CSV471I' then
 do
  do cnt = 4 to usr.Ø
    usr = strip(usr.cnt)
    "ISPEXEC TBADD LINKTØ2"
  end
  getrc = 8
 end
end
"CONSOLE DEACTIVATE"
"ISPEXEC TBTOP LINKTØ2"
do forever
  zcmd = ''
  "ISPEXEC TBDISPL LINKTØ2 PANEL(LINKPØ2) POSITION(UCRP)"
  if rc > 4 then leave
  "ISPEXEC VGET (ZVERB) ASIS"
  if zverb = '' then parse var zcmd zverb zcmd .
  ucrp = ztdtop
  if abbrev('FIND',zverb,1) then
  do
    ucrp = find_entry(zcmd,ucrp,'LINKTØ2','USR')
  end
  "ISPEXEC TBTOP LINKTØ2"
  "ISPEXEC TBSKIP LINKTØ2 NUMBER("ucrp")"
end
"ISPEXEC TBEND LINKTØ2"
return
/**/
get_linklist:
"ISPEXEC TBCREATE LINKTØ1 KEYS(DSNAME) NAMES(APF VOLSER)",
"NOWRITE REPLACE"
if rc ¬= Ø then do
  "ISPEXEC TBEND LINKTØ1"
  signal get_linklist
end
"CONSOLE ACTIVATE"
"CONSOLE SYSCMD(D PROG,LNKLST,NAME="arg(1)") CART(LINKLIST)"
drop lnk.
do until getrc ¬= Ø
getrc = getmsg('LNK.','SOL','LINKLIST',,'3Ø')
if getrc ¬= Ø then iterate
if word(lnk.1,1) = 'CSV47ØI' then
 do
  do cnt = 4 to lnk.Ø
    parse var lnk.cnt . 1Ø apf 11 . 14 volser 2Ø . 22 dsname +45 .
    "ISPEXEC TBMOD LINKTØ1"
  end
  getrc = 8
 end
end
"CONSOLE DEACTIVATE"
"ISPEXEC TBTOP LINKTØ1"
return
/**/
find_entry:
procedure
crp = arg(2)
tabname = arg(3)
field = arg(4)
"ISPEXEC TBTOP "tabname
do until skipret ¬= Ø
  "ISPEXEC TBSKIP "tabname" POSITION(XCRP)"
  skipret = rc
  if pos(arg(1),value(field)) ¬= Ø then
  do
    crp = xcrp
    skipret = 8
  end
```

```
end
return crp
/**/
delete_lnklib:
if status = 'CURRENT' | STATUS = 'ACTIVE' then
do
 zedlmsg = 'You cannot modify an active Linklist Set!'
 "ISPEXEC SETMSG MSG(ISRZØØ1)"
 dcurs='LCMD';drow=dcrp
 return
end
zedlmsg = 'No response from system to SETPROG command - check',
'SYSLOG for messages'
dcurs = 'ZCMD';drow=dcrp
"CONSOLE ACTIVATE"
"CONSOLE SYSCMD(SETPROG LNKLST,DELETE,NAME="set",DSNAME="dsname")",
"CART(LINKLIST)"
do forever
  getrc = getmsg('LNK.','SOL','LINKLIST',,'3Ø')
  if getrc ¬= Ø then leave
  if left(word(lnk.1,1),3) = 'CSV' then
   do
    zedlmsg = ''
    do num = 1 to lnk.Ø
      zedlmsg = zedlmsg lnk.num
    end
    dcrp = dcrp-1;if dcrp < 1 then dcrp=1
    dcurs = 'LCMD';drow=dcrp
    leave
  end
end
"CONSOLE DEACTIVATE"
"ISPEXEC SETMSG MSG(ISRZØØ1)"
return
/**/
insert_lnklib:
if status = 'CURRENT' | STATUS = 'ACTIVE' then
do
 zedlmsg = 'You cannot modify an active Linklist Set!'
 "ISPEXEC SETMSG MSG(ISRZØØ1)"
 dcurs='LCMD';drow=dcrp
 return
end
zedlmsg = 'No response from system to SETPROG command - check',
'SYSLOG for messages'
dcurs = 'ZCMD';drow=dcrp
"ISPEXEC ADDPOP"
"ISPEXEC DISPLAY PANEL(TSLNKPØ3)"
panret = rc
"ISPEXEC REMPOP"
if panret ¬= Ø then
```

```
do
  zedlmsg = 'Insert cancelled'
  "ISPEXEC SETMSG MSG(ISRZØØ1)"
  return
end
cmd = "SETPROG LNKLST,ADD,NAME="set",DSNAME="ndsn",AFTER="dsname
"CONSOLE ACTIVATE"
"CONSOLE SYSCMD("CMD") CART(LINKLIST)"
do forever
  getrc = getmsg('LNK.','SOL','LINKLIST',,'3Ø')
  if getrc ¬= Ø then leave
  if left(word(lnk.1,1),3) = 'CSV' then
   do
    zedlmsg = ''
    do num = 1 to lnk.Ø
      zedlmsg = zedlmsg lnk.num
    end
    dcurs = 'LCMD';drow=dcrp
    leave
  end
end
"CONSOLE DEACTIVATE"
"ISPEXEC SETMSG MSG(ISRZØØ1)"
return
/**/
get_current_linksets:
psa = Ø
cvt = getstor(psa,1Ø,,'X')
ecvt = getstor(cvt,8C,,'X')
dlcb = getstor(ecvt,88,,'X')
do while getstor(dlcb,8,,'D') ¬= Ø
   dlcb = getstor(dlcb,8,,'X')
end
do until x2d(dlcb) = Ø
   "ISPEXEC TBVCLEAR LINKTØØ"
   set = getstor(dlcb,24,8)
   statflag = getstor(dlcb,2Ø,1,'B')
   select
   when left(statflag,1) = '1' then status = 'CURRENT'
   when substr(statflag,2,1) = '1' then status = 'ACTIVE'
   otherwise status = ''
   end
   if status ¬= '' then jcnt = get_jobs(dlcb)
   else jcnt = Ø
   "ISPEXEC TBMOD LINKTØØ"
   dlcb = getstor(dlcb,4,,'X')
end
"ISPEXEC TBTOP LINKTØØ"
return
/**/
get_jobs:
```

```
procedure expose llaval
lltad = arg(1)
jnum = Ø
cvt = getstor(1Ø,,,'X')                               /* address cvt      */
asvt = getstor(cvt,22C,,'X')
ascbidx = 2ØC                                       /* ascb address index */
numascb = getstor(asvt,2Ø4,,'D')                      /* get no. of ascbs */
do numascb
  ascbidx = getstor(ascbidx,4,,'X','A')  /*increment address index   */
  ascb = getstor(asvt,ascbidx,,'X')                   /* address ascb     */
  if left(ascb,2) = '8Ø' then iterate                 /* invalid ASCB     */
  if ascb=getstor(ascbidx,4,,'X','A') then
     iterate     /*invalid ascb?*/
  ascbdat = getstor(ascb)                             /* get ASCB header  */
  if ascbdat ¬= 'ASCB' then iterate                   /* invalid ASCB?    */
  if ascbdat = '8ØØØØØØØ'x then leave                 /* end of ASCBs?    */
  stcad = getstor(ascb,BØ,,'X')                       /* address STC name */
  if stcad¬=Ø then                                    /* valid address?   */
    stcid = getstor(stcad,,8)                         /* get stcname      */
  assb = getstor(ascb,15Ø,,'X')                       /* get assb         */
  assbdlcb = getstor(assb,EC,,'X')                    /* get dlcb address */
  if assbdlcb = lltad then do                    /* does it match linkset */
    jnum = jnum + 1               /* .. if so , incrememnt job count */
    if stcid = 'LLA' then llaval = 'LLA' /* and mark if LLA using it */
  end
end
return jnum
/**/
getstor:
call trace ('O')
notrace:
/*
  GETSTOR
  Function: Internal REXX function to return value from storage
  Usage:
          val = getstor(addr{,offset,length,type,adval})
          where:
          address - is the address (hexadecimal) to fetch from
                    or a variable containing the address
          offset - is the offset (hex) to add to address
          length - is the length of storage to retrieve
                   (default is 4 bytes)
          type   - 'C' - returns storage in its raw, character form
                   'X' - returns storage as hex
                   'B' - returns storage as binary
                   'D' - returns storage as decimal
          adval  - 'C' - returns storage contents
                   'A' - returns address of storage instead of contents

*/
```

```
add = arg(1)
off = arg(2);if off = '' then off = 0
len = arg(3);if len = '' then len = 4
typ = arg(4);if typ = '' then typ = 'C'
adv = arg(5);if adv = '' then adv = 'C'
select
  when adv = 'C' then
    val = storage(d2x(x2d(add)+X2d(off)),len)
  otherwise
    val = d2c(x2d(add)+X2d(off))
end
select
  when typ = 'X' then return C2X(val)
  when typ = 'D' then return C2D(val)
  when typ = 'B' then return X2B(C2X(val))
  otherwise nop
end
return val
exit
```

ISPF PANEL LINKP00

```
)PANEL KEYLIST(ISPSNAB,ISP)
)Attr Default(%+_)
   ! type( input) intens(high) caps(on ) just(left ) pad('''')
   ^ type(output) intens(high) caps(off) skip(on) just(asis )
   @ area(dynamic)
   } type(char) intens(low) color(blue)
   > type(char) intens(high) color(red)
   * type(char) intens(low) color(white)
   $ type(dataout) intens(low)
   { type(text) intens(high) color(yellow)
)Body  Expand(##)
%-#-#-Dynamic Linklist Control Centre -#-#-
%Command ===>_zcmd                                # #%Scroll ===>_amt +
@DYN,SHADOW
@
%Currently Active Linklist Sets
{-#-#-
{C  Set              Status   LLA  Jobs  Name (for C and U commands
only)
{-#-#-
)Model
!z+^z              ^z       ^z   ^z    _z               +
)Init
  .ZVARS = '(Lcmd set status llaval jcnt name)'
  &lcmd = ''
  &name = ''
 &DYN     = '$Line Commands   : +
 Activate,Copy,Delete,Jobs,Generate,Select,Update'
 &SHADOW  = ' }}}}}}}}}}}}}}}}**+
 >********>****>******>****>********>******>*****'
)Reinit
)Proc
                                  /* Process )BODY fields here      */
 If (&ztdsels ^= 0000)            /* If user selected some rows ... */
                                  /* ... process )MODEL fields here */
   VER (&LCMD,LIST,A,C,D,J,G,S,U)
   if (&lcmd = 'C')
     if (ver (&name,nb))
     else
       &zedlmsg = 'Enter new linkset name'
       ver (&name,nb,msg=isrz001)
       goto done
   if (&lcmd = 'U')
     if (ver (&name,nb))
     else
       &zedlmsg = 'Enter Address space name, or ''*'' for all jobs'
       ver (&name,nb,msg=isrz001)
       goto done
done:
)End
```

ISPF PANEL LINKP01

```
)PANEL KEYLIST(ISPSNAB,ISP)
)Attr Default(%+_)
   ! type( input) intens(high) caps(on ) just(left ) pad('''')
   ^ type(output) intens(high) caps(off) skip(on) just(asis )
   @ area(dynamic)
   } type(char) intens(low) color(blue)
   > type(char) intens(high) color(red)
   * type(char) intens(low) color(white)
   $ type(dataout) intens(low)
   { type(text) intens(high) color(yellow)
)Body  Expand(##)
%-#-#-Dynamic Linklist Control Centre -#-#-
%Command ===>_zcmd                                # #%Scroll ===>_amt +
@DYN,SHADOW
@
@DYN1,SHADOW1
@
%Datasets for Linkset ^set
{-#-#-
{C  Dataset                                     APF Volser
{-#-#-
)Model
!z+^z                                           ^Z+ ^Z     +
)Init
```

```
   .ZVARS = '(Lcmd dsname apf volser)'
   &lcmd = ''
  &DYN     = '$Primary Commands: +
  Find'
  &SHADOW  = ' }}}}}}}}}}}}}}}}}**+
  >***'
  &DYN1    = '$Line Commands   : +
  Delete,Insert'
  &SHADOW1 = ' }}}}}}}}}}}}}}}}}**+
  >******>*****'
 )Reinit
 )Proc
                                 /* Process )BODY fields here      */
  If (&ztdsels ^= 0000)          /* If user selected some rows ... */
    VER (&LCMD,LIST,D,I)
                                 /* ... process )MODEL fields here */
 )End
```

ISPF PANEL LINKP02

```
)PANEL KEYLIST(ISPSNAB,ISP)
)Attr Default(%+_)
   ! type( input) intens(high) caps(on ) just(left ) pad('''')
   ^ type(output) intens(high) caps(off) skip(on) just(asis )
   @ area(dynamic)
   } type(char) intens(low) color(blue)
   > type(char) intens(high) color(red)
   * type(char) intens(low) color(white)
   $ type(dataout) intens(low)
   { type(text) intens(high) color(yellow)
)Body  Expand(##)
%-#-#-Dynamic Linklist Control Centre -#-#-
%Command ===>_zcmd                                 # #%Scroll ===>_amt +
@DYN,SHADOW
@
%Jobs using Linkset ^set
{-#-#-
{   User     Asid  User     Asid  User     Asid  User     Asid
{-#-#-
)Model
+  ^usr
)Init
 &DYN     = '$Primary Commands: +
 Find'
 &SHADOW  = ' }}}}}}}}}}}}}}}}}**+
 >***'
)Reinit
)Proc
)End
```

ISPF PANEL LINKP03

```
)PANEL KEYLIST(ISPSNAB,ISP)
)Attr Default(%+_)
   ! type(input) intens(high)
)Body EXPAND(##) WINDOW(7Ø,9)
%
%CMD==>_ZCMD
+ Enter New Linklist Data Set name
+
+ Name         ==>!ndsn                                        +
+
+  # # Enter-Confirm F12-Cancel # #
)Init
  .Cursor = ndsn
  &nset = ''
  &zwinttl = 'Dynamic Linklist Control Centre'
)Reinit
)Proc
ver (&ndsn,NB,dsname)
)End
```

ISPF SKELETON LINKS00

```
)CM  Name: LINKSØØ
)CM
)CM  Function : PROGxx skeleton for Linklist dialog Generate function
)CM  Called By: LINKRØØ REXX routine
)DEFAULT  )&?!<|>
ØØØ11ØØØ
LNKLST Define Name(&set)
)DOT LINKTØ1
LNKLST Add Name(&set)
       Dsname(&dsname)
)ENDDOT
LNKLST Activate Name(&SET)
```

Graham Taylor
Senior Systems Programmer (UK)

Cursor-sensitive ISPF commands

INTRODUCTION

ISPF Version 4 has point-and-shoot fields on some panels. You can put the cursor on them and press ENTER, then some action will occur. For example, on the ISPF primary option panel you can select any of the options this way. However, that requires the field to be defined as a point-and-shoot field. Nonetheless, it is possible to create your own cursor-sensitive functions, which don’t rely on specially defined panel fields. This article shows some examples of cursor-sensitive commands for ISPF. They are:

- An edit macro that uses the cursor position within the data text.
- A command that uses the cursor position anywhere on the screen.

‘E’ COMMAND

From an EDIT or VIEW of a PDS member, enter the command ‘E’ with the cursor positioned within the text. This EXEC gets the word at the current cursor position as a member-name to be edited, from the same PDS, and invokes an EDIT. The command is normally defined under a PF key (eg PF4 is free for this).

This is useful, for example, where you have a PDS with an INDEX member, which lists the names and functions of all the members. The user edits the INDEX member, puts the cursor on the desired member-name, then presses PF4 to edit that member. Alternatively, type ‘E’ on the command line then use the mouse to double-click the member name in the text.

The new edit can be ‘over’ the existing edit, or can be swapped to the other side of a split screen. The EXEC is shown here set to edit “over”, but if you change the code to: ‘swap = "YES"’ it will swap. This swapping requires extra code with all the functionality of the EDI EXEC from *MVS Update 149*, so this EXEC could also be used instead of the EDI EXEC as a command to edit a dataset. See the comments at the start of the EXEC for more details.

E EXEC

