194

MVS update

November 2002

In this issue

MVS Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office

Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1999 issue, are available separately to subscribers for £29.00 ($43.50) each including postage.

MVS Update on-line

Code from *MVS Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/mvs; you will need to supply a word from the printed issue.

Editor

Trevor Eddolls

Disclaimer

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

Contributions

When Xephon is given copyright, articles published in *MVS Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

Sending e-mail from MVS

It's not necessary to have an FTP server on MVS just to send e-mail like you would under Windows. You can keep the whole security and safety of MVS and be able to send e-mail automatically from Tivoli OPC, for example each Monday at 7:00pm. To do this, you need XMITIP, MVS freeware written by Lionel B Dyck, available from http://www.lbdsoftware.com.

LDAP stands for Lightweight Directory Access Protocol. As the name suggests, it is a lightweight protocol for accessing directory services, specifically X.500-based directory services. LDAP runs over TCP/IP or other connection-oriented transfer services. The nitty-gritty details of LDAP are defined in RFC2251.

What kind of information can be stored in the directory? The LDAP information model is based on entries. An entry is a collection of attributes that has a globally-unique Distinguished Name (DN). The DN is used to refer to the entry unambiguously. Each of the entry's attributes has a type and one or more values. The types are typically mnemonic strings, like 'cn' for common name, or 'mail' for an e-mail address. The syntax of values depends on the attribute type. For example, a cn attribute might contain the value Babs Jensen. A mail attribute might contain the value babs@example.com. A jpegPhoto attribute would contain a photograph in the JPEG (binary) format.

How is the information arranged? In LDAP, directory entries are arranged in a hierarchical tree-like structure. Traditionally, this structure reflected the geographic and/or organizational boundaries. Entries representing countries appear at the top of the tree. Below them are entries representing states and national organizations. Below them might be entries representing organizational units, people, printers, documents, or just about anything else you can think of.

As soon as Internet e-mail became popular, it was clear we needed a good phone book. Printed directories were obsolete before the ink was dry. Older Internet methods of looking up names, such as whois, Ph, or finger, were limited or arcane. Every e-mail program has a personal

address book, but how do you look up an address for someone who's never sent you e-mail? And how can an organization keep one centralized up-to-date phone book that everybody has access to?

That's why software companies such as Microsoft, IBM, Lotus, and Netscape agreed to support a new standard, called LDAP. LDAP-aware client programs can ask LDAP servers to look up entries in a wide variety of ways. LDAP servers index all the data in their entries, and filters may be used to select just the person or group you want, and return just the information you want. For example, here's an LDAP search translated into plain English: search for all people located in Chicago whose name contains Fred and who have an e-mail address. Please return their full name, e-mail, title, and description.

Permissions are set by the administrator to allow only certain people to access the LDAP database, and optionally keep certain data private. LDAP servers also provide authentication service, so that Web, e-mail, and file-sharing servers (for example) can use a single list of authorized users and passwords.

LDAP was designed at the University of Michigan to adapt a complex enterprise directory system (called X.500) to the modern Internet. A directory server runs on a host computer on the Internet, and various client programs that understand the protocol can log in to the server and look up entries. X.500 is too complex to support on desktops and over the Internet, so LDAP was created to provide this service for the rest of us.

LDAP servers exist at three levels: there are big public servers such as BigFoot and Infospace; organizational servers at universities and corporations; and little LDAP servers for workgroups.

You probably already have an LDAP-aware client installed on your computer. Most current e-mail clients are set up to search an LDAP directory for e-mail addresses. These include Outlook, Eudora, Netscape Communicator, QuickMail Pro, and Mulberry (but not Emailer, sorry).

LDAP has broader applications, such as looking up services and devices on the Internet (and intranets). Netscape Communicator (4.5 and later) can store user preferences and bookmarks on an LDAP server.

There is even a plan for linking all LDAP servers into a worldwide hierarchy, all searchable from your client.

LDAP promises to save users and administrators time and frustration, making it easy for everyone to connect with people without frustrating searches for email addresses and other trivia.

Before use, the XMITLDAP REXX EXEC needs to be edited thus:

- *ldap_server* – host name of your LDAP server, or 0 (zero) to disable completely.
- *ldap_o* – organization and country for LDAP queries.
- *local_nodes* – domain names from e-mail addresses that will be checked. All others will be ignored as they are probably external and not in your LDAP mail directory.

Examples:

```
ldap_server = mailhub.kp.org or ldap_server = Ø /* to disable */
ldap_o      =  o=Kaiser Permanente,c=US
local_nodes = kp.org ncal.kaiperm.org notes.kp.org
```

Other set-up is required.

In XMITIPCU:

```
 /* ————————————————————— *
    * Allow E-Mail address (ID) validation in batch         *
    * Ø = allow    1 = do NOT allow                         *
    * ————————————————————— */
    batch_idval = Ø
```

Below is my TCP/IP SMTP PROC:

```
//SMTP PROC MODULE=SMTP,DEBUG=,PARMS='/',SYSERR=SYSERR
//SETSMSG EXEC PGM=SETSMSG,PARM=ON
//SYSPRINT DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//SMTP EXEC PGM=MVPMAIN,REGION=6144K,TIME=144Ø,
//            PARM='&MODULE,PARM=&DEBUG,ERRFILE(&SYSERR),&PARMS'
//STEPLIB  DD DSN=SYS1.SEZATCP,DISP=SHR
//*SMTPNJE  DD DSN=TCPIP.SMTPNJE.HOSTINFO,DISP=SHR
//CONFIG   DD DSN=&SYSNAME..TCPIP.PARMLIB(SMTP&SYSNAME.),DISP=SHR
//*SECTABLE DD DSN=SMTP.SMTP.SECTABLE,DISP=SHR
//SMTPRULE DD DSN=TCPIP.SMTP.RULES,DISP=SHR
```

```
//* UTILISATION D'UN FICHIER TCPIP.DATA SPECIFIQUE
//* POUR EVITER LES RECHERCHES DE NOMS ET
//* FORCER LE ROUTAGE VERS BERCY
//SYSTCPD  DD DISP=SHR,DSN=TCPIP.SMTP.TCPIP.DATA
//*SYSMDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//SYSDEBUG DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//LOGFILE  DD SYSOUT=*
```

XMITB64 is an Assembler program used by XMITIP to encode binary data into base64 (aka MIME format) so that it can be sent as an attachment to an SMTP mail.

The program was developed by Mark Feldman and shared with the XMITIP author.