```
/*============================== REXX ================================*/
/*  E - RECURSIVE EDIT OF A DATASET/MEMBER                            */
/*                                                                    */
/*     This can be used either two ways:                              */
/*                                                                    */
/*  1. EDIT MACRO                                                     */
/*                                                                    */
/*     EDIT another member of the same dataset (that you are editting)*/
/*       a. It takes the word at the cursor position (in the data)    */
/*            as the member name to be editted, or a membername can   */
/*            be supplied a parameter ('E membername').               */
/*       b. Hint: this is often invoked via a PF Key, set to 'E'.     */
/*                                                                    */
/*     Note: The new EDIT will swap to an alternate screen if you     */
/*           set variable: swap = 'YES' (near start of EXEC).         */
/*           To achieve this, the EXEC recursively re-invokes itself  */
/*           using the ISPSTRT program, which is normally used by     */
/*           the ISPF 'START' command.                                */
/*                                                                    */
/*  2. ISPF COMMAND                                                   */
/*                                                                    */
/*     EDIT a specified dataset via a primary ISPF command            */
/*                                                                    */
/*     This could be defined as an ISPF command in an ISPF command    */
/*     table (in the ISPTLIB concatenation) like:                     */
/*                                                                    */
/*      ED  : "SELECT CMD(%E &ZPARM) NEWAPPL(ISR)"                    */
/*        The user enters parameter(s) like the following:            */
/*          a)  ED             - basic EDIT panel                     */
/*          b)  ED dsname      - EDIT "dsname"                        */
/*          c)  ED dsname vol  - EDIT "dsname" on volume "vol"        */
/*          d)  ED dsname parm - EDIT "dsname" with EDIT parameters   */
/*                                                                    */
/*      ED1 : "SELECT CMD(%E 'dsname' PANEL('pnl') MACRO('mac')"      */
/*        to always edit the same 'dsname' using the panel and macro  */
/*                                                                    */
/*     Note: The char ! is an alternative to ', for fully qualifying  */
/*           a dataset name, to give extra flexability.               */
/*           (eg this allows: ISPF 'ED !dsname!' from Info/Man.)      */
/*                                                                    */
/*--------------------------------------------------------------------*/
/*   Version  3.1                    last updated:  Feb '99           */
/*====================================================================*/
 Address ISPEXEC "CONTROL ERRORS RETURN"
 swap = 'NO '                           /* if 'YES' then swap screens  */
                                        /* .. but only if it's a macro */
  Address ISREDIT
  "MACRO (member)"
```

```
/*--------------------------------------------------------------------*/
/*  EDIT MACRO: edit the specified member of this dataset             */
/*--------------------------------------------------------------------*/
  If rc = 0 Then Do                    /* I am an EDIT macro!         */

     "(curdsn) = DATASET"              /* get current dataset name    */
     "(curmem) = MEMBER"               /* get current member name     */
     If rc > 0 Then Do
        ZEDSMSG = 'This is not a PDS'
        ZEDLMSG = 'ERROR:' curdsn 'not a PDS',
                  '- it is impossible to EDIT a member'
        Address ISPEXEC "SETMSG MSG(ISRZ001)"
        Return                         /* EXIT and show message       */
        End

     "(lineno,csrpos) = CURSOR"        /* get line no. & csr position */
     "(line) = LINE .ZCSR"             /* get text of current line    */

     If member = '' Then Do
        /*--------------------------*/
        /* is it a valid selection? */
        /*--------------------------*/
        If csrpos = 0 Then Do
           ZEDSMSG = 'Nothing selected'
           ZEDLMSG = 'ERROR: cursor was not positioned in the data',
                     '- no member name selected'
           Address ISPEXEC "SETMSG MSG(ISRZ001)"
           Return                      /* EXIT & show message         */
           End

        If Substr(line,csrpos,1) = ' ' Then Do
           ZEDSMSG = 'Nothing selected'
           ZEDLMSG = 'ERROR: cursor was positioned on a blank',
                     '- no member name selected'
           Address ISPEXEC "SETMSG MSG(ISRZ001)"
           Return                      /* EXIT and show message      */
           End

        /*-----------------------------------------------------------*/
        /* get the member name from line text at the cursor position */
        /*-----------------------------------------------------------*/
        valid_chars = 'abcdefghijklmnopqrstuvwxyz'!!,
                      'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789¬#$*'
        member = CSRWORD(line,csrpos,valid_chars)
        End

     /*---------------------------------------*/
     /* if it's a valid member name - EDIT it */
     /*---------------------------------------*/
     If Length(member) < 9 & member <> '' Then Do
```

```
        If swap = 'YES' Then Do         /* swap screens, EDIT member   */
           "(dataid) = DATAID"
           Address ISPEXEC
           "LMQUERY DATAID("dataid") VOLUME(vol)"   /* get the volser */
           "SELECT PGM(ISPSTRT)",
                  "PARM(CMD(%E '"curdsn"("member")' "vol")"
           Return
           End
        "EDIT" member                   /* EDIT the member             */
        If zerrmsg <> 'ZERRMSG' Then    /* only set for an ERROR       */
           Address ISPEXEC,
           "SETMSG MSG("zerrmsg")"      /* show the error message      */
        End

     Else Do
        ZEDSMSG = 'Invalid member name'
        ZEDLMSG = 'ERROR: text "'member'" is not a valid member name'
        Address ISPEXEC "SETMSG MSG(ISRZØØ1)"
        End
     End

/*-------------------------------------------------------------------*/
/*  EDIT the specified dataset                                       */
/*-------------------------------------------------------------------*/
  Else Do                               /* NOT an EDIT macro!          */
     Address ISPEXEC
     "CONTROL ERRORS RETURN"
     Parse Upper Arg dsn parm           /* get invocation arguments    */
     If dsn = '' Then                   /* if no arguments ...         */
        "SELECT PGM(ISREDIT) PARM(P,ISREDMØ1)"  /* primary EDIT panel */
     Else Do
        dsn = Translate(dsn,"'","!")    /* convert characters ! to '   */
        If Length(parm) = 6 & Pos('(',parm) = Ø Then  /* volser given */
           "EDIT DATASET("dsn") VOLUME("parm")"
        Else
           "EDIT DATASET("dsn")" parm
        End
     edit_rc = rc
     If zerrmsg <> 'ZERRMSG' Then Do    /* only set for an ERROR       */
        "SETMSG MSG("zerrmsg")"         /* show the error message      */
     /* "SETMSG MSG(ISRZØØ2)"  <- this message gives same result       */
        If swap <> 'YES' Then           /* directly invoked            */
           Return edit_rc
        Say 'EDIT ERROR: rc =' edit_rc
        Say zerrlm                      /* show long-message text      */
        End
     End
  Return
```

CSRWORD EXEC

```
/**====================>> REXX FUNCTION <<===========================**
 **                                                                  **
 ** CSRWORD:  Return the word at the specified cursor position in a  **
 **           string.                                                **
 **           The user can optionally specify the valid characters   **
 **           that can be in a word, or use the default a-z,A-Z,Ø-9  **
 **                                                                  **
 **==================================================================**/
 Parse Arg string, csrpos, valid_chars
 If valid_chars = '' Then
    valid_chars = 'abcdefghijklmnopqrstuvwxyz'!!,
                'ABCDEFGHIJKLMNOPQRSTUVWXYZØ123456789'
 /*------------------------------------------------------------------*/
 /* find first invalid character AFTER the csrpos, then truncate     */
 /*------------------------------------------------------------------*/
 n = Verify(string,valid_chars,,csrpos) /* Find 1st invalid character */
 If n > Ø Then
    string = Left(string,n-1)

 /*------------------------------------------------------------------*/
 /* find first invalid character BEFORE the cursor                   */
 /*------------------------------------------------------------------*/
 n = Verify(Reverse(string),valid_chars)  /* note: string is reversed */
 If n > Ø Then
    string = Right(string,n-1)

 Return string
```

WARNING

Be careful whenever editing a concatenation of PDS libraries. The EDIT panel shows a dataset name and member name on the title line, but it always shows the FIRST library in the concatenation, even if the member came from one of the following libraries! However, if you SAVE the data it will always be written to the FIRST library in the concatenation. In that sense, the EDIT title information is correct.

BROWSE and VIEW show the correct source dataset name/member name in the same situation. However, if you change the data in a VIEW then use the CREATE command to save the data – you will then see panel ISRCRA1 (Edit/View – Create) with the current dataset field showing the first concatenated library.

My recommendation is to avoid editing concatenated libraries.

MSG COMMAND

The second example is usually used when browsing job output. The command ‘MSG’ is typed on the command line, the cursor is put on a message number and ENTER is pressed. The EXEC then invokes BookManager using the word at the cursor position as a search argument. It searches through the MESSAGES bookshelf.

Another method is to use a single-click of the mouse to position the cursor on the message number, then a PF key to enter the ‘MSG’ command, invoking the EXEC.

Note: when you are already in split screen mode, the cursor and the ‘MSG’ command must be entered in the same logical screen.

The MSG1 EXEC invokes program ISPFSCRN to get the ISPF screen buffer and cursor position. The program uses information about ISPF control blocks that were published a long time ago, and it may become invalid with some future release of ISPF. However, this program has been used with ISPF Releases 2.3 to 4.5 successfully.

As shown here, the MSG1 EXEC does a screen swap for the BookManager display. This is done because I have often found it useful to be able to switch between the exact text of the message in the output and the description of the message in BookManager. Similarly to ‘E’, you can change the code to ‘swap = "NO"’ to display BookManager over the top of the original message text.

As shown here, the MSG2 EXEC invokes the MSGLIBS EXEC to allocate and deallocate the libraries for BookManager, in case they are not permanently allocated. Customize MSG2 (and MSGLIBS) as necessary.

MSG1 EXEC

```
/*=========================>> REXX <<===============================*/
/*  EXEC-Type  : EXEC/ISPF                                          */
/*                                                                  */
/*  Desc.      : Cursor-sensitive Message selection for BookManager   */
/*                                                                  */
/*               The EXEC will invoke program ISPFSCRN, which will
*/
/*               return screen image and cursor position in REXX      */
/*               variables. Next the message is extracted, then       */
/*               BookManager is invoked with a search for the msg.    */
/*                                                                    */
/*  Implement  : Create an ISPF command like the following            */
/*                  "MSG      SELECT CMD(%MSG1 &ZPARM)"               */
/*               Define a PF key as "MSG".                            */
/*                                                                    */
/*  Invoke     : Use one of the following two methods                 */
/*               a) put cursor on the message-number then press the   */
/*                    PF key defined for this                         */
/*               b) enter "MSG msgnum" on the command line            */
/*                                                                    */
/*  Externals  : ISPFSCRN    - Load   get screen buffer & cursor posn */
/*             : CSRWORD     - exec   get the word at the cursor posn */
/*             : MSG2        - exec   invoking BookManager            */
/*             : MSGLIBS     - exec   allocating/freeing libraries    */
/*====================================================================*/
  swap = 'YES'                        /* swap screens for BookManager  */
  Address ISPEXEC                     /* commands go to ISPEXEC        */

  Parse Arg textstring .              /* get search argument           */
  If textstring = '' Then Do
   /*-------- INVOKE SCREEN GET PGM ---------------------------------*/
   /* REXX variable:           ISPF_SCREEN = SCREEN BUFFER          */
   /*                          ISPF_CURSOR = CURSOR OFFSET POSITION  */
   /*----------------------------------------------------------------*/
     "SELECT PGM(ISPFSCRN) SUSPEND"

   /*----------------------------------------------------------------*/
   /* get the message no. from screen text at the cursor position    */
   /*----------------------------------------------------------------*/
     valid_chars = 'abcdefghijklmnopqrstuvwxyz'!!,
                   'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789$'
     textstring = CSRWORD(ispf_screen,ispf_cursor,valid_chars)
     End

  /*---------------------*/
  /* Invoke BookManager */
  /*---------------------*/
  If swap = 'YES' Then
     "SELECT PGM(ISPSTRT) PARM(CMD(MSG2" textstring "))"
  Else
     "SELECT CMD(%MSG2" textstring ")"

  Return
```

MSG2 EXEC

```
/*=========================>> REXX <<================================*/
/*  MSG2: Invoke BookManager to search for a message                  */
/*          Called by: MSG1 EXEC                                      */
/*                                                                    */
/*          a) CHANGE msgshelf to the name of your message bookshelf  */
/*                                                                    */
/*          b) REMOVE "SELECT CMD(%MSGLIBS ..." if your BookManager   */
/*             libraries are permanently allocated                    */
/*====================================================================*/
  Parse Arg textstring             /* get search argument           */
  msgshelf = "MESSAGES"            /* BookManager Message bookshelf */
  Address ISPEXEC                  /* commands go to ISPEXEC        */
  "SELECT CMD(%MSGLIBS ALLOC)"     /* allocate libraries            */
  "CONTROL NONDISPL ENTER"         /* simulate ENTER key            */
  "SELECT CMD(%EOXVSTRT",
              msgshelf,            /* bookshelf with msg manuals    */
             "CMD(SEARCH "textstring"))",
             "MODE(FSCR) SUSPEND",
             "NEWAPPL(EOXR) PASSLIB"
  "SELECT CMD(%MSGLIBS FREE)"      /* deallocate libraries          */
  Return
```

MSGLIBS EXEC

```
/*==========================>> REXX <<================================*/
/* MSGLIBS: ALLOCATE OR FREE LIBRARIES FOR BOOKMANAGER                */
/*          Called by: MSG2 EXEC                                      */
/*                                                                    */
/*         CHANGE the library names to the correct ones for your site */
/*====================================================================*/
  Arg action
  Select
    When action = "FREE" Then Do
      "ALTLIB  DEACT APPLICATION(EXEC )"
      "ISPEXEC LIBDEF ISPTLIB"
      "ISPEXEC LIBDEF ISPMLIB"
      "ISPEXEC LIBDEF ISPPLIB"
      "ISPEXEC LIBDEF ISPLLIB"
      End

    Otherwise    /* default action = "ALLOC" */
      "ISPEXEC LIBDEF ISPLLIB DATASET ID(",
            " 'BOOK.ISPF.ISPLLIB' 'EOY.SEOYLOAD') STACK"

      "ISPEXEC LIBDEF ISPPLIB DATASET ID(",
            " 'BOOK.ISPF.ISPPLIB' 'EOY.SEOYPENU') STACK"

      "ISPEXEC LIBDEF ISPMLIB DATASET ID(",
            " 'BOOK.ISPF.ISPMLIB' 'EOY.SEOYMENU') STACK"

      "ISPEXEC LIBDEF ISPTLIB DATASET ID(",
            " 'BOOK.ISPF.ISPTLIB' 'EOY.SEOYTENU') STACK"

      "ALTLIB  ACT APPLICATION(EXEC ) DA(",
            " 'BOOK.ISPF.EXEC' 'EOY.SEOYCLIB')"
    End
  Return
```

ISPFSCRN PROGRAM

```
          TITLE 'ISPFSCRN - ISPF SCREEN BUFFER/CURSOR CAPTURE'
*------------* FUNCTION *-----------------
*   THIS PROGRAM WILL RETRIEVE THE ISPF SCREEN BUFFER AND RELATIVE
* CURSOR POSITION.
*
*   TWO REXX VARIABLES WILL BE SET.
*
*   ISPF_SCREEN = SIZE WILL BE CALCULATED BY USING ZSCREENW/ZSCREEND
*   ISPF_CURSOR = RELATIVE CURSOR POSITION WITHIN SCREEN BUFFER
*
*   INVOCATION OF PROGRAM:
*
*   ADDRESS ISPEXEC "SELECT PGM(ISPFSCRN)"
*>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
*>>>NOTE: THE CURSOR POSITION IS MOVED TO POSITION 152Ø OR 16ØØ
*>>>===== (DEPENDING ON SPLIT SCREEN) IF THE REXX CONTAINING THE
*>>>      SELECT PGM IS INVOKED AS A PROGRAM, MEANING INVOKED
*>>>      WITHOUT THE %-SIGN. SO THEREFORE WHEN ISPFSCRN IS USED MAKE
*>>>      SURE THAT THE REXX EXEC IS INVOKED AS %REXXPGM - ALSO GOOD
*>>>      FOR PERFORMANCE.
*-------------------------------------------------------------------
ISPFSCRN CSECT
ISPFSCRN AMODE 31
ISPFSCRN RMODE ANY
         REGS                    EQU REGISTERS
&PROG    SETC  'ISPFSCRN'
         SAVE  (14,12),,&PROG-&SYSDATE-&SYSTIME
         LR    R12,R15           LOAD BASEADDRESS
         USING ISPFSCRN,R12      BASE REGISTER ESTABLISHED
         LR    R4,R1             SAVE PARAMETER ADDRESS
         L     R2,WORKBYTS       SIZE OF THE USER PROGRAM
         GETMAIN R,LV=(2)        GET STORAGE FOR PROGRAM VARIABLES
         LTR   R15,R15           DID WE GET STORAGE ?
         BZ    GOTSTOR           YES - CARRY ON
         ABEND (R15)             SAVE GETMAIN RC
GOTSTOR  DS    ØH
         LR    R11,R1            R11 CONTAINS ADDRESS OF STORAGE
         USING PGMAREA,R11       ESTABLISH DATA AREA ADDRESSABILITY
         ST    R13,SAVEAREA+4    . SAVE AREA CHAINING
         LA    R13,SAVEAREA      . NEW SAVEAREA
         ST    R1,A_FMAIN        SAVE ADDRESS FOR FREEMAIN
         LR    R1,R4             RESTORE PARAMETER ADDRESS
**********************************************************************
****                      MAIN PROCEDURE                          ****
**********************************************************************
         L     R4,Ø(R1)         R4->  ISPF CONTROL BLOCK
```