If you want your OS/390 SMTP Server to send all mail to an existing mail server within your enterprise do the following:

1 Find the dataset pointed to by the SYSTCPD DD in the TCPIP started task JCL and make a copy for use in the SMTP started task JCL.

2 In that copy, comment out the NSINTERADDR statements. These statements are used to define the domain name servers that will be used to resolve host names.

3 In the SMTP CONFIG data add the IPMAILERADDRESS statement. The syntax is IPMAILERADDRESS xx.xx.xx.xx where the xx.xx.xx.xx is the numeric IP address of the target mail server.

SMTP will now forward all mail for which it is unable to resolve the target host name to the server defined in the IPMAILERADDRESS.

Here is my job to send the MVS file SRS.CIC5.ETAT99.SYSOUT to user.windows@aol.com

```
//SENDMAIL EXEC PROC=BATCHMEL
//DDFILE   DD DISP=SHR,DSN=SRS.CIC5.ETAT99.SYSOUT
//DDMSG    DD *
Here is my file
/*
//SYSIN    DD *
%xmitip user.windows@aol.com -
```

```
   cc ( user2.windows@aol.com -
) -
   replyto user1.windows@aol.com -
   from tsouseramvs-dsn-name -
   subject 'Disponibilite Teletraitement' -
   msgdd ddmsg -
   filedd ddfile
```

Claude Dunand
Systems Programmer (France)

Displaying the contents of a register

Several years ago I wrote an Assembler macro that displayed the contents of all the registers. That macro was published in *MVS Update*, Issue 120. However, in practice, one hardly needs to see all the registers. Normally, we want to know only the contents of one or two, to solve a particular problem or to debug our program. So I decided to write a lighter version to display only one register. The advantage is that this new version occupies much less space, and thus it is better suited to being inserted several times within a program, in strategic positions. This new macro is called SHOWREG, and has two arguments.

The first is the register we want to observe. It can be specified either as a single number (SHOWREG 5) or, as most people will be used to writing, with an R before the number (SHOWREG R5).

The second argument is optional and represents the DDname of the output destination. If it is omitted, the result is written to TSO by means of a TPUT macro invocation. In this case, SHOWREG adds 128 bytes of code to your program. If a DDname is used, then I create a DCB, open it, 'PUT' the answer, and close it again. The overhead, in this case, is 246 bytes, so my suggestion is use this option only for batch executions. In this case, the simplest way is to use SYSPRINT as a DDname, or create a similar one and declare it in the JCL. In any case, the output is 12 bytes long and looks as follows:

```
R5  ØØ34FCE2
```

THE CODE

```
*===================================================================
* SHOWREG - This Assembler macro displays the contents of a single
*           register in hexadecimal.
* Format: SHOWREG register,outddname  where register is the number of
* the register and outddname is the DDname of the output destination.
* If outddname is blank, the output is written to a terminal by means
* of TPUT, otherwise a DCB is created for the ddname, followed by
* an OPEN, PUT, and CLOSE. This last method should be used for batch
* executions.
* Generated code length: 128 bytes without DDname, 246 with DDname.
*===================================================================
         MACRO
         SHOWREG &PØ,&DDOUT        Parms: register,outddname
&A       SETA  &SYSNDX             Index to ensure labels are unique
         B     XTR&A               Branch around working areas
         DS    ØF
XSTORE&A DS    3F                  Register store area (R15 RØ R1)
XTR1&A   DC    X'ØFØFØFØFØFØFØFØF' Formatters for hex display
XTR2&A   DC    C'Ø123456789ABCDEF'
XREG5&A  DS    ØCL5
XREG&A   DS    F                   Register to process
         DS    CL1
XUNP&A   DS    CL8                 Unpacked register
         DS    CL1
XOUTLB&A DC    CL1'R'              Output fields
XOUTL1&A DC    CL1' '
XOUTL2&A DC    CL2'  '
XOUT9&A  DS    ØCL9
XOUT8&A  DC    CL8'        '
         DC    CL1' '
*===================================================================
XTR&A    STM   R15,R1,XSTORE&A     Store regs affected by called
*                                  macros (R15,RØ,R1)
         AIF ('&PØ' EQ '').XEND    If no reg specified, exit
         AIF ('&PØ' NE 'Ø' AND '&PØ' NE 'RØ').XR1
         MVI   XOUTL1&A,C'Ø'
         ST    Ø,XREG&A
.XR1     AIF ('&PØ' NE '1' AND '&PØ' NE 'R1').XR2
         MVI   XOUTL1&A,C'1'
         ST    1,XREG&A
.XR2     AIF ('&PØ' NE '2' AND '&PØ' NE 'R2').XR3
         MVI   XOUTL1&A,C'2'
         ST    2,XREG&A
.XR3     AIF ('&PØ' NE '3' AND '&PØ' NE 'R3').XR4
         MVI   XOUTL1&A,C'3'
         ST    3,XREG&A
.XR4     AIF ('&PØ' NE '4' AND '&PØ' NE 'R4').XR5
         MVI   XOUTL1&A,C'4'
```

```
         ST    4,XREG&A
.XR5     AIF ('&PØ' NE '5' AND '&PØ' NE 'R5').XR6
         MVI   XOUTL1&A,C'5'
         ST    5,XREG&A
.XR6     AIF ('&PØ' NE '6' AND '&PØ' NE 'R6').XR7
         MVI   XOUTL1&A,C'6'
         ST    6,XREG&A
.XR7     AIF ('&PØ' NE '7' AND '&PØ' NE 'R7').XR8
         MVI   XOUTL1&A,C'7'
         ST    7,XREG&A
.XR8     AIF ('&PØ' NE '8' AND '&PØ' NE 'R8').XR9
         MVI   XOUTL1&A,C'8'
         ST    8,XREG&A
.XR9     AIF ('&PØ' NE '9' AND '&PØ' NE 'R9').XR1Ø
         MVI   XOUTL1&A,C'9'
         ST    9,XREG&A*
.XR1Ø    AIF ('&PØ' NE '1Ø'AND '&PØ' NE 'R1Ø').XR11
         MVI   XOUTL1&A,C'1'
         MVI   XOUTL2&A,C'Ø'
         ST    1Ø,XREG&A
.XR11    AIF ('&PØ' NE '11'AND '&PØ' NE 'R11').XR12
         MVI   XOUTL1&A,C'1'
         MVI   XOUTL2&A,C'1'
         ST    11,XREG&A
.XR12    AIF ('&PØ' NE '12'AND '&PØ' NE 'R12').XR13
         MVI   XOUTL1&A,C'1'
         MVI   XOUTL2&A,C'2'
         ST    12,XREG&A
.XR13    AIF ('&PØ' NE '13'AND '&PØ' NE 'R13').XR14
         MVI   XOUTL1&A,C'1'
         MVI   XOUTL2&A,C'3'
         ST    13,XREG&A
.XR14    AIF ('&PØ' NE '14'AND '&PØ' NE 'R14').XR15
         MVI   XOUTL1&A,C'1'
         MVI   XOUTL2&A,C'4'
         ST    14,XREG&A
.XR15    AIF ('&PØ' NE '15'AND '&PØ' NE 'R15').XUNPAC
         MVI   XOUTL1&A,C'1'
         MVI   XOUTL2&A,C'5'
         ST    15,XREG&A
.XUNPAC  UNPK  XOUT9&A,XREG5&A     Unpack reg
         NC    XOUT8&A,XTR1&A      Format it for display
         TR    XOUT8&A,XTR2&A
*=================================================================
* Choose output type and jump accordingly
*=================================================================
         AIF ('&DDOUT' NE '').XPRINT
         TPUT  XOUTLB&A,12
         AGO   .XEND
.XPRINT  ANOP
```

```
         B     XOPEN&A
PRINT&A  DCB   DSORG=PS,                                          X
               MACRF=(PM),                                       X
               DDNAME=&DDOUT
XOPEN&A  OPEN  (PRINT&A,OUTPUT)
         PUT   PRINT&A,XOUTLB&A,12
         CLOSE PRINT&A
.XEND    ANOP
XEXIT&A  LM    R15,R1,XSTORE&A       Restore saved regs
*=================================================================
* SHOWREG MACRO END
*=================================================================
         MEND
```