```
GETCSR   DS    ØH
         L     R5,128(R4)         R5-> CURRENT SCREEN IMAGE
         LH    R6,166(R4)         R6 = OFFSET TO CURSOR
         LOAD  EP=IRXEXCOM        LOAD ADDRESS OF REXX VARIABLE INTF.
         ST    RØ,AIRXEXCOM
         MVC   SHVBLK(32),CSHVBLK PRE-FILL REXX SHVBLK
*-------------------------------------------------------------------*
*-           CALCULATE ISPF FULL SCREEN SIZE                       -*
*-------------------------------------------------------------------*
         LA    R2,ZSCREENW               SETUP PARMLIST
         CALL  ISPLINK,(VDEFINE,D_ZSCREENW,(R2),CHAR,L4),VL
         LA    R2,ZSCREEND               SETUP PARMLIST
         CALL  ISPLINK,(VDEFINE,D_ZSCREEND,(R2),CHAR,L4),VL
         CALL  ISPLINK,(VGET,ZVAR_LIST,SHARED),VL
         PACK  ZSCREENW,ZSCREENW         PREPARE FOR MULTIPLY ..
         PACK  ZSCREEND,ZSCREEND
         ZAP   MULT_RESULT,ZSCREEND      ENLARGE RESULT OF MULTIPLY
         MP    MULT_RESULT,ZSCREENW      OK DO IT ..
         CVB   R2,MULT_RESULT            PREPARE FOR REXX VARIABLE
*-------------------------------------------------------------------*
*-           CREATE VARIABLE ISPF_SCREEN                           -*
*-------------------------------------------------------------------*
         ST    R2,SHVVALL         LENGTH OF ISPF_SCREEN VARIABLE
         ST    R5,SHVVALA         R5 = ADDRESS OF ISPF SCREEN BUFFER
         LA    R3,CVAR1
         ST    R3,SHVNAMA          - ADDRESS OF VARIABLE NAME
         MVC   SHVNAML,=A(L'CVAR1) - LENGTH OF VARIABLE
         BAS   R1Ø,REXXVAR
*-------------------------------------------------------------------*
*-           CREATE VARIABLE ISPF_CURSOR                           -*
*-------------------------------------------------------------------*
         CVD   R6,D               R6 = CURSOR-OFFSET IN SCREEN BUFFER
         MVC   WK,=X'FØ2Ø2Ø2Ø2Ø2Ø2Ø2Ø'
         ED    WK,D+4             CONVERT TO ZONED DECIMAL
         LA    R3,WK
         ST    R3,SHVVALA         ADDRESS OF VARIABLE DATA
         MVC   SHVVALL,=A(L'WK)   LENGTH OF VARIABLE DATA
         LA    R3,CVAR2
         ST    R3,SHVNAMA          - ADDRESS OF VARIABLE NAME
         MVC   SHVNAML,=A(L'CVAR2) - LENGTH OF VARIABLE
         BAS   R1Ø,REXXVAR
**********************************************************************
****                     RETURN LINKAGE                           ****
**********************************************************************
         L     R13,4(R13)                RESTORE SAVE AREA
         LR    R1Ø,R15                   SAVE    RC
         L     R2,WORKBYTS               SIZE OF THE USER PROGRAM
         L     R3,A_FMAIN                ADDRESS OF GETMAIN STORAGE
         FREEMAIN R,LV=(2),A=(3)         FREE GETMAINED STORAGE
RESTORC  DS    ØH
         LR    R15,R1Ø                   RESTORE RC
         RETURN (14,12),RC=(15)          RETURN
```

```
*-------------------------------------------------------------------*
*-              CALL REXX VARIABLE INTERFACE                        -*
*-------------------------------------------------------------------*
REXXVAR  DS    ØH
         L     R15,AIRXEXCOM
         LA    R1,SHVBLK
         CALL  (15),(IRXSTR,Ø,Ø,(1)),VL
         BR    R1Ø
WORKBYTS DC    A(PGMSIZE)         AMOUNT OF WORKING STORAGE NEEDED
IRXSTR   DC    CL8'IRXEXCOM'      STRING FOR REXX VARIABLE INTERFACE
CVAR1    DC    CL11'ISPF_SCREEN'
CVAR2    DC    CL11'ISPF_CURSOR'
CSHVBLK  DC    A(Ø)                NEXT SHVBLOCK IN CHAIN - NONE
         DC    F'Ø'                LENGTH OF BUFFER FOR SHVNAMA
         DC    C'S'                CODE FOR FUNCTION S=SET VAIABLE
         DC    CL1'D'              DIRECT - NO SUBST. OR CASE XLATE
         DC    H'Ø'                RESERVED
         DC    F'Ø'                LENGTH OF BUFFER FOR FETCH REQUEST
         DC    A(Ø)                ADDRESS OF THE VARIABLE NAME
         DC    F'Ø'                LENGTH OF VARIABLE NAME
         DC    A(Ø)                ADDRESS OF VARIABLE-VALUE
         DC    F'Ø'                LENGTH OF VARIABLE-VALUE
D_ZSCREENW DC    C'ZSCREENW'       ISPF VARS DEFINITION
D_ZSCREEND DC    C'ZSCREEND'       ISPF VARS DEFINITION
ZVAR_LIST  DC    C'(ZSCREENW,ZSCREEND)' ISPF VGET LIST
CHAR       DC    C'CHAR'                ISPF VAR  TYPE PARAMETER
L4         DC    F'4'                   ISPF VAR  LGTH PARAMETER
VDEFINE    DC    CL8'VDEFINE'           ISPF SERVICE
VGET       DC    CL8'VGET'              ISPF SERVICE
SHARED     DC    CL8'SHARED'            ISPF SERVICE PARAMETER
         LTORG
PGMAREA  DSECT
SAVEAREA DS    18F                SAVE AREA FOR PROGRAM
*....... INSERT DS'S HERE
A_FMAIN  DS    F
         DS    ØD
D        DS    PL8                 CONVERT TO DECIMAL WORKAREA
WK       DS    CL8                 -
EDWK     DS    ØCL17               -
         DS    CL16                -
SIGN     DS    C                   -
EDWK_END DS    X                   END OF FIELD MARKER FOR EDWK
FULLSCR  DS    CL4                 # OF BYTES FOR FULL SCREEN
MULT_RESULT DS D                   RESULT OF SCREEN WIDTH*DEPTH
ZSCREENW DS    CL4                 ISPF VAR SCREEN WIDTH
ZSCREEND DS    CL4                 ISPF VAR SCREEN DEPTH
SHVBLK   DS    A                   NEXT SHVBLOCK IN CHAIN - NONE
SHVUSER  DS    F                   LENGTH OF BUFFER FOR SHVNAMA
SHVCODE  DS    C                   CODE FOR FUNCTION S=SET VAIABLE
SHVRET   DS    CL1                 DIRECT - NO SUBST. OR CASE XLATE
         DS    H                   RESERVED
```

```
SHVBUFL  DS    F                     LENGTH OF BUFFER FOR FETCH REQUEST
SHVNAMA  DS    A                     ADDRESS OF THE VARIABLE NAME
SHVNAML  DS    F                     LENGTH OF VARIABLE NAME
SHVVALA  DS    A                     ADDRESS OF VARIABLE-VALUE
SHVVALL  DS    F                     LENGTH OF VARIABLE-VALUE
AIRXEXCOM DS   F                     ADDRESS OF REXX VARIABLE MODULE
         DS    ØD
PGMSIZE  EQU   *-PGMAREA             WORKING STORAGE SIZE CALCULATION

         END   ISPFSCRN
```

An earlier version of MSG was sent to IBM in the early 1990s, but the only apparent result was the introduction in SDSF Release 4 of the BOOK command, which does *not* use the cursor position.

JES3 is used where I am currently working, plus EJES for the JES spool processing (it is like SDSF). EJES also defines a PF key with the value BOOK, so I have added the command book to my ISPF USERCMDS table to use the MSG command. Thus, the ISPF commands in Figure 1 are defined.

Command	*Table*	*Description / action*
BOOK	USER	Search for message in BookManager ALIAS MSG
ED	USER	Recursive EDIT SELECT CMD(%E &ZPARM) SUSPEND NEWAPPL(ISR)
MSG	USER	Search for message in BookManager SELECT CMD (%MSG1 ZPARM) NEWAPPL(ISR)

Table 1: Defined ISPF commands

Note that the BOOK command must be before the MSG command in the table for the ALIAS function to work correctly.

CONCLUSION

As you can see from these examples, there are some useful things you can do with cursor-sensitive commands. In a future article I will present a more extensive, cursor-sensitive command, which gets the screen buffer and cursor position using REXX code.

Ron Brown
Systems Programmer (Germany)

JES2 checkpoint sizing

THE PROBLEM

Recently I had to increase the number of jobs that JES could support at our site. Obviously, my first thought was: would the current checkpoint take the increase? My method of checking this was to simply look at the checkpoint on another of our LPARs, where I knew the number of allowable jobs was higher, and compare the relative sizes. As the checkpoint on the LPAR to be changed was three times the size of the other, it seemed a safe option to carry out the change. Unfortunately, when I started JES, I received the message £HASP537, telling me that my checkpoint was too small. My error turned out to be that I was looking at a catalogued version of the checkpoint on my reference LPAR, and not the uncatalogued one, which was actually being used. It was a case of a leftover from the OS/390 install.

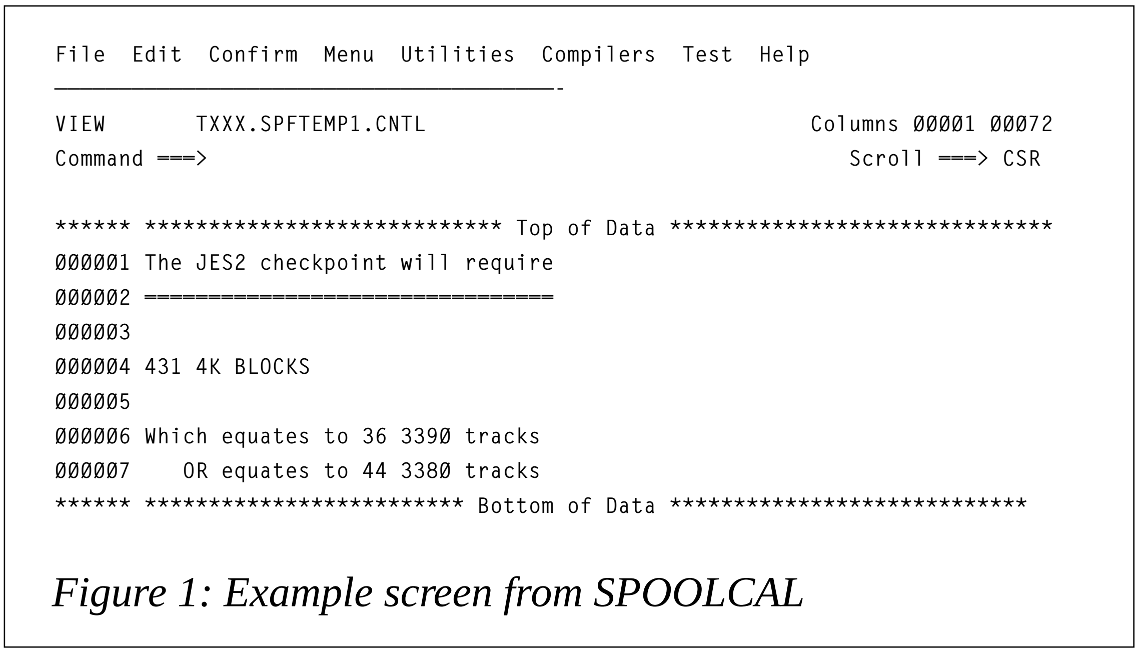
A SOLUTION

The result of making such a foolish error was that I went back to the manuals to ensure that I would not make the same mistake again. In the *JES2 Initialization and Tuning Guide* there is a detailed method for calculating the checkpoint in the same manner as JES does before issuing the £HASP537. In order to make this calculation easier, I have translated it into REXX and arranged for the REXX to attempt to scan SYS1.PARMLIB for the values to carry out the calculation.

Should you wish to use this REXX simply install it into your SYSPROC as member SPOOLCAL and issue the command TSO SPOOLCAL your.parmlib(jesparm) to obtain a screen like the one shown in Figure 1.

OPERATIONAL ENVIRONMENT

SPOOLCAL requires OS/390, JES2, and TSO/E to run.


```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
─────────────────────────────────────────────────────────-
VIEW       TXXX.SPFTEMP1.CNTL                            Columns 00001 00072
Command ===>                                                  Scroll ===> CSR

****** **************************** Top of Data ******************************
000001 The JES2 checkpoint will require
000002 ================================
000003
000004 431 4K BLOCKS
000005
000006 Which equates to 36 3390 tracks
000007    OR equates to 44 3380 tracks
****** ************************* Bottom of Data ****************************
```

Figure 1: Example screen from SPOOLCAL

SPOOLCAL REXX

```
/* REXX */
arg dsname
/* */
/* This REXX reads the JES2 parm member to pick up the necessary */
/* information to allow a calculation of the number of 4K blocks */
/* needed to estimate the JES checkpoint size                    */
/* */
x=OUTTRAP("save.") /* eliminate messages */
'FREE FI(SPONGE)'
"ALLOC FI(SPONGE) DA("dsname") SHR"
'EXECIO * DISKR SPONGE (FINIS'
DO QUEUED()
PULL line
IF INDEX(line,'TGSPACE=(MAX=')¬=0 THEN DO  /* max found */
   PARSE var line .'=(MAX=' max ')' .
   PARSE VAR max max ',' .
   END
IF INDEX(line,'JOENUM=')¬=0 THEN DO  /* max found */
   PARSE var line 'JOENUM=' joenum
   PARSE VAR joenum joenum ',' .
   END
IF INDEX(line,'JOBNUM=')¬=0 THEN DO  /* max found */
   PARSE var line 'JOBNUM=' jobnum
   PARSE VAR jobnum jobnum ',' .
   END
IF INDEX(line,'SPOOLNUM=')¬=0 THEN DO  /* max found */
   PARSE var line 'SPOOLNUM=' spoolnum
   PARSE VAR spoolnum spoolnum ',' .
   END
IF INDEX(line,'LOGSIZE=')¬=Ø THEN DO  /* max found */
   PARSE var line 'LOGSIZE=' logsize
   PARSE VAR logsize logsize ',' .
   END
END
"FREE FI(SPONGE)"
/* */
/* default corrections                                */
/* if logsize not specified assume 1                  */
/* max must be multiple of 16288                      */
/* */
IF logsize='' THEN logsize=1
rem=max//16288
IF rem¬=Ø THEN max=16288*((max%16288)+1)
/* now calculate the size of the ckpt */
/* CONSTANTS */
prefix=24 /* NUMBER OF BYTES FOR EACH CONTROL BLOCK */
rnd=Ø.5   /* rounding factor */
pg=4Ø96   /* size of a page in bytes */
/* */
/* ALL VALUES CALCULATED ARE IN BYTES. THESE NEED TO BE CONVERTED */
/* TO 4K BLOCKS, AND ALL FRACTIONS MUST BE ROUNDED UP.            */
/* */
tgm=(max/4)+prefix;tgm=FORMAT((tgm/pg)+rnd,,Ø)
scq=(32*32*16)+prefix;scq=FORMAT((scq/pg)+rnd,,Ø)
jix=(32767*2)+prefix;jix=FORMAT((jix/pg)+rnd,,Ø)
jobq=(jobnum+1)*(96+(spoolnum/8))+prefix;jobq=FORMAT((jobq/pg)+rnd,,Ø)
pst=(joenum*4)+prefix;pst=FORMAT((pst/pg)+rnd,,Ø)
jot=(joenum*1Ø4)+52Ø+prefix;jot=FORMAT((jot/pg)+rnd,,Ø)
tgr=(32*3*255)+prefix;tgr=FORMAT((tgr/pg)+rnd,,Ø)
rsØ=9999+prefix;rsØ=FORMAT((rsØ/pg)+rnd,,Ø)
lck=(56*8)+prefix;lck=FORMAT((lck/4Ø96)+rnd,,Ø)
das=(spoolnum*212)+prefix;das=FORMAT((das/4Ø96)+rnd,,Ø)
/* */
/* THEREFORE CHECKPOINT RECORDS IS */
/* */
total=tgm+scq+jix+jobq+pst+jot+tgr+rsØ+lck+das
/* */
/* NOW CALCULATE THE MASTER RECORD */
/* */
hct=58Ø;QSE=2ØØ*32;extension=4ØØØ
kit=1Ø*36;ckptio=4*total;dase=2*spoolnum
master_total=hct+QSE+extension+kit+ckptio+dase
master_total=FORMAT((master_total/pg)+rnd,,Ø)
/* */
/* NOW NEED THE SIZE OF THE CHANGE LOG */
/* */
logsize=1
/* */
/* THEREFORE THE total NUMBER OF 4K BLOCKS IS */
/* */
total=total+master_total+logsize
/* */
/* ALLOCATE A TEMPORARY FILE */
/* */
ADDRESS ISPEXEC
'FTOPEN TEMP'
'FTCLOSE'
'VGET ZTEMPN'
X=LISTDSI(ZTEMPN 'FILE')
ADDRESS TSO
/* */
/* CREATE THE INFORMATION */
/* */
QUEUE 'The JES2 checkpoint will require'
QUEUE '================================'
QUEUE ' '
QUEUE total '4K BLOCKS'
QUEUE ' '
QUEUE 'Which equates to' FORMAT((total/12)+rnd,,Ø) '339Ø tracks'
QUEUE '   OR equates to' FORMAT((total/1Ø)+rnd,,Ø) '338Ø tracks'
/* */
/* Now view the report */
/* */
'EXECIO' QUEUED() 'DISKW' ZTEMPN '(FINIS'
"ISPEXEC VIEW DATASET("sysdsname") VOLUME("sysvolume")"
```

Systems Programmer (UK) © Xephon 1999

An edit macro to add numbers to JCL cards

INTRODUCTION

It is common practice for certain utilities to refer to DD statements from SYSIN parameters. A typical example is IDCAMS, where the IFILE and OFILE parameters point to pre-defined DD statements. With the risks involved in mismatching these, it is a good idea to have a safe easy-to-use tool to generate these file numbers, rather than having to type them in manually. This is an example of such a deck of JCL cards:

```
//SAVELIZ EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//IØØ1 DD DISP=OLD,DSN=FIRST.DATASET.TO.SAVE
//IØØ2 DD DISP=OLD,DSN=SECOND.DATASET.TO.SAVE
//IØØ3 DD DISP=OLD,DSN=THIRD.DATASET.TO.SAVE
  etc
//OØØ1 DD DISP=(,CATLG),DSN=FIRST.BACKUP.DATASET(with SMS installed)
//OØØ2 DD DISP=(,CATLG),DSN=SECOND.BACKUP.DATASET
//OØØ3 DD DISP=(,CATLG),DSN=THIRD.BACKUP.DATASET
  etc
//SYSIN DD *
  REPRO IFILE(IØØ1) OFILE(OØØ1)
  REPRO IFILE(IØØ2) OFILE(OØØ2)
  REPRO IFILE(IØØ3) OFILE(OØØ3)
  etc
/*
```

OPERATIONAL ENVIRONMENT AND USAGE

The macro was developed under ISPF Version 4.1, but should run under any late version of the product. The macro can be saved with any 8-byte name and should be accessible through the SYSPROC concatenation under ISPF. (We have called it NUMERATE.)

It accepts three input parameters: <NMin>, <NMax>, and <LBn>, where '<NMin>' is the starting number and '<NMax>' the ending number.

(Note that < NMin > is altered to be in the same format as <Nmax>, eg NUMERATE 1 003 is the same as entering NUMERATE 001 003.)

'<LBn>' indicates the column number to add to the number on each line. This parameter is optional with a default value of 1.

A particular line/block command ('N') is provided to control the target lines.

Use 'N' (single line command) to INSERT the number list after the selected line. (Note the default: if the 'N' command is not supplied data is inserted before the first line.)

Use 'Nxxx' or 'NN - NN' (block command) to MERGE the number list over the selected lines. This is possible only if the position to be overwritten (at <LBn> column) is free (blank chars). A check is performed against each single line to verify that. If not, a message is given.

Note that if the block number of lines is smaller than the range <NMin> <NMax>, there are two possibilities:

- Stop merging numbers after the last line of the block is reached (this is the default).
- Keep on inserting numbers after the last line of the block is reached.

Refer to the comments in the REXX routine to set the default action to be taken.

EXAMPLE

```
Command ===> Numerate 1    Ø8    5                         Scroll===> CSR
****** ****************** Top of Data  ****************************
ØØØØØ1 //TESTJOB JOB (ACCT),'TEST NUMERATE'
ØØØØØ2 //*
nnØØØ3 //DD    DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER1)
ØØØØØ4 //DD    DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER2)
ØØØØØ5 //DD    DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER3)
ØØØØØ6 //DD    DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER4)
ØØØØØ7 //DD    DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER5)
nnØØØ8 //DD    DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER6)
ØØØØØ9 //*
****** ****************  Bottom of Data ***************************
```

With option 1 selected (stop merging at the end of the block) the following is the result:

```
     Command ===>                                          Scroll===> CSR
****** ****************** Top of Data  ****************************
ØØØØØ1 //TESTJOB JOB (ACCT),'TEST NUMERATE'
ØØØØØ2 //*
ØØØØØ3 //DDØ1  DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER1)
ØØØØØ4 //DDØ2  DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER2)
ØØØØØ5 //DDØ3  DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER3)
ØØØØØ6 //DDØ4  DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER4)
ØØØØØ7 //DDØ5  DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER5)
ØØØØØ8 //DDØ6  DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER6)
ØØØØØ9 //*
****** ****************  Bottom of Data ***************************
```

With option 2 selected (keep on inserting lines) the following is the result:

```
Command ===>                                               Scroll===> CSR
****** ****************** Top of Data  ****************************
ØØØØØ1 //TESTJOB JOB (ACCT),'TEST NUMERATE'
ØØØØØ2 //*
ØØØØØ3 //DDØ1  DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER1)
```

```
000004 //DD02  DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER2)
000005 //DD03  DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER3)
000006 //DD04  DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER4)
000007 //DD05  DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER5)
000008 //DD06  DD DISP=SHR,DSN=DATASET.TEST.NUMERATE(MEMBER6)
000009         07
000010         08
000011 //*
****** *****************  Bottom of Data ****************************
```

NUMERATE

```
/*  REXX                                                             */
/* ----------------------------------------------------------------- */
/*  DESCRIPTION: Creates a numeric list from <NMin> to <NMax>        */
/*               inserting numbers at column <LBn>                   */
/*                                                                   */
/*   - If an "N" line command (single) is present                    */
/*     the list is INSERTED after the specified line..               */
/*   - If an "Nxxx" line command or a "NN" block command is present */
/*     the list is MERGED over the selected lines.                   */
/*     In the latter case a check is performed against the line to  */
/*     ensure that the position to be overridden is filled with     */
/*     blanks, failing which a message is given.                     */
/*                                                                   */
/*  INPUT PARAMETERS:                                                */
/*        <NMin> Start number                                        */
/*                    <NMax> End number                              */
/*                    <LBn>  Left Bound (start column) - Optional    */
/*                                                                   */
/*  NOTES: The length of the NMin parm is filled with 'zero' chars  */
/*         to reach the length of NMax parm.                         */
/* ----------------------------------------------------------------- */
ADDRESS ISREDIT
"MACRO (NMin, NMax, LBn) NOPROCESS"
"NULLS OFF "
Block = 0

ADDRESS ISPEXEC "CONTROL ERRORS RETURN"

IF NMin > NMax THEN
  DO
    ZEDSMSG = 'Parameters ERROR '
    ZEDLMSG = 'NMIN greater then NMAX '
    ADDRESS ISPEXEC "SETMSG MSG(ISRZ001)"
    SIGNAL Exit_point
  END

IF LBn > '72' THEN
  DO
    ZEDSMSG = 'Bound ERROR '
    ZEDLMSG = 'Left bound MUST be < 72 '
    ADDRESS ISPEXEC "SETMSG MSG(ISRZØØ1)"
    SIGNAL Exit_point
  END

IF LBn = '' THEN LBn = '1'
FirstRow = 1                                    /* Member start-line */

"CURSOR = 1 1 "
"PROCESS B RANGE N"
SELECT
  WHEN RC = 4 THEN                 /* Line-command >> N << not found */
    DO
      Block = Ø
      StaLine = Ø
      StoLine = Ø
    END
  WHEN RC = 16 THEN
    DO
      ZEDSMSG = 'Invalid Block '
      ZEDLMSG = 'Block or Block-end (with >> NN <<) not found '
      ADDRESS ISPEXEC "SETMSG MSG(ISRZØØ1)"
      SIGNAL Exit_point
    END
  OTHERWISE
    DO
      "(Block) = RANGE_CMD "
      "(StaLine) = LINENUM .ZFRANGE "
      "(StoLine) = LINENUM .ZLRANGE "
      IF StaLine = StoLine THEN Block = Ø
      ELSE                      Block = 1
    END
END

IF Block THEN                               /* Block command (MERGE) */
  DO Num = NMin TO NMax

/* ---------------------------------------------------------------- *
 * -  Comment this statement and uncomment the following ones    - *
 * -  to insert the numbers declared with <NMax> parm             - *
 * -  that exceed the length of the Block                         - *
 * ---------------------------------------------------------------- *
 *                                                                  */
    IF StaLine > StoLine THEN LEAVE
/*                                                                 *
    IF StaLine > StoLine THEN
      DO
        ChkSpace = LENGTH(NMax)
        InsNum = RIGHT(RIGHT(Num,ChkSpace,Ø),LBn+ChkSpace-1)
        "LINE_BEFORE "StaLine" = '"InsNum"' "
        StaLine = StaLine + 1
        ITERATE
      END
 *                                                                        *
 * ———————————————————————————————————————————————————————————————————— */

    ChkSpace = LENGTH(NMax)
    "(SLine) = LINE "StaLine
    InsNum = RIGHT(Num,LENGTH(NMax),Ø)
    IF SUBSTR(SLine,LBn,ChkSpace) <> '' THEN
      DO
        ZEDSMSG = 'Insert ERROR '
        ZEDLMSG = 'Unable to insert number 'InsNum,
                  'on line 'StaLine,
                  'at column 'LBn
        ADDRESS ISPEXEC "SETMSG MSG(ISRZØØ1)"
        SIGNAL Exit_point
      END
    SLine = SUBSTR(SLine,1,LBn-1)||InsNum||SUBSTR(SLine,LBn+ChkSpace)
    "LINE "StaLine" = '"SLine"' "
    StaLine = StaLine + 1
  END
ELSE                                        /* Line command (INSERT) */
  DO Num = NMin TO NMax
    ChkSpace = LENGTH(NMax)
    InsNum = RIGHT(RIGHT(Num,ChkSpace,Ø),LBn+ChkSpace-1)
    "LINE_AFTER "StaLine" = '"InsNum"' "
    StaLine = StaLine + 1
  END

/* ———————————————-———————————————— */
Exit_point:
  "CURSOR = "StaLine" 1 "
  "NULLS ON STD "
  "MEND "

EXIT
```

Luciano Lorini
System programmmer
Cariverona Banca Spa (Italy)

Assembler instruction trace – part 4

This month we continue our look at the code for the Assembler instruction trace.

```
            XR    R1,R1
            IC    R1,XCELL+3
            N     R1,=F'15'
            SLL   R1,2
            LA    R1,REGTBL(R1)
            L     R2,Ø(,R1)
         ENDIF
         LM    RØ,R1,REGTBL
         L     R15,REGTBL+15*4
         STM   R14,R2,Ø(R3)
         MODEXIT
         EJECT
DO_EREG  MODENTRY
         LR    R1,R7
         STM   RØ,R14,TEMPREGS
         EREG  R2,R14
         STM   R2,R14,EREGSAVE+8-MYSAVE(R1)
         LM    R2,R14,TEMPREGS+8-MYSAVE(R1)
         IAC   R4
         SAC   Ø
         STAM  R2,R14,AR_WORK+8
         EREG  R15,R1
         STAM  RØ,R1,AR_WORK
         EAR   R2,R15
         ST    R2,AR_WORK+15*4
         LAM   R2,R14,=16F'Ø'
         SAC   Ø(R4)
         L     R2,CUR_PR
         LA    R3,PR_STACK
         IF    CR,R2,GE,R3
            MVC   EREGSAVE(8),8(R2)
            MVC   EREGSAVE+15*4(4),4(R2)
         ELSE
            STM   RØ,R1,EREGSAVE
            ST    R15,EREGSAVE+15*4
         ENDIF
         XR    R3,R3
         IC    R3,XCELL+3
         LR    R5,R3
         SRL   R3,4
         N     R5,=F'15'
         IF    CR,R5,GE,R3
            SR    R5,R3
            LA    R5,1(,R5)
            SLL   R5,2
            SLL   R3,2
            LA    R14,REGTBL(R3)
            LA    R15,EREGSAVE(R3)
            EX    R5,MOV_EREG
            LA    R14,AR_SAVE(R3)
            LA    R15,AR_WORK(R3)
            EX    R5,MOV_EREG
         ELSE
            LA    R1,16
            SR    R1,R3
            SLL   R1,2
            LR    R15,R3
            SLL   R15,2
            LR    RØ,R15                    .BACKUP R15
            LA    R14,REGTBL(R15)
            LA    R15,EREGSAVE(R15)
            EX    R1,MOV_EREG
            LR    R15,RØ                   .RESTORE R15
            LA    R14,AR_SAVE(R15)
            LA    R15,AR_WORK(R15)
            EX    R1,MOV_EREG
            LR    R15,RØ                   .RESTORE R15
            LA    R1,1(,R5)
            SLL   R1,2
            LA    R14,REGTBL
            LA    R15,EREGSAVE
            EX    R1,MOV_EREG
            LR    R15,RØ                   .RESTORE R15
            LA    R14,AR_SAVE(R15)
            LA    R15,AR_WORK(R15)
            EX    R1,MOV_EREG
         ENDIF
         MODEXIT
MOV_EREG MVC   Ø(Ø,R14),Ø(R15)
         EJECT
WTOFLGS  WTO   'ASMTRACE - FLAGS=XXXX ',MF=L
WTOFLGS_LEN    EQU *-WTOFLGS
PRT_B2   MODENTRY
         PERF  SHOWINST
         LA    R6,FIELDS
         SELECT EVERY
         WHEN  TM,FLAGS+1,B2R1BIT,O     .GR1 USED (SUBCHAN CMDS)?
            LA    R3,X'1Ø'
            PERF  SHOW_GRS
            LA    R6,FIELDS
         WHEN  TM,FLAGS+1,B2RBIT,O      .OTHER REG USED?
            IC    R3,XCELL+3
            PERF  REG_OPS
            LR    R5,R6
            PERF  SHOW_GRS
         WHEN  TM,FLAGS+1,B2RØBIT,O          .GPRØ IMPLIED USE
```

```
            MVC   DR2B+12(3),=C'RØ='
            UNPK  DR2B+15(9),REGTBL(5)
            MVI   DR2B+15+8,X'4Ø'
            TR    DR2B+15(8),HEXCHAR-C'Ø'
         WHEN  TM,FLAGS+1,B2ADRBIT+B2STGBIT,NZ    .STORAGE/ADDR REF?
            LA    R3,XCELL+2
            PERF  SHOW_BD
            SELECT
            WHEN  TM,FLAGS+1,B2ADRBIT,O
               XR    R5,R5
            WHEN  TM,FLAGS,DBLBIT,O
               LA    R5,8
            WHEN  NONE
               LA    R5,4
            ENDSEL
            SHOW_EFA FROM=(XCELL+2),TO=(EFA1-1)  .LENGTH IN R5
         WHEN  CLI,XCELL+1,EQ,X'18'    .PC?
            PERF  SHOW_PC
         ENDSEL
         MODEXIT
         EJECT
SHOW_PC  MODENTRY
         LH    R1,XCELL+2
         LR    R2,R1
         SRL   R1,12                       .DROP DISPLACEMENT
         N     R1,=F'15'                   .DROP POSS. NEG. BITS
         SLL   R1,2                        .MULTIPLY BY 4
         LA    R1,OLDREGS(R1)              .POINT INTO REG. TBL
         L     R3,Ø(,R1)                   .GET PC BASE REG
         N     R2,=A(X'FFF')               .GET INTR. OFFSET
         AR    R3,R2                       .ADD TO BASE REG
         LR    R2,R3                       .BACKUP PC-NO
         ST    R3,DUB                      .DISPLAY PC-NO
         UNPK  GR_1(9),DUB(5)
         MVI   GR_1+8,X'4Ø'
         TR    GR_1(8),HEXCHAR-C'Ø'
         MVC   GR_1-5(5),=C'PCNO='
         SRL   R3,8                        .DROP OFF ENTRY NO
         IF    C,R3,LT,=A(HI_LX)
            SLL   R3,2
            A     R3,=A(LXLIST)
            L     R15,Ø(,R3)               .GET A(LX ENTRIES)
            N     R2,=F'255'               .KEEP ONLY ENTRY NO
            IF    C,R2,LT,Ø(,R15)          .ENTRY NO < MAX ENTRIES?
               MH    R2,=H'3Ø'
               LA    R2,4(R2,R15)
               MVC   DR1(3Ø),Ø(R2)
            ENDIF
         ENDIF
         MODEXIT
         EJECT
EXEC_SSM MODENTRY NEWBASE=R1Ø
```