Dynamically loading an external program from a CPP program

PROBLEM ADDRESSED

IBM is increasingly pushing C, and CPP, as a mainframe programming language. However, other than DLLs (Dynamic Link Libraries), the CPP language does not provide any means of dynamically loading an external program. This contrasts with the C language, which provides the fetch() function.

This forced early (static) binding of a load module has several disadvantages, in particular when legacy routines need to be invoked. It is often neither practicable nor possible to statically include such external programs, for example programs specified in the link pack. The CLINK function described in this article solves the problem of dynamically loading legacy programs or subprograms.

SOLUTION

The CLINK function loads the specified program from the assigned load library (JOBLIB or STEPLIB in the batch environment). To

improve performance for subsequent calls, the entry point of the loaded module is returned to the caller. If on subsequent calls this address is non-zero, CLINK directly invokes the previously loaded module. If this entry-point address was initialized appropriately (for example, with the address of a service routine obtained from the CVT), the module will not be loaded the first time.

The third and any subsequent arguments passed to CLINK are passed to the invoked module using standard MVS calling conventions. This means a module invoked as a main program can receive only a single parameter prefixed with a halfword that contains the length of the following data (JCL convention) – see SAMPLE PROGRAM 2; SAMPLE PROGRAM 1 shows a subroutine invoked dynamically.

Note: although the CPP programs are used as examples, C programs can also use CLINK() in place of fetch().

The CLINK calling sequence is:

```
int rc CLINK(const char pgmname[9], volatile long *apgm, ...);
```

where:

- *pgmname* – the name of the program to be loaded (left-justified, right-padded with blanks) and executed.
- *apgm* – the address of the loaded program (must be initialized to zero or the preloaded address of the previously loaded program). This address remains as 0 if the program could not be loaded.

  CLINK sets this address and reuses it for subsequent calls to avoid a reload. A reload can be forced by resetting this address to 0.
- *rc* – return code from the called program or the negated return code from the LOAD service macro if the program could not be loaded.

The third and any subsequent parameters are passed to the called program.

CLINK PROGRAM CODE

```
        TITLE 'CLINK - Dynamic Link to C (C++) Program'
        PRINT NOGEN
```

```
* Load a named module into storage and execute.
* Calling sequence:
*  int rc CLINK(const char pgmname[9],volatile long *apgm,...);
*  <pgmname>: Name of program to be loaded (left-justified,
*          right-padded with blanks) and executed.
*  <apgm>: Address of the loaded program (must be initialized to zero).
*          CLINK sets this address and reuses it for subsequent
*          calls to avoid a reload. A reload can be forced by
*          resetting this address to Ø.
*  <rc>: Return code from the executed program.
CLINK    CSECT
CLINK    AMODE 31
CLINK    RMODE ANY
         EDCPRLG USRDSAL=8,BASEREG=9
         USING WKDSECT,R13
* Entry conditions:
*  R1: A(program name pointer)
         SPACE 1
         LA    R4,8(R1)               user parameters
         L     R2,Ø(R1)               A(program name)
         MVC   PROGNAME,Ø(R2)         store program name
         L     R3,4(R1)               A(loaded program address)
         ICM   R15,15,Ø(R3)           test passed address
         JNZ   LOADED                 already loaded
* initialise BLDL
         MVC   BLDLNO,=H'1'           one entry
         MVC   BLDLLEN,=AL2(ELEN)
         LA    RØ,BLDLLIST
         BLDL  Ø,(RØ)
         LNR   R15,R15                negate
         JNZ   EOP                    program could not be loaded
         LA    R2,PROGNAME
         LOAD  DE=(R2)                load program
         LR    R15,RØ                 entry point address
         ST    R15,Ø(R3)              return entry-point address
LOADED   LR    R1,R4                  pass pointer to user parameters
         BASSM R14,R15                invoke loaded program
EOP      EDCEPIL ,                    terminate
         SPACE
WKDSECT  EDCDSAD ,                    dynamic save (work) area
* BLDL entry
BLDLLIST DS    ØH
BLDLNO   DS    H
BLDLLEN  DS    H
PROGNAME DS    CL8
TTR      DS    XL3
KZC      DS    3XL1
BLDLUSER DS    CL44
BLDLEND  EQU   *
```

```
ELEN     EQU   BLDLEND-PROGNAME        entry length
         SPACE
* register equates
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END
```

CLINK SAMPLE PROGRAM 1

This sample program invokes the external GETDSN subroutine. This subroutine requires two parameters:

1 An 8-character field containing the DDname (nine characters including the 0-delimiter).

2 A 44-character field that will be returned with the associated DSname (45 characters including the 0-delimiter).

```
#include <strstream.h>

extern "C" int CLINK(const char *, volatile long *, \
               const char *, char *);

int main()
{
  int rc;
  char pgmname[9] = "GETDSN  ";
  char ddname[9] = "ISPPROF ";
  char dsname[45];
  long ptr = Ø; // address of the loaded subprogram

  rc =  CLINK(pgmname,&ptr,ddname,dsname);
  if (ptr == Ø) {
    cout << "program " << pgmname << " could not be loaded" << endl;
    return 8;
  }
  dsname[44] = Øx ØØ; // append string delimiter
  cout << "RC:" << rc << " DSN:" << dsname << endl;
  return rc;
}
```

CLINK SAMPLE PROGRAM 2

This sample program invokes an external main program, here IEWL (the program name of the LinkageEditor or Binder). The parameter passed to a program must conform to JCL conventions, ie prefixed with a halfword that contains the length of the following data.

```
#include <strstream.h>
#include <string.h>

struct EXECPARM {
  short parmlen;
  char  parmdata[1Ø1];
};

extern "C" int CLINK(const char*, volatile long*, \
               struct EXECPARM*);

int main()
{
  int rc;
  char pgmname[9] = "IEWL    ";
  long ptr = Ø; // address of the loaded program
  struct EXECPARM execparm;

  /* initialise EXEC parameter for Linkage Editor */
  strcpy(execparm.parmdata,"XREF,NOMAP,AMODE(31)");
  execparm.parmlen = strlen(execparm.parmdata);
  rc =  CLINK(pgmname,&ptr,&execparm);
  if (ptr == Ø) {
    cout << "program " << pgmname << " could not be loaded" << endl;
    return 8;
  }
  return rc;
}
```


Who owns a particular dataset

INTRODUCTION

When you need to see who owns (or wants to own) a particular dataset, you have the option of entering an MVS command (D

GRS,RES=(SYSDSN,dsname) or, if it is a partitioned dataset, you can try to compress it and then press PF1 twice to see owners/requestors. I have written a simple REXX program to do this job. It issues the above MVS command and displays the result in an ISPF view session. It is very handy!

INSTALLATION

To install the GRS command follow the steps given below:

1 Send the code to your mainframe (ASCII mode in FTP or ASCII and CR/LF in Personal Communications file transfer).

2 Store it as a member named GRS in a library in your SYSPROC or SYSEXEC concatenation.

USAGE

GRS is a TSO command. It must be placed in a library in the SYSEXEC or SYSPROC concatenation. To use it either:

1 In any TSO command line enter:

```
TSO GRS 'data_set_name'
```

2 Display a list of datasets with ISPF Option 3.4 and the type GRS at the left of the dataset name you want to check.

As a result you will get a view of temporary datasets showing requestors (if there are any) for the dataset given as the argument to GRS.

Example:

```
DSLIST - Data Sets Matching MARCIN.JCL                    Row 1 of 8
Command ===>                                           Scroll ===> CSR

Command - Enter "/" to select action          Message          Volume
-------------------------------------------------------------------------
         MARCIN.JCL                                             USR002
         MARCIN.JCL.CICS                                        USR001
         MARCIN.JCL.COMPILE                                     USR004
         MARCIN.JCL.MQS                                         USR001
         MARCIN.JCL.PROCLIB                                     USR003
         MARCIN.JCL.RACF                                        USR001
```

```
GRS      MARCIN.JCL.REXX                                          USR004
         MARCIN.JCL.VSAM                                          USR001
************************ End of Data Set list ************************
```

Result (in cases where there are no requestors):

```
VIEW       SYS02109.T160723.RA000.MARCIN.R0132549    Columns 00001 00072
Command ===>                                            Scroll ===> CSR
****** ************************ Top of Data ***************************
000001  ISG343I 16.07.22 GRS STATUS 988
000002  NO REQUESTORS FOR RESOURCE  SYSDSN   MARCIN.JCL.REXX
****** ************************ Bottom of Data ************************
```

Result (if there are any requestors):

```
VIEW       SYS02109.T161321.RA000.MARCIN.R0132565    Columns 00001 00072
Command ===>                                            Scroll ===> CSR
000001  ISG343I 16.13.21 GRS STATUS 171
000002  S=SYSTEM  SYSDSN   USER.MACRO
000003  SYSNAME     JOBNAME       ASID       TCBADDR   EXC/SHR     STATUS
000004  SERAT2    SPMKE           0019       008FDE48   SHARE       OWN
000005  SERAT2    AGA             0082       008FDE48   SHARE       OWN
000006  SERAT2    MALECEK         006A       008FDE48   SHARE       OWN
000007  SERAT2    PIOTRH          0085       008FDE48   SHARE       OWN
000008  SERAT2    MARCIN          0076       008FDE48   SHARE       OWN
000009  SERAT2    MAREK           0094       008FDE48   SHARE       OWN
000010  SERAT2    NIEROB          0010       008E7A70 EXCLUSIVE    WAIT
****** ************************ Bottom of Data ************************
```

You can see jobs which own (or would like to own) the dataset in question either in share or in exclusive mode.

After exiting from the view (press PF3 or PF12 to do so) the temporary dataset will be deleted.

PROGRAMMING-RELATED INFORMATION

The GRS command has been written in REXX. To retrieve the information about requestors for the given dataset, it uses the MVS command D GRS. This is done via the TSO CONSOLE command, then the GETMSG function is used to retrieve the results. To display the result, I use a standard ISPEXEC service, VIEW.