```
         MVC   CODEFLD(4),XCELL
         LH    R1,XCELL+2
         IF    N,R1,=A(X'FØØØ'),NZ
            SRL   R1,12-2
            LR    R3,R1
            L     R4,REGTBL(R1)
            NI    CODEFLD+2,X'ØF'
            OI    CODEFLD+2,X'4Ø'
            IF    IAC,R4,NZ
               LA    R15,AR_SAVE(R1)
               LAM   AR4,AR4,Ø(R1)
            ENDIF
         ENDIF
         RUN_INST
         PERF  SHOWINST
         LA    R3,XCELL+2
         LA    R6,FIELDS
         PERF  SHOW_BD
         SHOW_EFA FROM=(XCELL+2),FOR=1,TO=(SS_EFA1-1)
         IF    IAC,R14,NZ
            MVC   AR_LINE,PRTLINE
            MVC   PRTLINE,=CL133' '
            SHOW_AR TO=(SS_EFA1-5),FROM=(XCELL+2)
            MVC   I_PTR(35),=C'RELATED ACCESS REGS FOR ABOVE INSTR'
         ENDIF
         MODEXIT
         EJECT
EXEC_PLO MODENTRY NEWBASE=R1Ø
         MVC   CODEFLD(6),XCELL
         MVI   CODEFLD+1,X'ØØ'
         PERF  PLO_PRIME_REGS
         LH    R1,CODEFLD+2
         IF    N,R1,=A(X'FØØØ'),NZ
            SRL   R1,12-2
            L     R14,REGTBL(R1)
            NI    CODEFLD+2,X'ØF'
            OI    CODEFLD+2,X'EØ'
            IF    IAC,R6,NZ
               LA    R1,AR_SAVE(R1)
               LAM   AR14,AR14,Ø(R1)
            ENDIF
         ENDIF
         LH    R1,CODEFLD+4
         IF    N,R1,=A(X'FØØØ'),NZ
            SRL   R1,12-2
            L     R15,REGTBL(R1)
            OI    CODEFLD+4,X'FØ'
            IF    IAC,R6,NZ
               LA    R1,AR_SAVE(R1)
               LAM   AR15,AR15,Ø(R1)
            ENDIF
         ENDIF
         LM    RØ,R1,REGTBL
         RUN_INST
         STM   RØ,R1,REGTBL
         LAM   AR14,AR15,=2F'Ø'
         PERF  PLO_UPDATE_REGS
         PERF  SHOWINST
         PERF  PLO_SHOW_OPS
         PERF  PLO_PRT_REGS
         PERF  PLO_PRT_B2D2
         MODEXIT
         EJECT
PLO_SHOW_OPS   MODENTRY NEWBASE=R1Ø,BAKR=YES
         LA    R6,FIELDS
         IC    R3,XCELL+1
         PERF  REG_OPS
         MVI   FIELDS+3,C','
         LA    R6,FIELDS+4
         LA    R3,XCELL+2
         PERF  SHOW_BD
         MVI   Ø(R6),C','
         LA    R6,1(,R6)
         IC    R3,XCELL+1
         PERF  REG_OP2
         MVI   Ø(R6),C','
         LA    R6,1(,R6)
         LA    R3,XCELL+4
         PERF  SHOW_BD
         MODEXIT
         EJECT
PLO_UPDATE_REGS    MODENTRY NEWBASE=R1Ø,BAKR=YES
         XR    R1,R1
         IC    R1,XCELL+1
         LR    R15,R1
         IF    N,R1,=A(X'FØ'),NZ
            SRL   R1,4-2
            LA    R1,REGTBL(R1)
            IF    TM,XCELL+1,X'1Ø',O
               ST    R3,Ø(,R1)
            ELSE
               STM   R2,R3,Ø(R1)
            ENDIF
         ENDIF
         IF    N,R15,=A(X'ØF'),NZ
            SLL   R15,2
            LA    R1,REGTBL(R15)
            IF    TM,XCELL+1,X'Ø1',O
               ST    R5,Ø(,R1)
            ELSE
               STM   R4,R5,Ø(R1)
            ENDIF
         ENDIF
         MODEXIT
         EJECT
PLO_PRT_REGS   MODENTRY NEWBASE=R1Ø,BAKR=YES
         MVC   GR_1(7),=C'GRØ-1: '
         UNPK  GR_1+6(9),OLDREGS(5)
         MVI   GR_1+6+8,X'4Ø'
         TR    GR_1+6(8),HEXCHAR-C'Ø'
         UNPK  GR_1+15(9),OLDREGS+4(5)
         MVI   GR_1+15+8,X'4Ø'
         TR    GR_1+15(8),HEXCHAR-C'Ø'
         IF    TM,OLDREGS+3,1,Z              .EVEN FUNCTION CODE?
            PERF  PLO_SHOW_R1                .YES- SHOW
         ENDIF
         IF    CLI,OLDREGS+3,EQ,Ø,OR,CLI,OLDREGS+3,EQ,8,OR,            +
               CLI,OLDREGS+3,EQ,12,ORIF,                               +
               CLI,OLDREGS+3,NE,4,AND,CLI,OLDREGS+3,NE,5,AND,          +
               IAC,R1,NZ
            IF    CLI,GR_1+26,NE,X'4Ø'
               PERF  WRITE
            ENDIF
            PERF  PLO_SHOW_R3
         ENDIF
         MODEXIT
         EJECT
PLO_SHOW_R1    MODENTRY NEWBASE=R1Ø,BAKR=YES
         IC    R15,XCELL+1
         N     R15,=A(X'FØ')
         SRL   R15,4
         CVD   R15,DUB
         OI    DUB+7,X'ØF'
         MVC   GR_1+26(4),=C'GRØØ'
         UNPK  GR_1+28(2),DUB+6(2)
         IF    TM,XCELL+1,X'1Ø',O         .ODD R1?
            MVI   GR_1+3Ø,C':'
            LA    R6,GR_1+32
         ELSE
            MVI   GR_1+3Ø,C'-'
            MVC   GR_1+31(2),GR_1+42
            OI    GR_1+32,1
            MVI   GR_1+33,C':'
            LA    R6,GR_1+35
         ENDIF
         SLL   R15,2
         LA    R15,REGTBL(R15)
         UNPK  Ø(9,R6),Ø(5,R15)
         MVI   8(R6),X'4Ø'
         TR    Ø(8,R6),HEXCHAR-C'Ø'
         IF    TM,XCELL+1,X'1Ø',Z            .EVEN R1
            UNPK  9(9,R6),4(5,R15)
            MVI   17(R6),X'4Ø'
            TR    9(8,R6),HEXCHAR-C'Ø'
         ENDIF
         MODEXIT
         EJECT
PLO_SHOW_R3    MODENTRY NEWBASE=R1Ø,BAKR=YES
         IC    R1,XCELL+1
         N     R1,=A(X'ØF')
         MVC   GR_1+26(4),=C'GRØØ'
         IF    TM,XCELL+1,X'Ø1',Z           .EVEN R1
            MVI   GR_1+3Ø,C':'
            LA    R6,GR_1+32
         ELSE
            MVI   GR_1+3Ø,C'-'
            MVC   GR_1+31(2),GR_1+42
            OI    GR_1+32,1
            MVI   GR_1+33,C':'
            LA    R6,GR_1+35
         ENDIF
         CVD   R1,DUB
         OI    DUB+7,X'ØF'
         UNPK  GR_1+28(2),DUB+6(2)
         SLL   R1,2
         LA    R1,REGTBL(R1)
         UNPK  GR_1+32(9),Ø(5,R1)
         MVI   GR_1+4Ø,X'4Ø'
         TR    GR_1+32(8),HEXCHAR-C'Ø'
         IF    TM,XCELL+1,X'Ø1',Z           .EVEN R1
            UNPK  9(9,R6),4(5,R1)
            MVI   17(R6),X'4Ø'
            TR    9(8,R6),HEXCHAR-C'Ø'
         ENDIF
         MODEXIT
         EJECT
PLO_PRT_B2D2   MODENTRY NEWBASE=R1Ø,BAKR=YES
         MODEXIT
         EJECT
EXEC_E5  MODENTRY NEWBASE=R1Ø LIST=YES
         MVC   CODEFLD(6),XCELL
         XR    R1,R1
         IC    R1,XCELL+1
         SLL   R1,1
         A     R1,=A(E5FLAGS)
         MVC   FLAGS,Ø(R1)
         LH    R1,CODEFLD+2
         LR    R15,R1
         IF    N,R1,=A(X'FØØØ'),NZ
            N     R15,=A(X'FFF')
            O     R15,=A(X'2ØØØ')
            STH   R15,CODEFLD+2
            SRL   R1,12-2
            LA    R15,REGTBL(R1)
            LM    R2,R3,Ø(R15)
            IF    IAC,R14,NZ
               LA    R15,AR_SAVE(R1)
               LAM   R2,R3,Ø(R15)
```

```
   ENDIF
ENDIF
LH    R1,CODEFLD+4
LR    R6,R1
IF    N,R1,=A(X'FØØØ'),NZ
   N     R6,=A(X'FFF')
   O     R6,=A(X'4ØØØ')
   STH   R6,CODEFLD+4
   SRL   R1,12-2
   LA    R6,REGTBL(R1)
   LM    R4,R5,Ø(R6)
   IF    IAC,R14,NZ
      LA    R15,AR_SAVE(R1)
      LAM   R2,R3,Ø(R15)
   ENDIF
ENDIF
SELECT
WHEN  CLI,CODEFLD+1,EQ,X'ØE',OR,CLI,CODEFLD+1,EQ,X'ØF'
   LM    RØ,R1,REGTBL              .MVCSK/MVCDK
WHEN  CLI,CODEFLD+1,GT,1
   B     ILGLOP
ENDSEL
RUN_INST
PERF  SHOWINST
LA    R3,XCELL+2
LA    R6,FIELDS
PERF  SHOW_BD
MVI   Ø(R6),C','
LA    R6,1(,R6)
IF    TM,XCELL+4,X'FØ',Z
   MVC   Ø(7,R6),=C'X''1234'''
   UNPK  2(5,R6),XCELL+4(3)
   MVI   6(R6),C''''
   TR    2(4,R6),HEXCHAR-C'Ø'
ELSE
   LA    R3,XCELL+4
   PERF  SHOW_BD
ENDIF
SELECT
WHEN  CLI,XCELL+1,EQ,Ø             .LASP
   SHOW_EFA FROM=(XCELL+2),FOR=8,TO=(SS_EFA1-1)
   SHOW_EFA FROM=(XCELL+4),FOR=Ø,TO=(SS_EFA2-1)
WHEN  CLI,XCELL+1,EQ,1             .TPROT
   SHOW_EFA FROM=(XCELL+2),FOR=Ø,TO=(SS_EFA1-1)
   MVC   SS_EFA1+1Ø(4),=C'KEY='
   LH    R1,XCELL+4
   LR    R2,R1
   IF    N,R1,=A(X'FØØØ'),NZ
      SRL   R1,12-2
      LA    R1,OLDREGS(R1)
      L     R1,Ø(,R1)
   ENDIF
```

```
            N     R2,=A(X'FFF')
            AR    R1,R2
            SRL   R1,4
            N     R1,=F'15'
            CVD   R1,DUB
            OI    DUB+7,X'ØF'
            UNPK  SS_EFA1+14(2),DUB+6(2)
            MVC   SS_EFA1+2Ø(7),=C'RESULT:'
            SELECT
            WHEN  TM,REALCC,X'3Ø',O
               MVC   SS_EFA1+2Ø+8(15),=C'TRANSLATION N/A'
            WHEN  CC=8               .COND=ZEROS
               MVC   SS_EFA1+2Ø+8(11),=C'UNPROTECTED'
            WHEN  TM,REALCC,X'2Ø',O
               MVC   SS_EFA1+2Ø+8(9),=C'PROTECTED'
            WHEN  NONE
               MVC   SS_EFA1+2Ø+8(15),=C'FETCH PROTECTED'
            ENDSEL
         WHEN  CLI,XCELL+1,EQ,X'ØE'          .MVCSK
            L     R5,OLDREGS
            N     R5,=F'255'
            AL    R5,=F'1'
            SHOW_EFA FROM=(XCELL+2),MAX=21,TO=(SS_EFA1-1)
            SHOW_EFA FROM=(XCELL+4),FOR=Ø,TO=(SS_EFA3-1)
            PERF  WRITE
            MVC   SS_EFA1+2Ø(22),=C'SOURCE KEY FROM GR1 = '
            PERF  E5_KEY_LEN
         WHEN  CLI,XCELL+1,EQ,X'ØF'          .MVCDK
            L     R5,OLDREGS
            N     R5,=F'255'
            AL    R5,=F'1'
            SHOW_EFA FROM=(XCELL+4),MAX=21,TO=(SS_EFA1-1)
            SHOW_EFA FROM=(XCELL+4),FOR=Ø,TO=(SS_EFA3-1)
            PERF  WRITE
            MVC   SS_EFA1+2Ø(22),=C'DEST.  KEY FROM GR1 = '
            PERF  E5_KEY_LEN
         ENDSEL
         LA    R9,6(,R9)
         MODEXIT
         EJECT
E5_KEY_LEN     MODENTRY NEWBASE=R1Ø,BAKR=YES,LIST=YES
         MVC   SS_EFA1(15),=C'LEN FROM GRØ = '
         L     R1,OLDREGS
         N     R1,=F'255'
         LA    R1,1(,R1)
         CVD   R1,DUB
         OI    DUB+7,X'ØF'
         UNPK  SS_EFA1+15(3),DUB+6(2)
         L     R1,OLDREGS+4
         SRL   R1,4
         N     R1,=F'15'
         CVD   R1,DUB
         OI    DUB+7,X'ØF'
         UNPK  SS_EFA1+42(2),DUB+6(2)
         MODEXIT
         EJECT
EXEC_A7  MODENTRY NEWBASE=R1Ø
         XR    R1,R1
         IC    R1,XCELL+1
         LR    R3,R1                     .HI-NIBBLE, REGISTER OR CC
         SRL   R3,4
         LR    R15,R3                    .BACKUP FOR BRC
         N     R1,=F'15'                 .LOW-NIBBLE, PART OF OPCODE
         LR    R2,R1
         SLL   R1,1
         A     R1,=A(A7FLAGS)
         MVC   FLAGS,Ø(R1)
         SLL   R3,2
         LA    R1,REGTBL(R3)
         SELECT
         WHEN  TM,FLAGS,ILGLBIT,O
            PERF  ILGLOP
         WHEN  TM,FLAGS,BRBIT,Z
            LM    R4,R5,Ø(R1)
            NI    CODEFLD+1,X'ØF'        .LOW-NIBBLE PART OF OPCODE
            OI    CODEFLD+1,X'4Ø'        .USE R4, REGARDLESS
            RUN_INST
            STM   R4,R5,Ø(R1)
            LA    R9,4(,R9)
         WHEN  TM,XCELL+1,5,O            .BRAS - BR. RELATIVE & SAVE
            LA    R14,4(,R9)
            BSM   R14,Ø
            ST    R14,Ø(,R1)
            LH    R14,XCELL+2
            SLA   R14,1
            AR    R9,R14
         WHEN  TM,XCELL+1,6,O            .BRCT - B REL. ON CNT
            L     R2,Ø(,R1)
            IF    S,R2,=F'1',Z
               LA    R9,4(,R9)
            ELSE
               LH    R15,XCELL+2
               SLA   R15,1
               AR    R9,R15
            ENDIF
            ST    R2,Ø(,R1)
         WHEN  NONE                     .BRC - B REL. ON COND
            XR    R14,R14
            IC    R14,REALCC            .GET CC + PGM MASKS
            SRL   R14,4                 .DROP PGM MASKS
            IC    R14,HEXCC(R14)        .GET TESTABLE CC
            IF    NR,R14,R15,Z          .NO CORRESPONDING BITS
               LA    R9,4(,R9)
            ELSE
```