The program logic is:

1 Accept the argument.

2 Issue the MVS command.

3 Get the result.

4 Save it in a temporary dataset.

5 Display it.

6 Clean up and exit.

REXX

```
/* REXX ********************************** by Marcin Grabinski, SPIN */
/*                                                                    */
/* display GRS info for a given DSN                                   */
/*                                                                    */
ARG dsn
ADDRESS TSO
name = 'GRS'TIME('s')
cmd = '"CONSOLE ACTIVATE NAME('name')"'
INTERPRET cmd
"CONSPROF SOLDISPLAY(NO) SOLNUM(200)"
cart = TIME('s')
cmd = '"CONSOLE SYSCMD(D GRS,RES=(SYSDSN,'dsn')) CART('"'"cart"'"')"'
INTERPRET cmd
rcode = GETMSG('msg.','SOL',cart,,5)
DO i = 1 TO msg.0
  QUEUE msg.i
END
"CONSOLE DEACTIVATE"
QUEUE ''
l = SYSVAR('SYSWTERM')
'ALLOCATE DDN(GRSTEMP) NEW REUSE LRECL('l')'
'EXECIO * DISKW GRSTEMP (FINIS'
ADDRESS ISPEXEC
'ISPEXEC LMINIT DATAID(ID) DDNAME(GRSTEMP)'
'ISPEXEC VIEW DATAID('id')'
ADDRESS TSO
'FREE DDN(GRSTEMP)'
RETURN
```

Marcin Grabinski
System Engineer
SPIN (Poland)

Simple COBOL (batch) debug tool

INTRODUCTION

Some years ago, when I was working in the VM/CMS world, I was approached by one of our programmers with a problem. He had recently taken over the responsibility for a suite of COBOL programs, didn't have a clue about their structure, and was expected to maintain them. He thought it would be a great idea to insert a DISPLAY statement after each paragraph and section and let the programs run – that way he would be able to see at least the program flow and thereby increase his understanding. This great idea came with three big problems: the first was the incredible amount of work he must invest to insert all these DISPLAYs; the second was that for a couple of the programs the output was too vast; and the third was that he needed to return the programs to their original state without the DISPLAY statements, undoing all his previous work, before implementation in the production environment. He came to me and asked if it was possible to automate the process. I wrote at first a very simple routine, which was further developed until it filled the extended needs of our programmer, while at the same time retaining its simplicity. The routine generates COBOL DISPLAY statements at stategic points to allow program flow debugging of a COBOL batch program.

Recently I have had the chance to use my new-found knowledge of ISPF together with edit macros to rewrite the routine for use with TSO.

The REXX part remained pretty much the same, with the same logic; however, it needed to be adapted to use the TSO as opposed to the CMS edit commands and call ISPF instead of XEDIT for the user interface.

One major difference within TSO is that the EXEC must be declared as an edit macro; this is achieved with the ISREDIT MACRO statement.

As is the normal case with addressing within TSO, one can decide either to address globally or per command. I have decided to address the edit macro commands globally with ADDRESS ISREDIT and to address the ISPF commands locally, (eg ADDRESS ISPEXEC "ADDPOP COLUMN(10) ROW(5)"), ie per command.

FUNCTION

The original version was a very simple routine. It searched for the PROCEDURE DIVISION statement and then inserted, after every section or paragraph that followed, a DISPLAY statement with the name of the section or paragraph contained within. The next phase was to build in the function to extract (delete) all the inserted DISPLAY statements and return the code to its original state. To allow the non-display of very repetitive areas of code (reduce output) or to concentrate on a specific part, a range control was built into the routine. After testing on programs, it was quickly discovered that the destination (paragraph/ section) was clear to see, but the originating (calling) position could be one of several. To help with this problem two further functions were incorporated – to display from within a GO TO and/or from within a non-inline PERFORM statement. To differentiate one from another, a counter was also included in these DISPLAYS. One further function to increase readabilty after the implementation of XDP is the option to 'exclude' XDP comment lines.

The default control range is from the first line after the PROCEDURE DIVISION statement to the last line of code. The control range starting point may be increased and the ending point may be decreased to concentrate on a specific area for the insertion of DISPLAY statements and equally well for the deletion of DISPLAY statements when the extract (delete) function is used.

The standard use is to select one of the options S, G, or P (or all together with option A) to insert DISPLAYs, and option X without changing the range to delete all previously inserted DISPLAYs. There is, however, a huge amount of flexibility within the routine which allows XDP to be called time and time again to insert more than one specific range of lines of code or also to delete just a specific range of XDP DISPLAY statements. At the end, all the inserted lines may then be effortlessly extracted with the X option.

Although the routine is quite simple, I have included a couple of things to make it more user friendly. These include various Help screens (general and field specific), input field testing, and error messages. To show that XDP has been successfully used (or not) a final display screen is displayed to show the statistics for the DISPLAY lines inserted or, in the case of option X, deleted.

The XDP routine is invoked when the programmer is in a COBOL edit (ISPF) session; they just need to enter XDP on the command line and press the *Enter* key.

The following are the files (members), names, and locations:

- Edit macro – REXX (SYSEXEC, hlq.EXEC) – XDP.
- ISPF PANELS (ISPPLIB, hlq.PANELS) – XDP (main panel); XDPA (info panel); XDP01H (help panel); XDP01H1 (help panel); XDP01H2 (help panel); XDP01H3 (help panel); XDP01H4 (help panel); XDP01H5 (help panel).
- ISPF messages (ISPMLIB, hlq.MSGS) – XDP01 (Info/Warning/Error messages).