```
               LH    R15,XCELL+2            .NO OF HALF-WORDS
               SLA   R15,1                  .TO BRANCH (POS OR NEG)
               AR    R9,R15
            ENDIF
         ENDSEL
         PERF  SHOWINST
         IF    TM,FLAGS,BRBIT,Z
            IC    R3,XCELL+1
            PERF  SHOW_GRS
            IC    R3,XCELL+1
            PERF  REG_OPS
         ELSE
            IC    R1,XCELL+1            .THEN SHOW
            SRL   R1,4                  .WHICH COND WAS
            N     R1,=F'15'             .REQUESTED
            IC    R1,HEXCHAR(R1)
            STC   R1,FIELDS
            MVI   FIELDS+1,C','
            LA    R6,FIELDS+2
         ENDIF
         MVC   Ø(7,R6),=C'X''CCCC'''
         UNPK  2(5,R6),CODEFLD+2(3)
         MVI   2+4(R6),C''''
         TR    2(4,R6),HEXCHAR-C'Ø'
         MODEXIT
         EJECT
EXEC_UPT MODENTRY NEWBASE=R1Ø
         MODEXIT
         EJECT
EXEC_BR  MODENTRY  NEWBASE=R1Ø .INSTRUCTION WHICH MAY GEN BRANCH,
         SELECT    ,           .EXCEPT BC,BCR,BXH,BXLE
         WHEN  CLI,XCELL,EQ,X'45',OR,CLI,XCELL,EQ,5      .BAL,BALR
            PERF  EXEC_BAL
         WHEN  CLI,XCELL,EQ,X'46',OR,CLI,XCELL,EQ,6      .BCT,BCTR
            PERF  EXEC_BCT
         WHEN  CLI,XCELL,EQ,X'4D',OR,CLI,XCELL,EQ,X'ØD'  .BAS,BASR
            PERF  EXEC_BAS
         WHEN  CLI,XCELL,EQ,X'ØB'                        .BSM
            PERF  EXEC_BSM
         WHEN  CLI,XCELL,EQ,X'ØC'                        .BASSM
            PERF  EXEC_BASSM
         ENDSEL
         PERF  SHOWINST
         IC    R3,XCELL+1
         PERF  REG_OPS
         PERF  SHOW_GRS
         IF    TM,FLAGS+1,RXBIT,O
            LA    R6,FIELDS+4
            LA    R3,XCELL+2
            PERF  SHOW_BD
            SHOW_EFA TO=(EFA1-1),FOR=Ø,FROM=(XCELL+2)
         ENDIF
         MODEXIT
         EJECT
EXEC_BAL MODENTRY
         IF    TM,FLAGS+1,RRBIT,O
            LA    R3,2(,R9)
            L     RØ,=X'4ØØØØØØØ'       .BALR IN 24-BIT MODE SETS BIT 1
         ELSE
            LA    R3,4(,R9)
            L     RØ,=X'8ØØØØØØØ'       .24-BIT BAL, SETS BIT Ø
         ENDIF
         TST31 R3                       .IF 31-BIT MODE, R3 WILL BE NEG
         IF    LTR,R3,R3,NM             .NOT NEGATIVE, SO 24-BIT
            OR    R3,RØ                 .OR IN ILC
            ICM   RØ,B'1ØØØ',REALCC     .GET COND CODE + PGM MASKS
            OR    R3,RØ                 .AND OR IN
         ENDIF
         SELECT
         WHEN  TM,FLAGS+1,RXBIT,O       .RX = BAL
            LA    R8,XCELL+2
            PERF  EVALBD
            LR    R9,R1
         WHEN  TM,XCELL+1,X'ØF',Z       .BALR, DEST REG=RØ
            LA    R9,2(,R9)             .NO BRANCH GENNED
         WHEN  NONE                     .BALR, VALID REG
            XR    R1,R1
            IC    R1,XCELL+1
            N     R1,=F'15'
            SLL   R1,2                  .POINT NEW INSTR PTR, BUT USE
            L     R9,OLDREGS(R1)        .OLDREGS TO CATER FOR BALR 14,14
         ENDSEL
         XR    R1,R1
         IC    R1,XCELL+1
         N     R1,=XL4'FØ'
         SRL   R1,2
         ST    R3,REGTBL(R1)            .STORE LINK REG
         MODEXIT
         EJECT
EXEC_BCT MODENTRY
         IC    R2,XCELL+1
         N     R2,=XL4'FØ'
         SRL   R2,2
         L     R3,REGTBL(R2)
         SELECT
         WHEN  S,R3,=F'1',Z             .CAN'T USE BCTR, NO CC GENNED
            LA    R9,2(,R9)             .ADD 2 TO INSTR PTR (BCTR)
            IF    TM,FLAGS+1,RXBIT,O    .WAS IT REALLY A BCT?
               LA    R9,2(,R9)          .YES, ADD ANOTHER 2 TO INST PTR
            ENDIF
         WHEN  TM,FLAGS+1,RXBIT,O       .BCT, REG NOT ZERO
            LA    R8,XCELL+2
            PERF  EVALBD
            LR    R9,R1
         WHEN  TM,XCELL+1,B'1111',Z    .BCTR, DEST = RØ
            LA    R9,2(,R9)
         WHEN  NONE                    .BCTR, DEST REG = VALID
            XR    R1,R1
            IC    R1,XCELL+1
            N     R1,=F'15'
            SLL   R1,2
            L     R9,REGTBL(R1)
         ENDSEL
         ST    R3,REGTBL(R2)
         MODEXIT
         EJECT
EXEC_BAS MODENTRY
         IF    TM,FLAGS+1,RRBIT,O      .BASR, NOT BAS
            LA    R3,2(,R9)            .USE R3 AS LINK REG
            IF    TM,XCELL+1,15,Z      .BASR RX,RØ
               LR    R9,R3             .NO BRANCH
            ELSE
               IC    R1,XCELL+1
               N     R1,=F'15'
               SLL   R1,2
               L     R9,REGTBL(R1)     .SET NEW INST PTR
            ENDIF
         ELSE
            LA    R3,4(,R9)            .LINK REG, BAS
            LA    R8,XCELL+2           .GET DEST ADDR.
            PERF  EVALBD
            LR    R9,R1
         ENDIF
         TST31 R3                      .WILL SET HI-BIT IF IN 31-BIT
         IC    R1,XCELL+1
         N     R1,=A(X'FØ')
         SRL   R1,4-2
         ST    R3,REGTBL(R1)           .UPDATE CORRECT LINK REG
         MODEXIT
         EJECT
EXEC_BSM MODENTRY
         IF    TM,XCELL+1,B'1111ØØØØ',NZ      .LINK REG NOT ZERO?
            IC    R1,XCELL+1
            N     R1,=XL4'FØ'
            SRL   R1,2
            L     R3,REGTBL(R1)
            BSM   R3,Ø
            ST    R3,REGTBL(R1)
         ENDIF
         PERF  CHGMODE
         MODEXIT
         SPACE 3
EXEC_BASSM MODENTRY
         LA    R3,2(,R9)
         BSM   R3,Ø
         IC    R1,XCELL+1
         N      R1,=XL4'FØ'
         SRL    R1,2
         ST     R3,REGTBL(R1)           .UPDATE CORRECT LINK REG
         PERF   CHGMODE
.BASSMØ1 ANOP
         MODEXIT
         SPACE 3
CHGMODE  MODENTRY
         IF     TM,XCELL+1,15,Z         .DEST REG = RØ?
            LA     R9,2(,R9)            .GO TO N.S.I.
         ELSE
            IC     R2,XCELL+1           .GET VALUE OF DEST REG
            N      R2,=F'15'
            SLL    R2,2
            L      R9,OLDREGS(R2)       .GET NEW INSTR PTR
            LA     R15,CHG_MODE         .PRIME R15 FOR BSM
            LR     R14,R9               .COPY DEST REG
            N      R14,=A(X'8ØØØØØØØ')  .RETAIN JUST AMODE BIT
            OR     R15,R14              .SET AMODE INTO R15
            BSM    Ø,R15                .AND SWITCH MODE
         ENDIF
CHG_MODE DS     ØH
         MODEXIT
         EJECT
EXEC_FLT MODENTRY  NEWBASE=R1Ø
         IF     CLI,XCELL,NE,X'B2',AND,TM,FLAGS+1,RXBIT,O
            LH     R1,XCELL+2
            LR     R15,R1
            IF     N,R1,=XL4'FØØØ',NZ
               N      R15,=F'4Ø95'
               SRL    R1,12
               SLL    R1,2
               L      R6,REGTBL(R1)
               O      R15,=XL4'6ØØØ'
               STH    R15,CODEFLD+2
            ENDIF
            IC     R1,XCELL+1
            IF     N,R1,=F'15',NZ
               SLL    R1,2
               L      R2,REGTBL(R2)
               IC     R1,XCELL+1
               N      R1,=XL4'FØ'
               O      R1,=F'2'
               STC    R1,CODEFLD+1
            ENDIF
         ENDIF
         RUN_INST
         STD    RØ,FLTRØ
         STD    R2,FLTR2
         STD    R4,FLTR4
         STD    R6,FLTR6
         PERF   PRT_FLT
         LA    R9,2(,R9)
         IF    TM,FLAGS+1,RXBIT+RSBIT,NZ
            LA    R9,2(,R9)
         ENDIF
         MODEXIT
         EJECT
PRT_FLT  MODENTRY
         PERF  SHOWINST
         IF    CLI,XCELL,EQ,X'B2'
            IC    R3,XCELL+3
            PERF  REG_OPS
         ELSE
            IC    R3,XCELL+1
            PERF  REG_OPS
            IF    TM,FLAGS+1,RXBIT,O
               LA    R3,XCELL+2
               PERF  SHOW_BD
            ENDIF
         ENDIF
         XR    R3,R3
         IC    R3,XCELL+1
         LR    R1,R3
         SRL   R1,2
         LA    R1,FLTREGS(R1)
         UNPK  GR_1(9),Ø(5,R1)
         UNPK  GR_1+8(9),4(5,R1)
         IF    TM,FLAGS+1,EXBIT,O
            UNPK  GR_1+16(9),8(5,R1)
            UNPK  GR_1+24(9),12(5,R1)
            MVI   GR_1+32,X'4Ø'
            TR    GR_1(32),HEXCHAR-C'Ø'
         ELSE
            MVI   GR_1+16,X'4Ø'
            TR    GR_1(16),HEXCHAR-C'Ø'
         ENDIF
         SELECT
         WHEN  CLI,XCELL,EQ,X'B2'
            PERF  FLTREG2
         WHEN  TM,FLAGS+1,RRBIT,O
            PERF  FLTREG2
         WHEN  NONE
            LA    R8,XCELL+2
            SHOW_EFA FROM=(XCELL+2),FOR=8,TO=(EFA1-1)
         ENDSEL
         MODEXIT
         EJECT
FLTREG2  MODENTRY
         LR    R1,R3
         N     R1,=F'15'
         SLL   R1,2
         LA    R1,FLTREGS(R1)
         UNPK  FR2(9),Ø(5,R1)
         UNPK  FR2+8(9),4(5,R1)
         IF    TM,FLAGS+1,EXBIT,O
            UNPK  FR2+16(9),8(5,R1)
            UNPK  FR2+24(9),12(5,R1)
            MVI   FR2+32,X'4Ø'
            TR    FR2(32),HEXCHAR-C'Ø'
         ELSE
            MVI   FR2+16,X'4Ø'
            TR    FR2(16),HEXCHAR-C'Ø'
         ENDIF
         MODEXIT
         EJECT
EXEC_RS  MODENTRY  NEWBASE=R1Ø
         LH    R1,CODEFLD+2
         LR    R8,R1
         IF    N,R1,=XL4'FØØØ',NZ
            N     R8,=F'4Ø95'
            O     R8,=XL4'6ØØØ'
            STH   R8,CODEFLD+2
            SRL   R1,12
            SLL   R1,2
            L     R6,REGTBL(R1)
            LR    R15,R1
            SRL   R15,2
            A     R15,=A(AR_ØØ)
            IF    TM,Ø(R15),AR_B2,O,AND,TM,XCELL+2,X'FØ',NZ
               LAM   R6,R6,Ø(R1)
            ENDIF
         ENDIF
         IF    CLI,XCELL,NE,X'B6',AND,CLI,XCELL,NE,X'B7'  .STCTL,LCTL
            LA    R1,X'2Ø'
            IC    R8,XCELL+1
            N     R8,=XL4'FØ'
            SRL   R8,2
            L     R2,REGTBL(R8)
            L     R3,REGTBL+4(R8)
            SELECT
            WHEN  CLI,XCELL,GE,X'BD',AND,CLI,XCELL,LE,X'BF'
               IC    R15,XCELL+1       .CLM,STCM,ICM?
               N     R15,=F'15'
               OR    R1,R15
            WHEN  CLI,XCELL,EQ,X'BA',OR,CLI,XCELL,EQ,X'BB'
               O     R1,=F'4'          .CS, CDS
               IC    R14,XCELL+1
               N     R14,=F'15'
               SLL   R14,2
               L     R4,REGTBL(R14)
               L     R5,REGTBL+4(R14)
            ENDSEL
            STC   R1,CODEFLD+1
         ENDIF
         RUN_INST
         LAM   R6,R6,=F'Ø'
         IF    CLI,XCELL,NE,X'B6',AND,CLI,XCELL,NE,X'B7'
            ST    R2,REGTBL(R8)
            ST    R3,REGTBL+4(R8)
            IF    CLI,XCELL,EQ,X'BA',OR,CLI,XCELL,EQ,X'BB'
               ST    R4,REGTBL(R14)
               ST    R5,REGTBL+4(R14)
            ENDIF
         ENDIF
         PERF  SHOWINST
         PERF  PRT_RS
         LA    R9,4(,R9)
         MODEXIT
         EJECT
PRT_RS   MODENTRY
         IC    R3,XCELL+1
         PERF  REG_OPS
         SELECT
         WHEN  CLI,XCELL,GE,X'BD',AND,CLI,XCELL,LE,X'BF'  .ICM,STCM,CLM
            PERF  RS_MASK              .DISPLAY BIT MASK
         WHEN  TM,FLAGS,SHIFTBIT,O     .REGISTER SHIFT?
            PERF  RS_SHFT              .DISLPAY SHIFT VALUE
         WHEN  NONE                    .ELSE, STANDARD R-S INSTRUCTION
            IF    CLI,XCELL,EQ,X'B6',OR,CLI,XCELL,EQ,X'B7',OR,         +
               CLI,XCELL,EQ,X'BA',OR,CLI,XCELL,EQ,X'BB'
               MVI   Ø(R6),C','        .CS,CDS,LCTL,STCTL USE 2 REGS
               LA    R6,1(,R6)         .AS WELL AS STORAGE AREA
            ENDIF
            LA    R3,XCELL+2
            PERF  SHOW_BD
            IC    R3,XCELL+1
            IF    CLI,XCELL,NE,X'BB'   .SHOW INVOLVED REGS,
               PERF  SHOW_GRS          .EXCEPT FOR CDS, WHICH HAS 4
            ENDIF                      .REGS, AS WELL AS STORAGE
            SELECT
            WHEN  CLI,XCELL,EQ,X'B6',OR,CLI,XCELL,EQ,X'B7'
               PERF  RS_CTL            .SPECIAL FOR LCTL, STCTL
            WHEN  CLI,XCELL,EQ,X'BB'   .CDS, STG = 2 BYTES
               SHOW_EFA FROM=(XCELL+2),FOR=8,TO=(GR_1-1)
            WHEN  NONE                 .ALL OTHERS, DISPLAY FULLWORD
               SHOW_EFA FROM=(XCELL+2),FOR=4,TO=(EFA2-1)
            ENDSEL
         ENDSEL
         IF    IAC,R15,NZ
            XR    R15,R15
            IC    R15,XCELL
            A     R15,=A(AR_ØØ)
            IF    TM,Ø(R15),AR_B2,O
               MVC   AR_LINE,PRTLINE
               MVC   PRTLINE,=CL133' '
               SHOW_AR FROM=(XCELL+2),TO=(EFA2-5)
               MVC   I_PTR(35),=C'RELATED ACCESS REGS FOR ABOVE INSTR'
            ENDIF
         ENDIF
         MODEXIT
         EJECT
RS_MASK  MODENTRY       .SHOW BITMASK FOR CLM,STCM,ICM
         MVC   Ø(8,R6),=C'B''ØØØØ'','
         SELECT EVERY
         WHEN  TM,XCELL+1,8,O
            MVI   2(R6),C'1'
         WHEN  TM,XCELL+1,4,O
            MVI   3(R6),C'1'
         WHEN  TM,XCELL+1,2,O
            MVI   4(R6),C'1'
         WHEN  TM,XCELL+1,1,O
            MVI   5(R6),C'1'
         ENDSEL
         LA    R6,8(,R6)
         LA    R3,XCELL+2
         PERF  SHOW_BD
         IC    R3,XCELL+1
         PERF  SHOW_GRS
         SHOW_EFA FROM=(XCELL+2),FOR=4,TO=(EFA2-1)
         MODEXIT
         EJECT
RS_SHFT  MODENTRY       .BITSHIFT INSTRUCTIONS
         IF    TM,XCELL+2,X'FØ',NZ      .SHIFT VALUE IN BDDD?
            LA    R3,XCELL+2            .YES, SO GET EFFECTIVE VALUE
            PERF  SHOW_BD               .SHOW OPERAND IN DDDD(B) FMT
            MVC   DR2B(2Ø),=C'ACTUAL SHIFT VALUE: '
            LA    R6,DR2B+2Ø
         ENDIF
         LA    R8,XCELL+2
         PERF  EVALBD
         N     R1,=F'63'                .ONLY LOW-ORDER 6 BITS USED
         CVD   R1,DUB                   .SHOW DECIMAL VALUE
         OI    DUB+7,X'ØF'
         UNPK  Ø(2,R6),DUB+6(2)
         IC    R3,XCELL+1
         PERF  SHOW_GRS
         MODEXIT
         EJECT
RS_CTL   MODENTRY  .STCTL, LCTL - SHOW ONLY STORAGE AREA INVOLVED
         IC    R1,XCELL+1
         LR    R5,R1
         N     R1,=XL4'FØ'
         SRL   R1,4
         N     R5,=F'15'
         IF    CR,R5,LT,R1
            LA    R5,16(,R5)
         ENDIF
         SR    R5,R1
         LA    R5,1(,R5)
         SLL   R5,2
         SHOW_EFA FROM=(XCELL+2),MAX=29,TO=(GR_1-1)
         MODEXIT
         EJECT
EXEC_EXTLONG   MODENTRY  NEWBASE=R1Ø
         MVI   CODEFLD+1,X'24'          .I AM GOING TO USE R2 & R4
         IC    R1,XCELL+1               .RR
         N     R1,=XL4'FØ'              .RØ
         SRL   R1,2                     .RØ*4
         LA    R15,AR_SAVE(R1)
         LA    R1,REGTBL(R1)
         LM    R2,R3,Ø(R1)              .PRIME R2, AND R3 FOR DBL REGS
         IF    IAC,R14,NZ
            XR    R14,R14
            IC    R14,XCELL
            A     R14,=A(AR_ØØ)
            IF    TM,Ø(R14),AR_R1,O
               IF    TM,XCELL+1,X'FØ',Z
                  LAM   R2,R2,=F'Ø'
               ELSE
                  LAM   R2,R2,Ø(R15)
               ENDIF
            ENDIF
         ENDIF
         IC    R1,XCELL+1               .RR
         N     R1,=F'15'                .ØR
         SLL   R1,2                     .*4
         LA    R15,AR_SAVE(R1)
         LA    R1,REGTBL(R1)
         LM    R4,R5,Ø(R1)              .PRIME R4, AND R5 FOR DBL REGS
         IF    IAC,R14,NZ
            XR    R14,R14
            IC    R14,XCELL
            A     R14,=A(AR_ØØ)
            IF    TM,Ø(R14),AR_R2,O
               IF    TM,XCELL+1,X'ØF',Z
                  LAM   R4,R4,=F'Ø'
               ELSE
                  LAM   R4,R4,Ø(R15)
               ENDIF
            ENDIF
         ENDIF
         RUN_INST
         STM   R4,R5,Ø(R1)              .SAVE REGS, IN CASE THEY CHANGED
         LA    R9,4(,R9)                .INCREMENT INSTRUCTION PTR.
         IC    R1,XCELL+1               .RR
         N     R1,=XL4'FØ'              .RØ
         SRL   R1,2                     .R*4
         LA    R1,REGTBL(R1)
         STM   R2,R3,Ø(R1)              .SAVE R1 PAIR
         PERF  SHOWINST
         IC    R3,XCELL+1
         PERF  REG_OPS
         MVI   FIELDS+7,C','
         MVC   FIELDS+8(7),=C'X''ØØØØ'''
         UNPK  FIELDS+1Ø(5),XCELL+2(3)
         MVI   FIELDS+1Ø+4,C''''
         TR    FIELDS+1Ø(4),HEXCHAR-C'Ø'
         PERF  SHOW_GRS
         LA    R8,XCELL+2
         PERF  EVALBD
         STC   R1,DUB
         MVC   DR2B+12(9),=C'PAD=X''ØØ'''
         UNPK  DR2B+12+6(3),DUB(2)
         MVI   DR2B+12+6+2,C''''
         TR    DR2B+12+6(2),HEXCHAR-C'Ø'
         IF    IAC,R14,2
            XR    R15,R15
            IC    R15,XCELL
            A     R15,=A(AR_ØØ)
            IF    TM,Ø(R15),B'ØØ111111',NZ
               PERF  WRITE
               SHOW_AR FROM=(XCELL+1),TO=(GR_1-5)
               IC    R1,XCELL+1
               SLL   R1,4
               STC   R1,DUB
               SHOW_AR FROM=DUB,TO=(GR_2-5)
               MVC   I_PTR(35),=C'RELATED ACCESS REGS FOR ABOVE INSTR'
            ENDIF
         ENDIF
         PERF  WRITE
         XR    R1,R1
         IC    R1,XCELL+1
         SRL   R1,4
         PERF  DISPLAY_LONG
         PERF  WRITE
         IC    R1,XCELL+1
         N     R1,=F'15'
         PERF  DISPLAY_LONG
         MVC   I_PTR(3),=C'OP2'
         MODEXIT
         EJECT
EXEC_SS  MODENTRY  NEWBASE=R1Ø,LIST=YES
* PROCESS DEST FIELD
         LH    R1,CODEFLD+2
         LR    R2,R1
         IF    N,R1,=XL4'FØØØ',NZ
            N     R2,=F'4Ø95'
            O     R2,=XL4'6ØØØ'
            STH   R2,CODEFLD+2
            SRL   R1,12
            SLL   R1,2
            L     R6,AR_SAVE(R1)
            SAR   R6,R6
```