XDP EXEC

```
/* REXX */                     /* required REXX identifier           */
/* Procedure to insert strategic displays to aid debugging          */
/* ***************************************************************** */
'ISREDIT MACRO'                /* required EDIT MACRO identifier     */
ADDRESS ISREDIT                /* set MODE to ISREDIT                */
trace o                        /* trace switch                       */
'(bnd1,bnd2) = BOUNDS'

optcntl =' '              /* set default option Sections/Paragraphs */
exccntl ='/'              /* set exclude option as default          */
'SEEK "PROCEDURE DIVISION" FIRST'   /* establish start of code      */
do forever
  '(linecnt) = LINENUM .ZCSR'           /* find current line number */
  '(lcontent) = LINE (linecnt)'         /* extract line data        */
  comment  =  substr(lcontent,7,1)      /* check column 7 (comment) */
  if comment ='' then leave
 'SEEK "PROCEDURE DIVISION" NEXT'
end
'(fromcntl) = LINENUM .ZCSR'        /* set default start position    */
'(tocntl) = LINENUM .ZLAST'         /* set default end position      */

                         /*  set the ranges                          */
fromcntl = fromcntl + 1  /* first position after PROCEDURE DIVISION */
fromcntl = strip(fromcntl,'L',Ø)    /* strip leading zeroes          */
tocntl = strip(tocntl,'L',Ø)        /*             ""                */
lowerlim = fromcntl      /* PROCEDURE DIVISION start = minimum      */
upperlim = tocntl        /* PROCEDURE DIVISION end   = maximum      */
                         /*                                         */
/* ***************************************************************** */
```

```
ADDRESS ISPEXEC "ADDPOP COLUMN(10) ROW(5)"    /* add pop-up window   */
ADDRESS ISPEXEC "DISPLAY PANEL(XDP)"     /* open the control screen  */
if rc = 8 then return     /* PF3 PF4 or CANCEL - return to EDIT sess */
ADDRESS ISPEXEC "REMPOP"
/* ************************************************************** */

/* XDP comment lines used to nest the DISPLAY statement              */
topline ='      * -------- .XDPS * START * ---------- DO NOT CHANGE ----
----'
botline ='      * -------- .XDPE * END   * ---------- DO NOT CHANGE ----
----'
parxx = 0                              /* initialize counters      */
secxx = 0                             /*               ""           */
gotoxx = 0                            /*               ""           */
perfxx = 0                            /*               ""           */
delxx = 0                             /*               ""           */

select               /* select the routine per control option        */
  when optcntl ='S' then do        /* Sections/Paragraphs           */
    call snp_dis
  end
  when optcntl ='X' then do        /* eXtract/remove XDPs           */
    call X_tract
  end
  when optcntl ='G' then do        /* GOTO structures               */
    call goto_dis
  end
  when optcntl ='P' then do        /* Perform structures            */
    call perf_dis
  end
  when optcntl ='A' then do        /* Options S,P and G together    */
    call snp_dis
    call perf_dis
    call goto_dis
  end
  otherwise
    say'control not recognised'    /* not S, X, G, A or P           */
end
if exccntl ='/' then do       /* exclude XDP comment lines          */
 'EXCLUDE ".XDPS" ALL'
 'EXCLUDE ".XDPE" ALL'
end
'CURSOR = (fromcntl)'     /* set CURSOR to start position            */
'DOWN CURSOR'             /* scroll to cursor                        */
parxx = strip(parxx,'L',0)          /* strip leading zeroes          */
secxx = strip(secxx,'L',0)          /*               ""              */
gotoxx = strip(gotoxx,'L',0)        /*               ""              */
perfxx = strip(perfxx,'L',0)        /*               ""              */
delxx = strip(delxx,'L',0)          /*               ""              */
/* ************************************************************** */
```

```
ADDRESS ISPEXEC "ADDPOP COLUMN(7) ROW(2)"     /* add pop-up window   */
ADDRESS ISPEXEC "DISPLAY PANEL(XDPA)"    /* open final message screen*/
ADDRESS ISPEXEC "REMPOP"
/* ************************************************************** */
exit

/* ************************************************************** */
snp_dis:             /* Section/Paragraphs routine                */
                     /* Insert a DISPLAY statement after each     */
                     /* Section or Paragraph within the given range  */
                     /*                                           */
'CURSOR = (fromcntl)'     /* set CURSOR to start position         */
do forever
  '(linecnt) = LINENUM .ZCSR'            /* find current line number */
   if linecnt >= tocntl then leave       /* exit when end reached    */
  '(lcontent) = LINE (linecnt) '         /* extract line data        */
   acontent =  substr(lcontent,8,4)      /* check the A Margin       */
   comment  =  substr(lcontent,7,1)      /* check column 7 (comment) */
   if acontent ¬='    ' & comment ¬='*' then do
     call di_para                        /* insert display block     */
   end
   else do                               /* not Section/Para header  */
     '(lineno,colno) = CURSOR'           /* move cursor to next line */
     lineno = lineno + 1                 /*        ""                */
     'CURSOR = (lineno)'                 /*        ""                */
   end
end
return

/* ************************************************************** */
di_para:             /* Section/Paragraphs display block insertion   */

w1         = Ø
wo         =' '

'(lcontent) = LINE (linecnt)'       /* extract line contents         */
w1 = find(lcontent,"SECTION.")

if w1 ¬= Ø then               /* check if SECTION or PARAGRAPH      */
  do
     displn1 ='           DISPLAY "SEC ==>'
     secxx = secxx + 1
  end
  else do
     displn1 ='           DISPLAY "PAR ==>'
     parxx = parxx + 1
  end

rcontent =  substr(lcontent,8,72)   /* read the whole non-comment    */
wo = subword(rcontent,1,1)          /* complete DISPLAY statement    */
```

```
displn2 = ' ".'                          /*       ""               */
displn = insert(wo,displn1,28)           /*       ""               */
displn = insert('"',displn,68,1,' ')     /*       ""               */
displn = insert('.',displn,7Ø,1,' ')     /*       ""               */

'(lineno,colno) = CURSOR'                /* current cursor position  */
'LINE_AFTER (lineno) = DATALINE (topline)'    /* insert comment line */
tocntl = tocntl + 1                      /* increment last line      */
'(lineno,colno) = CURSOR'                /* increment the cursor     */
lineno = lineno + 1                      /*       ""               */
'CURSOR = (lineno)'                      /*       ""               */

'(lineno,colno) = CURSOR'                /* current cursor position  */
'LINE_AFTER (lineno) = DATALINE (displn)' /* insert DISPLAY line */
tocntl = tocntl + 1                      /* increment last line      */
'(lineno,colno) = CURSOR'                /* increment the cursor     */
lineno = lineno + 1                      /*       ""               */
'CURSOR = (lineno)'                      /*       ""               */

'(lineno,colno) = CURSOR'                /* current cursor position  */
'LINE_AFTER (lineno) = DATALINE (botline)'
tocntl = tocntl + 1                      /* increment last line      */
'(lineno,colno) = CURSOR'                /* increment the cursor     */
lineno = lineno + 1                      /*        ""                */
'CURSOR = (lineno)'                      /*        ""                */
return

/* ************************************************************** */
x_tract:             /* Delete all XDP blocks within the given range */
                     /*                                              */
rc = Ø
'RESET EXCLUDED  .ZFIRST .ZLAST'
'CURSOR = (fromcntl)'          /* set cursor to start line          */
'SEEK ".XDPS" NEXT'            /* search for first XDP block start  */
'LABEL .ZCSR = .SPTR Ø'        /* set start pointer/label           */
'CURSOR = (tocntl)'            /* set cursor to last line           */
'SEEK ".XDPE" PREV'            /* search for last XDP block end     */
'LABEL .ZCSR = .EPTR Ø'        /* set end pointer/label             */

if rc = Ø then do
  '(startpos) = LINENUM .SPTR'      /* establish first line         */
  startpos = startpos -1            /* jump back 1 (NEXT|)          */
  'CURSOR = (startpos)'             /* set cursor to first line -1  */
  do forever                             /*                         */
    'SEEK ".XDPS" NEXT'                  /* XDP block start search  */
    if rc ¬= Ø then leave                /* no more found = exit    */
    '(delstart) = LINENUM .ZCSR'    /* set DELETE start position    */
    'SEEK ".XDPE" NEXT'                  /* XDP block end search    */
    if rc ¬= Ø then leave                /* no end found = exit     */
    '(delend) = LINENUM .ZCSR'      /* set DELETE end position      */
```

```
    if delend > tocntl then leave   /* exit if pointers incorrect    */
    'DELETE' delstart delend            /* delete XDP block          */
    '(lineno,colno) = CURSOR'           /* decrement the cursor      */
    lineno = lineno - 1                 /*         ""       (NEXT|) */
    'CURSOR = (lineno)'                 /*         ""                */
    countdown = delend - delstart + 1   /* reset last line position */
    tocntl = tocntl - countdown         /*         ""                */
    delxx = delxx + 1
  end
end
return

/* ************************************************************** */
goto_dis:           /* GO TO routine                               */
                    /* Insert a DISPLAY statement before each GO TO */
                    /* statement within the given range. The       */
                    /* DISPLAY statement will contain per GO TO a  */
                    /* unique counter value and the name of the TO  */
                    /* Section/Paragraph                           */
fromcntl = fromcntl - 1
'CURSOR = (fromcntl)'             /* set cursor to first line -1   */
'SEEK "GO TO" NEXT'               /* search for next GO TO         */
if rc ¬= Ø then return            /* return when no GO TO s found  */
'LABEL .ZCSR = .GPTR Ø'           /* set start limit pointer/label */
'CURSOR = (tocntl)'                   /* set cursor to last line  */
'LABEL .ZCSR = .TPTR Ø'           /* set end limit pointer/label   */

'SEEK "GO TO" .GPTR .TPTR ALL'        /* count GO TO occurrences  */
                                      /* between limits           */
if rc = Ø then do
  '(scntr,lcntr) = SEEK_COUNTS'       /* set loop counter         */
  '(startpos) = LINENUM .GPTR'        /* establish first line -1  */
  startpos = startpos - 1             /*        ""                */
  'CURSOR = (startpos)'               /*        ""                */
  do lcntr                            /*                          */
    'SEEK "GO TO" NEXT'               /* search for next GO TO    */
    if rc ¬= Ø then leave             /* not found = exit         */
    gotoxx = gotoxx + 1               /* increment GO TO counter  */
    gotoxx = right(gotoxx,3,'Ø')      /* format GO TO counter     */
    '(linecnt) = LINENUM .ZCSR'       /* find current line number */
    if linecnt >= tocntl then leave   /* exit when limits false   */
    '(lcontent) = LINE (linecnt)'     /* extract line contents    */
    w1 = find(lcontent,"TO")          /* extract destination      */
    w2 = subword(lcontent,w1+1,1)     /*        ""                */
                              /* build rest of DISPLAY statement   */
    displn ='           DISPLAY "GO  ==>'gotoxx' *'
    displn = insert(w2,displn,35,32)
    displn = insert(",",,displn,67,1)

    '(lineno,colno) = CURSOR'         /* current cursor position  */
```

```
    lineno = lineno - 1                 /* -1                        */
    'CURSOR = (lineno)'                 /* set cursor                */
    'LINE_AFTER (lineno) = DATALINE (topline)'     /* insert comment */
    tocntl = tocntl + 1                 /* increment last line       */
    '(lineno,colno) = CURSOR'           /* increment the cursor      */
    lineno = lineno + 1                 /*        ""                 */
    'CURSOR = (lineno)'                 /*        ""                 */

    '(lineno,colno) = CURSOR'           /* current cursor position   */
    'LINE_AFTER (lineno) = DATALINE (displn)'
    tocntl = tocntl + 1                 /* increment last line       */
    '(lineno,colno) = CURSOR'           /* increment the cursor      */
    lineno = lineno + 1                 /*        ""                 */
    'CURSOR = (lineno)'                 /*        ""                 */

    '(lineno,colno) = CURSOR'           /* current cursor position   */
    'LINE_AFTER (lineno) = DATALINE (botline)'
    tocntl = tocntl + 1                 /* increment last line       */
    '(lineno,colno) = CURSOR'           /* increment the cursor      */
    lineno = lineno + 1                 /*        ""                 */
    'CURSOR = (lineno)'                 /*        ""                 */
    '(lineno,colno) = CURSOR'           /* increment the cursor      */
    lineno = lineno + 1                 /*        ""                 */
    'CURSOR = (lineno)'                 /*        ""                 */
  end
end
return

/* ****************************************************************** */
perf_dis:           /* Perform routine                                */
                    /* Insert a DISPLAY statement before each non-    */
                    /* inline PERFORM statement within the given      */
                    /* range. The DISPLAY statement will contain      */
                    /* per PERFORM a unique counter value and the     */
                    /* name of the called Section/Paragraph.          */
                    /*                                                */
fromcntl = fromcntl - 1             /* set cursor to first line      */
'CURSOR = (fromcntl)'               /* -1                            */
'SEEK " PERFORM " NEXT'             /* seek first occurance          */
if rc ¬= Ø then return          /* return when no PERFORM found      */
'LABEL .ZCSR = .PPTR Ø'             /* set start pointer/label       */
'CURSOR = (tocntl)'                 /* set cursor to last line       */
'LABEL .ZCSR = .QPTR Ø'             /* set end pointer/label         */

'SEEK " PERFORM " .PPTR .QPTR ALL'  /* count xdp occurrences         */
                                    /* between limits                */
if rc = Ø then do
  '(scntr,lcntr) = SEEK_COUNTS'     /*         ""                    */
  '(startpos) = LINENUM .PPTR'      /* establish first line          */
  startpos = startpos - 1
```

```
  'CURSOR = (startpos)'                  /* set cursor to first line */
  do lcntr                               /*                          */
    'SEEK " PERFORM " NEXT'              /* set start position       */
    if rc ¬= Ø then leave
    '(linecnt) = LINENUM .ZCSR'          /* find current line number */
    if linecnt >= tocntl then leave
    '(lcontent) = LINE (linecnt)'
    w1 = find(lcontent," PERFORM ")
    w3 = find(lcontent," UNTIL ")
    w4 = find(lcontent," VARYING ")
    ww = w3 + w4
    if ww = Ø then do                    /* not inline PERFORM ?     */
      perfxx = perfxx + 1                /* increment counter        */
      perfxx = right(perfxx,3,'Ø')       /* format counter           */
      w2 = subword(lcontent,w1+1,1)      /* extract called Paragraph */
                  /* build rest of DISPLAY statement               */
      displn ='           DISPLAY "PER ==>'perfxx' *'
      displn = insert(w2,displn,35,32)
      displn = insert(,",,displn,67,1)

      '(lineno,colno) = CURSOR'          /* current cursor position  */
      lineno = lineno - 1                /* -1                       */
      'CURSOR = (lineno)'                /* set cursor               */
      'LINE_AFTER (lineno) = DATALINE (topline)'
      tocntl = tocntl + 1
      '(lineno,colno) = CURSOR'          /* increment the cursor     */
      lineno = lineno + 1                /*        ""                */
      'CURSOR = (lineno)'                /*        ""                */

      '(lineno,colno) = CURSOR'          /* current cursor position  */
      'LINE_AFTER (lineno) = DATALINE (displn)'
      tocntl = tocntl + 1
      '(lineno,colno) = CURSOR'          /* increment the cursor     */
      lineno = lineno + 1                /*        ""                */
      'CURSOR = (lineno)'                /*        ""                */

      '(lineno,colno) = CURSOR'          /* current cursor position  */
      'LINE_AFTER (lineno) = DATALINE (botline)'
      tocntl = tocntl + 1
      '(lineno,colno) = CURSOR'          /* increment the cursor     */
      lineno = lineno + 1                /*         ""               */
      'CURSOR = (lineno)'                /*         ""               */
      '(lineno,colno) = CURSOR'          /* increment the cursor     */
      lineno = lineno + 1                /*         ""               */
      'CURSOR = (lineno)'                /*         ""               */
    end
  end
end
return
/* ************************************************************** */
```