```
               L      R6,REGTBL(R1)
            ENDIF
* PROCESS SOURCE FIELD
            LH    R1,CODEFLD+4
            LR    R2,R1
            IF    N,R1,=XL4'FØØØ',NZ
               N      R2,=F'4Ø95'
               O      R2,=XL4'5ØØØ'
               STH   R2,CODEFLD+4
               SRL   R1,12
               SLL   R1,2
               L      R5,AR_SAVE(R1)
               SAR   R5,R5
               L      R5,REGTBL(R1)
            ENDIF
            SELECT
            WHEN  CLI,XCELL,GE,X'D9',AND,CLI,XCELL,LE,X'DB'
               MVI   CODEFLD+1,X'24'       .MVCK,MVCP,MVCS
               IC    R1,XCELL+1            .USE 2 REGS + 2 ADDR'S
               N     R1,=F'15'
               SLL   R1,2
               L     R4,REGTBL(R1)
               IC    R1,XCELL+1
               N     R1,=XL4'FØ'
               SRL   R1,2
               L     R2,REGTBL(R1)
            WHEN  CLI,XCELL,EQ,X'DD'       .TRT?
               LM    R1,R2,REGTBL+4        .R1 AND R2 ALWAYS USED
            ENDSEL
            IF    IAC,R15,NZ
               XR    R15,R15
               IC    R15,XCELL
               A     R15,=A(AR_ØØ)
               IF    TM,Ø(R15),AR_B1,Z,OR,TM,XCELL+2,X'FØ',Z
                  LAM   R6,R6,=F'Ø'
               ENDIF
               IF    TM,Ø(R15),AR_B2,Z,OR,TM,XCELL+4,X'FØ',Z
                  LAM   R5,R5,=F'Ø'
               ENDIF
            ENDIF
            RUN_INST
            LAM   R5,R6,=2F'Ø'
            IF    CLI,XCELL,EQ,X'DD'
               STM   R1,R2,REGTBL+4
            ENDIF
            PERF  SHOWINST
            LA    R3,XCELL+2
            LA    R6,FIELDS
            PERF  SHOW_BD
            IC    RØ,XCELL+1
            N     RØ,=F'255'
            LR    R1,R6
```

```
         S     R1,=F'4'
         SELECT
         WHEN  CLI,XCELL,GE,X'D9',AND,CLI,XCELL,LE,X'DB'
            PERF  PRT_XMS              .MVCK,MVCP,MVCS
         WHEN  CLI,XCELL,GE,X'FØ'
            PERF  PACKFMT              .PACKED FORMAT
         WHEN  NONE
            PERF  STD_SS               .NORMAL S-S INSTR.
         ENDSEL
         XR    R15,R15
         IC    R15,XCELL
         A     R15,=A(AR_ØØ)
         ST    R15,AR_FLAG
         IF    TM,Ø(R15),AR_B1+AR_B2,NZ,AND,IAC,R15,NZ
            MVC   AR_LINE,PRTLINE
            MVC   PRTLINE,=CL133' '
            L     R15,AR_FLAG
            IF    TM,Ø(R15),AR_B1,O
               SHOW_AR TO=(SS_EFA1-5),FROM=(XCELL+2)
            ENDIF
            L     R15,AR_FLAG
            IF    TM,Ø(R15),AR_B2,O
               SHOW_AR TO=(SS_EFA2-5),FROM=(XCELL+4)
            ENDIF
            MVC   I_PTR(35),=C'RELATED ACCESS REGS FOR ABOVE INSTR'
         ENDIF
         LA    R9,6(,R9)
         MODEXIT
         EJECT
PRT_XMS  MODENTRY
         MVC   4(4,R1),Ø(R1)
         SRL   RØ,4
         CVD   RØ,DUB
         OI    DUB+7,X'ØF'
         UNPK  Ø(3,R1),DUB+6(2)
         MVI   Ø(R1),C'R'
         MVI   3(R1),C','
         MVI   8(R1),C','
         LA    R6,9(,R1)
         LA    R3,XCELL+4
         PERF  SHOW_BD
         IC    RØ,XCELL+1
         N     RØ,=F'15'
         CVD   RØ,DUB
         OI    DUB+7,X'ØF'
         UNPK  1(3,R6),DUB+6(2)
         MVC   Ø(2,R6),=C',R'
         IC    R5,XCELL+1
         N     R5,=XL4'FØ'
         SRL   R5,2
         L     R5,REGTBL(R5)
         IF    CLI,XCELL,EQ,X'DB'      .MVCS?
```

```
            LA    R8,XCELL+2
            PERF  EVALBD
            IC    R3,XCELL+1
            N     R3,=F'15'
            SLL   R3,2
            L     R3,REGTBL(R3)
            MVCP  XMS_WRK(R5),Ø(R1),R3
         ENDIF
         SHOW_EFA FROM=(XCELL+2),MAX=21,TO=(GR_1-1)
         SHOW_EFA FROM=(XCELL+4),FOR=Ø,TO=(SS_EFA3-1)
         PERF  WRITE
         EPAR  R3
         ESAR  R2
         MVC   PRTLINE(29),=C'====>  PASID=XXXX, SASID=XXXX'
         CVD   R3,DUB
         OI    DUB+7,X'ØF'
         UNPK  PRTLINE+13(4),DUB+5(3)
         CVD   R2,DUB
         OI    DUB+7,X'ØF'
         UNPK  PRTLINE+25(4),DUB+5(3)
         MODEXIT
         EJECT
PACKFMT  MODENTRY
         SRL   RØ,4
         A     RØ,=F'1'
         CVD   RØ,DUB
         OI    DUB+7,X'ØF'
         MVC   3(3,R1),Ø(R1)
         MVI   2(R1),C','
         MVC   6(2,R1),=C'),'
         UNPK  Ø(2,R1),DUB+6(2)
         LR    R5,RØ
         LA    R4,8(,R1)
         IF    CLI,XCELL,EQ,X'FØ'
            PERF  PRT_SRP
         ELSE
            SHOW_EFA FROM=(XCELL+2),MAX=9,TO=(SS_EFA1-1)
            LA    R3,XCELL+4
            LR    R6,R4
            PERF  SHOW_BD
            LR    R1,R6
            S     R1,=F'4'
            MVC   3(3,R1),Ø(R1)
            MVI   6(R1),C')'
            MVI   2(R1),C','
            IC    RØ,XCELL+1
            N     RØ,=F'15'
            A     RØ,=F'1'
            CVD   RØ,DUB
            OI    DUB+7,X'ØF'
            UNPK  Ø(2,R1),DUB+6(2)
            LR    R5,RØ
```

```
           SHOW_EFA FROM=(XCELL+4),MAX=9,TO=(SS_EFA2-1)
        ENDIF
        MODEXIT
        EJECT
PRT_SRP MODENTRY
        SHOW_EFA FROM=(XCELL+2),MAX=16,TO=(SS_EFA1-1)
        LR    R6,R4
        IF    TM,XCELL+4,X'FØ',NZ      .SHIFT IN BDDD FMT, B NOT RØ?
           LA    R3,XCELL+4
           PERF  SHOW_BD
           LR    R4,R6
           LA    R6,SS_EFA3+4
           MVC   SS_EFA3-1Ø(14),=C'ACTUAL SHIFT: '
        ENDIF
        LA    R8,XCELL+4
        PERF  EVALBD
        N     R1,=F'63'                .LOW-ORDER 6 BITS USED
        IF    C,R1,GT,=F'31'           .VALUE > 31, NEGATIVE SHIFT
           S     R1,=F'64'             .(IE. RIGHT SHIFT, 6-BIT
           MVC   Ø(3,R6),=C'64-'       .2'S COMPLIMENT)
           LA    R6,3(,R6)
        ENDIF
        CVD   R1,DUB
        OI    DUB+7,X'ØF'
        UNPK  Ø(2,R6),DUB+6(2)
        IF    TM,XCELL+4,X'FØ',NZ
           LR    R6,R4
        ELSE
           LA    R6,2(,R6)
        ENDIF
        MVI   Ø(R6),C','
        IC    R1,XCELL+1
        N     R1,=F'15'
        CVD   R1,DUB
        OI    DUB+7,X'ØF'
        UNPK  1(2,R6),DUB+6(2)
        MODEXIT
        EJECT
STD_SS  MODENTRY
        A     RØ,=F'1'
        CVD   RØ,DUB
        OI    DUB+7,X'ØF'
        MVC   Ø(4,R6),Ø(R1)
        MVI   3(R1),C','
        UNPK  Ø(3,R1),DUB+6(2)
        MVI   8(R1),C','
        LA    R6,9(,R1)
        LA    R3,XCELL+4
        PERF  SHOW_BD
        LR    R5,RØ
        SELECT
        WHEN  CLI,XCELL,EQ,X'DD'
```

```
               LA    R3,X'1Ø'
               PERF  SHOW_GRS
               XR    R5,R5
               IC    R5,XCELL+1
               LA    R5,1(,R5)
               SHOW_EFA FROM=(XCELL+2),MAX=11,TO=GR_2
               SHOW_EFA FROM=(XCELL+4),FOR=Ø,TO=SS_EFA3
            WHEN  CLC,XCELL+2(2),EQ,XCELL+4
               SHOW_EFA FROM=(XCELL+2),MAX=29,TO=(SS_EFA1-1)
            WHEN  CLI,XCELL,EQ,X'D2',OR,CLI,XCELL,EQ,X'BC',OR,            +
                  CLI,XCELL,EQ,X'BD',OR,CLI,XCELL,EQ,X'DC'
               SHOW_EFA FROM=(XCELL+2),MAX=21,TO=(SS_EFA1-1)
               SHOW_EFA FROM=(XCELL+4),FOR=Ø,TO=(SS_EFA3-1)
            WHEN  NONE
               SHOW_EFA FROM=(XCELL+2),MAX=9,TO=(SS_EFA1-1)
               LR    R5,RØ
               SHOW_EFA FROM=(XCELL+4),MAX=9,TO=(SS_EFA2-1)
            ENDSEL
            MODEXIT
            EJECT
            LTORG
            EJECT
***********************************************************************
*              DEFINE FLAG BITS USED THROUGHOUT THE TRACE ROUTINE     *
*              TO SPECIFY THE ATTRIBUTES OF THE MACHINE INSTRUCTIONS  *
***********************************************************************
ILGLBIT  EQU   B'1ØØØØØØØ'             ILLEGAL INSTRUCTION
CCBIT    EQU   B'Ø1ØØØØØØ'             INSTRUCTION SETS CONDITION CODE
BRBIT    EQU   B'ØØ1ØØØØØ'             INSTRUCTION IS BRANCH OR EXECUTE
HALFBIT  EQU   B'ØØØ1ØØØØ'             HALF WORD INSTRUCTION
FULLBIT  EQU   B'ØØØØ1ØØØ'             FULL WORD INSTRUCTION
DBLBIT   EQU   B'ØØØØØ1ØØ'             DOUBLE WORD INSTRUCTION
FLOATBIT EQU   B'ØØØØØØ1Ø'             FLOATING POINT INSTRUCTION
SHIFTBIT EQU   B'ØØØØØØØ1'             SHIFT INSTRUCTION
***********************************************************************
*        ATTRIBUTES WHICH MAY OCCUR IN THE SECOND BYTE                *
*              OF 'FLAGS'                                             *
***********************************************************************
RRBIT    EQU   B'1ØØØØØØØ'             TYPE RR INSTRUCTION
RXBIT    EQU   B'Ø1ØØØØØØ'             TYPE RX INSTRUCTION
RSBIT    EQU   B'ØØ1ØØØØØ'             TYPE RS INSTRUCTION
SIBIT    EQU   B'ØØØ1ØØØØ'             TYPE SI INSTRUCTION
SSBIT    EQU   B'ØØØØ1ØØØ'             TYPE SS INSTRUCTION
ARBIT    EQU   B'ØØØØØ1ØØ'             USES ACCESS REGS
LMSTMBIT EQU   B'ØØØØØØ1Ø'             IMSTRUCTION IS LM OR STM
EXBIT    EQU   B'ØØØØØØØ1'             INSTRUCTION IS EXECUTE (EX)
***********************************************************************
* SPECIAL BIT SWITCHES FOR B2XX EXTENDED OPCODES                      *
***********************************************************************
B2RBIT   EQU   B'1ØØØØØØØ'
B2R2BIT  EQU   B'Ø1ØØØØØØ'
B2R1BIT  EQU   B'ØØ1ØØØØØ'
```