XDP PANEL

```
)ATTR
  [   TYPE(INPUT) CAPS(ON) INTENS(HIGH) COLOR(WHITE)
  $   TYPE(OUTPUT) INTENS(HIGH) COLOR(PINK)
  ]   TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
  }   TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
  {   TYPE(TEXT) INTENS(HIGH) COLOR(GREEN)
  #   TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
)BODY DEFAULT(%+_) EXPAND(//) WINDOW(6Ø,26) OUTLINE(BOX)
%/-/#COBOL batch debug aid }eXtra-DisPlay /-/
$zuser   +/ /$timestmp        +
+/ /]info                   / /+
/ /}xx      xx  dddddd      pppppppp  / /
/ /}xx      xx  ddddddd     ppppppppp / /
/ /} xx    xx   dd    dd    pp     ppp/ /
/ /} xx    xx   dd     dd   pp      pp/ /
/ /}  xx  xx    dd      dd  pp      pp/ /
/ /}  xx  xx    dd      dd  pp     ppp/ /
/ /}   xxxx     dd      dd  pppppppppp / /
/ /}   xxxx     dd      dd  pppppppp  / /
/ /}  xx  xx    dd      dd  pp        / /
/ /}  xx  xx    dd      dd  pp        / /
/ /} xx    xx   dd     dd   pp        / /
/ /} xx    xx   dd    dd    pp        / /
/ /}xx      xx  ddddddd     pp        / /
/ /}xx      xx  dddddd      pp        / /
+
&  Option:  ===>[z+ S,G,P,A or X
+Limits
&   Start:  ===>_z     +
&     End:  ===>_z     +
+              [z&Exclude XDP comments
)INIT
.HELP = XDPØ1H
.zvars ='(optcntl,fromcntl,tocntl,exccntl)'
&timestmp ='&zday..&zmonth..&zstdyear. &ztime'
)REINIT
REFRESH(*)
)PROC
VER(&optcntl,NB,LIST,S,G,P,A,X,MSG=XDPØ11)
VER(&fromcntl,RANGE,&lowerlim,&upperlim,MSG=XDPØ12)
VER(&tocntl,RANGE,&lowerlim,&upperlim,MSG=XDPØ13)
VER(&tocntl,RANGE,&fromcntl,&upperlim,MSG=XDPØ14)
IF (.MSG NE'')
  &info ='######## ERROR ########'
IF (&exccntl NE'')
  &exccntl ='/'
)HELP
FIELD(optcntl) PANEL(XDPØ1H1)
```

```
FIELD(fromcntl) PANEL(XDPØ1H2)
FIELD(tocntl) PANEL(XDPØ1H3)
FIELD(exccntl) PANEL(XDPØ1H4)
)END
```