```
B2STGBIT EQU   B'00010000'
B2ADRBIT EQU   B'00001000'
*        EQU   B'00000100'
*        EQU   B'00000010'
B2R0BIT  EQU   B'00000001'
LEFT     EQU   B'100000000'            PLACE IN LEFT HALF FLAGS
RIGHT    EQU   B'00000001'             PLACE IN RIGHT HALF OF FLAGS
         TITLE '*********** AR-FLAGS FOR ALL OP-CODES ***************'
AR_B2    EQU   B'00000001'
AR_B1    EQU   B'00000010'
AR_R2    EQU   B'00000100'
AR_R1    EQU   B'00001000'
AR_UR2   EQU   B'00010000'
AR_UR1   EQU   B'00100000'
*                XXXXXXXX
*                  ||||||
*                  |||||+—— B2 FIELD MAY REFERENCE AR IN AR MODE (1)
*                  ||||+——— B1 FIELD MAY REFERENCE AR IN AR MODE (2)
*                  |||+——— R2 FIELD MAY REFERENCE AR IN AR MODE (4)
*                  ||+———— R1 FIELD MAY REFERENCE AR IN AR MODE (8)
*                  |+———— R2 FIELD WILL ALWAYS REFERENCE AR (10)
*                  +————— R1 FIELD WILL ALWAYS REFERENCE AR (20)
AR_00    DC    X'0,0,0,0,0,0,0,0'  .0,PR/UPT,0,0,SPM,BALR,BCTR,BCR
AR_08    DC    X'0,0,0,0,0,0,C,C'  .0,0,SVC,BSM,BASSM,BASR,MVCL,CLCL
AR_10    DC    X'0,0,0,0,0,0,0,0'  .LPR,LNR,LTR,LCR,NR,CLR,OR,XR
AR_18    DC    X'0,0,0,0,0,0,0,0'  .LR,CR,AR,SR,MR,DR,ALR,SLR
AR_20    DC    X'0,0,0,0,0,0,0,0'  .LPDR,LNDR,LTDR,LCDR,HDR,LRDR,MXR,
*                                  .MXDR
AR_28    DC    X'0,0,0,0,0,0,0,0'  .LDR,CDR,ADR,SDR,MDR,DDR,AWR,SWR
AR_30    DC    X'0,0,0,0,0,0,0,0'  .LPER,LNER,LTER,LCER,HER,LRER,AXR,
*                                  .SXR
AR_38    DC    X'0,0,0,0,0,0,0,0'  .LER,CER,AER,SER,MER,DER,AUR,SUR
AR_40    DC    X'1,0,1,1,0,0,0,0'  .STH,LA,STC,IC,EX,BAL,BCT,BC
AR_48    DC    X'1,1,1,1,1,0,1,1'  .LH,CH,AH,SH,MH,BAS,CVD,CVB
AR_50    DC    X'1,21,0,0,1,1,1,1' .ST,LAE,0,0,N,CL,O,X
AR_58    DC    X'1,1,1,1,1,1,1,1'  .L,C,A,S,M,D,AL,SL
AR_60    DC    X'1,0,0,0,0,0,0,1'  .STD,0,0,0,0,0,0,MXD
AR_68    DC    X'1,1,1,1,1,1,1,1'  .LD,CD,AD,SD,MD,DD,AW,SW
AR_70    DC    X'1,0,0,0,0,0,0,0'  .STE,0,0,0,0,0,0,0
AR_78    DC    X'1,1,1,1,1,1,1,1'  .LE,CE,AE,SE,ME,DE,AU,SU
AR_80    DC    X'1,0,1,0,0,0,0,0'  .SSM,0,LPSW,DIAGNOSE,0,0,BXH,BXLE
AR_88    DC    X'0,0,0,0,0,0,0,0'  .SRL,SLL,SRA,SLA,SRDL,SLDL,SRDA,SLDA
AR_90    DC    X'1,2,2,1,2,2,2,2'  .STM,TM,MVI,TS,NI,CLI,OI,XI
AR_98    DC    X'1,1,D,D,0,0,0,0'  .LM,TRACE,LAM,STAM
AR_A0    DC    X'0,0,0,0,0,0,0,0'
AR_A8    DC    X'0,0,0,0,2,2,0,0'  .0,0,0,0,STNSM,STOSM,SIGP,MC
AR_B0    DC    X'0,1,0,0,0,0,1,1'  .0,LRA,0,0,0,0,STCTL,LCTL
AR_B8    DC    X'0,0,1,1,0,1,1,1'  .0,0,CS,CDS,0,CLM,STCM,ICM
AR_C0    DC    X'0,0,0,0,0,0,0,0'
AR_C8    DC    X'0,0,0,0,0,0,0,0'
AR_D0    DC    X'0,3,3,3,3,3,3,3'  .0,MVN,MVC,MVZ,NC,CLC,OC,XC
AR_D8    DC    X'0,3,0,0,3,3,3,3'  .0,MVCK,MVCP,MVCS,TR,TRT,ED,EDMK
```

```
AR_EØ    DC    X'Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø'
AR_E8    DC    X'3,Ø,Ø,Ø,Ø,Ø,3,Ø'  .MVCIN,
AR_FØ    DC    X'2,3,3,3,Ø,Ø,Ø,Ø'  .SRP,MVO,PACK,UNPK,
AR_F8    DC    X'3,3,3,3,3,3,Ø,Ø'  .ZAP,CP,AP,SP,MP,DP
         TITLE '*********** AR-FLAGS FOR B2  OP-CODES ***************'
AR_B2_ØØ DC    X'Ø,Ø,1,Ø,1,1,1,1'  .Ø,Ø,STIDP,SCK,STCK,SCKC,STCKC
AR_B2_Ø8 DC    X'1,1,Ø,Ø,Ø,Ø,Ø,Ø'  .SPT,STPT,SPKA,IPK,PTLB,
AR_B2_1Ø DC    X'1,1,1,Ø,Ø,Ø,Ø,Ø'  .SPX,STPX,STAP,Ø,SIE
AR_B2_18 DC    X'Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø'  .PC,SAC,CFC,
AR_B2_2Ø DC    X'Ø,Ø,Ø,4,Ø,Ø,Ø,Ø'  .Ø,IPTE,IPM,IVSK,IAC,SSAR,EPAR,ESAR
AR_B2_28 DC    X'Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø'  .PT,ISKE,RRBE,SSKE,TB,DXR,
AR_B2_3Ø DC    X'Ø,Ø,1,1,1,1,1,Ø'  .CSCH,HSCH,MSCH,SSCH,STSCH,TSCH,TPI,
*                                  .SAL
AR_B2_38 DC    X'Ø,1,1,Ø,Ø,Ø,Ø,Ø'  .RSCH,STCRW,STCPS,RCHP,SCHM,
AR_B2_4Ø DC    X'Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø'  .BAKR,Ø,Ø,Ø,Ø,Ø,STURA,MSTA
AR_B2_48 DC    X'Ø,3Ø,Ø,Ø,2Ø,3Ø,2Ø,1Ø' .PALB,EREG,ESTA,LURA,TAR,CPYA
*                                  .SAE,EAR
AR_B2_5Ø DC    X'Ø,Ø,Ø,C,C,Ø,C,Ø'  .Ø,Ø,Ø,MVPG,MVST,Ø,CUSE,Ø
AR_B2_58 DC    X'Ø,Ø,Ø,Ø,Ø,C,4,Ø'  .Ø,Ø,Ø,Ø,Ø,CLST,SRST,Ø
AR_B2_6Ø DC    X'Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø'  .Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
AR_B2_68 DC    X'Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø'  .Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
AR_B2_7Ø DC    X'Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø'  .Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
AR_B2_78 DC    X'Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø'  .Ø,SACF,Ø,Ø,Ø,Ø,Ø,Ø
         TITLE '************** FLAGS FOR ALL OP-CODES ***************'
OPFLAGS  DS    ØH
         DC    4AL2(ILGLBIT*LEFT+RRBIT)          .ØØ-Ø3
         DC    AL2(FULLBIT*LEFT+RRBIT)           .Ø4 (SPM)
         DC    3AL2(BRBIT*LEFT+RRBIT)            .Ø5-Ø7 (BALR,BCTR,BCR)
         DC    2AL2(ILGLBIT+LEFT+RRBIT)          .Ø8-Ø9(SSK,ISK)NOTESA
         DC    AL2(RRBIT+LMSTMBIT)               .ØA (SVC)
         DC    3AL2(BRBIT*LEFT+RRBIT)            .ØB-ØD (BSM-BASR)
         DC    2AL2((CCBIT+DBLBIT)*LEFT+RRBIT)   .ØE-ØF (MVCL,CLCL)
OPF_1Ø   EQU   (*-OPFLAGS)/2
         DC    8AL2((CCBIT+FULLBIT)*LEFT+RRBIT)  .1Ø-17 (LPR-XR)
         DC    AL2(FULLBIT*LEFT+RRBIT)           .18 (LR)
         DC    3AL2((CCBIT+FULLBIT)*LEFT+RRBIT)  .19-1B (CR-SR)
         DC    2AL2(DBLBIT*LEFT+RRBIT)           .1C-1D (MR,DR)
         DC    2AL2((CCBIT+FULLBIT)*LEFT+RRBIT)  .1E-1F (ALR,SLR)
OPF_2Ø   EQU   (*-OPFLAGS)/2
         DC    4AL2((CCBIT+DBLBIT+FLOATBIT)*LEFT+RRBIT)
*                                                .2Ø-23 (LPDR-LCDR)
         DC    2AL2((DBLBIT+FLOATBIT)*LEFT+RRBIT) .24-25 (HDR,LRDR)
         DC    2AL2((DBLBIT+FLOATBIT)*LEFT+RRBIT+EXBIT)
*                                                .26-27 (MXR,MXDR)
         DC    AL2((DBLBIT+FLOATBIT)*LEFT+RRBIT) .28 (LDR)
         DC    3AL2((CCBIT+DBLBIT+FLOATBIT)*LEFT+RRBIT)
*                                                .29-2B (CDR-SDR)
         DC    2AL2((DBLBIT+FLOATBIT)*LEFT+RRBIT) .2C-2D (MDR,DDR)
         DC    2AL2((CCBIT+DBLBIT+FLOATBIT)*LEFT+RRBIT)
*                                                .2E-2F (AWR,SWR)
OPF_3Ø   EQU   (*-OPFLAGS)/2
         DC    4AL2((CCBIT+FULLBIT+FLOATBIT)*LEFT+RRBIT)
```

```
*                                          .3Ø-33 (LPER-LCER)
         DC    2AL2((FULLBIT+FLOATBIT)*LEFT+RRBIT) .34-35 (HER,LRER)
         DC    2AL2((CCBIT+DBLBIT+FLOATBIT)*LEFT+RRBIT+EXBIT)
*                                          .36-37 (AXR,SXR)
         DC    AL2((FULLBIT+FLOATBIT)*LEFT+RRBIT) .38 (LER)
         DC    3AL2((CCBIT+FULLBIT+FLOATBIT)*LEFT+RRBIT)
*                                          .39-3B (CER-SER)
         DC    2AL2((FULLBIT+FLOATBIT)*LEFT+RRBIT) .3C-3D (MER,DER)
         DC    2AL2((CCBIT+FULLBIT+FLOATBIT)*LEFT+RRBIT)
*                                          .3E-3F (AUR,SUR)
OPF_4Ø   EQU   (*-OPFLAGS)/2
         DC    AL2(HALFBIT*LEFT+RXBIT)           .4Ø (STH)
         DC    3AL2(FULLBIT*LEFT+RXBIT)          .41-43 (LA-IC)
         DC    AL2((BRBIT+FULLBIT)*LEFT+RXBIT+EXBIT)
*                                          .44 (EX)
         DC    3AL2(BRBIT*LEFT+RXBIT)            .45-47 (BAL-BC)
         DC    AL2(HALFBIT*LEFT+RXBIT)           .48 (LH)
         DC    3AL2((CCBIT+HALFBIT)*LEFT+RXBIT)  .49-4B (CH-SH)
         DC    AL2(HALFBIT*LEFT+RXBIT)           .4C (MH)
         DC    AL2(BRBIT*LEFT+RXBIT)             .4D (BAS)
         DC    4AL2(FULLBIT*LEFT+RXBIT)          .4E-51 (CVD-LAE)
OPF_52   EQU   (*-OPFLAGS)/2
         DC    2AL2(ILGLBIT*LEFT+RXBIT)          .52-53
         DC    4AL2((CCBIT+FULLBIT)*LEFT+RXBIT)  .54-57 (N-X)
         DC    AL2(FULLBIT*LEFT+RXBIT)           .58 (L)
         DC    3AL2((CCBIT+FULLBIT)*LEFT+RXBIT)  .59-5B (C-S)
         DC    2AL2(DBLBIT*LEFT+RXBIT)           .5C-5D (M,D)
         DC    2AL2((CCBIT+FULLBIT)*LEFT+RXBIT)  .5E-5F (AL,SL)
OPF_6Ø   EQU   (*-OPFLAGS)/2
         DC    AL2((DBLBIT+FLOATBIT)*LEFT+RXBIT) .6Ø (STD)
         DC    6AL2(ILGLBIT*LEFT+RXBIT)          .61-66
         DC    AL2((DBLBIT+FLOATBIT)*LEFT+RXBIT+EXBIT)
*                                          .67 (MXD)
         DC    AL2((DBLBIT+FLOATBIT)*LEFT+RXBIT) .68 (LD)
         DC    3AL2((CCBIT+DBLBIT+FLOATBIT)*LEFT+RXBIT)
*                                          .69-6B (CD-SD)
         DC    2AL2((DBLBIT+FLOATBIT)*LEFT+RXBIT) .6C-6D (MD,DD)
         DC    2AL2((CCBIT+DBLBIT+FLOATBIT)*LEFT+RXBIT)
*                                          .6E-6F (AW,SW)
OPF_7Ø   EQU   (*-OPFLAGS)/2
         DC    AL2((FULLBIT+FLOATBIT)*LEFT+RXBIT) .7Ø (STE)
         DC    1AL2(FULLBIT*LEFT+RXBIT)          .71 (MS)
         DC    6AL2(ILGLBIT*LEFT+RXBIT)          .72-77
         DC    AL2((FULLBIT+FLOATBIT)*LEFT+RXBIT) .78 (LE)
         DC    3AL2((CCBIT+FULLBIT+FLOATBIT)*LEFT+RXBIT)
*                                          .79-7B (CE-SE)
         DC    2AL2((FULLBIT+FLOATBIT)*LEFT+RXBIT) .7C-7D (ME,DE)
```

Editor's note: this article will be continued in the next issue.

Pieter Wiid
Advisory Systems Engineer
Persetel (South Africa)

MVS news

IBM has announced OS/390 Version 2 Release 7, with key focus areas being network support and system management. There is a more streamlined process for adding new TCP/IP stacks and for scaling up TCP/IP resources without system disruption and fewer IP addresses will now be needed within a Parallel Sysplex.

There is also tighter integration between the Tivoli management framework and the SystemView-based S/390 environment. The Tivoli Management Agent has been integrated into Release 7, allowing mainframe facilities such as the Security Server (RACF) to support their Tivoli counterparts, while having a consistent Tivoli view across networks that include S/390s.

Initiatives to help run Unix applications on the S/390 include improvements to the performance of the Unix Hierarchical File System (HFS), and a simplified process of porting Unix applications to the mainframe. A new Open Cryptographic Services Facility supplements the established ICSF and is aimed specifically at OS/390 Unix applications.

E-commerce is supported by the WebSphere Application Server 1.1 which is built in and provides a development and test environment for Java applets and Web server capabilities, including support for strong authentication of digital certificates. The new HTTP Server 5.1, replacing Domino Go Server, also boasts greatly enhanced performance for static Web pages.

Contact your local IBM representative for further information.

* * *

Cybermation has announced the availability of Version 5 Release 1 of its ESP Workload Manager, and the release of ESP Workstation 3.0.

Workload Manager is a job scheduling and workload management tool. The latest version of ESP Workload Manager provides enhancements such as critical path analysis, granular and large application support, enhanced dataset and file triggering capabilities, cashing functionality and event streaming.

The ESP Workstation 3.0 allows users to control their entire distributed enterprise workload, across multiple platforms, from a single point, specifically a Windows 95 workstation, using a graphical application.

It allows real time view and control of workload, and enables centralized, integrated management of distributed workload from any chosen point of control. ESP Workstation 3.0 will be available for Windows 95, 98, and NT.

For further information contact:
Cybermation Canada, 80 Tiverton Court, Markham, Ontario, Canada, L3R OG4.
Tel: (905) 479 4611
Fax: (905) 479 5474 or

Cybermation UK, 2440 The Quadrant, Aztec West, Almondsbury, Bristol, BS32 4AQ, UK.
Tel: 01454 878 745
Fax: 01454 878 651
http://www.cybermation.com

* * *

xephon