XDPA PANEL

```
)ATTR
  [   TYPE(OUTPUT) INTENS(HIGH) COLOR(WHITE) CAPS(OFF)
  $   TYPE(OUTPUT) INTENS(HIGH) COLOR(PINK)
  ]   TYPE(OUTPUT) INTENS(HIGH) COLOR(RED) CAPS(OFF)
  }   TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
  {   TYPE(OUTPUT) INTENS(HIGH) COLOR(GREEN) CAPS(OFF)
  #   TYPE(OUTPUT) INTENS(HIGH) COLOR(TURQ) CAPS(OFF)
  ¬   TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
)BODY DEFAULT(%+_) EXPAND(//) WINDOW(66,26) OUTLINE(BOX)
%/-/¬COBOL batch debug aid }eXtra-DisPlay /-/
$zuser   +/ /$timestmp        +
+/ /]info                   / /+
/ /}xx      xx  dddddd      pppppppp  / /
/ /}xx      xx  ddddddd     ppppppppp / /
/ /} xx    xx   dd    dd    pp     ppp/ /
/ /} xx    xx   dd     dd   pp      pp/ /
/ /}  xx  xx    dd      dd  pp      pp/ /
/ /}  xx  xx    dd      dd  pp     ppp/ /
/ /}   xxxx     dd      dd  pppppppppp / /
/ /}   xxxx     dd      dd  pppppppp  / /
/ /}  xx  xx    dd      dd  pp        / /
/ /}  xx  xx    dd      dd  pp        / /
/ /} xx    xx   dd     dd   pp        / /
/ /} xx    xx   dd    dd    pp        / /
/ /}xx      xx  ddddddd     pp        / /
/ /}xx      xx  dddddd      pp        / /
¬/-/
+/ /[headline                    #varØ +/ /
+
/ /#var1  {line1        +/ /
/ /#var2  {line2        +/ /
/ /#var3  {line3        +/ /
/ /#var4  {line4        +/ /
¬/-/
+Press }ENTER+to continue
)INIT
IF (&delxx NE'')
  &headline =' DISPLAY statements deleted:'
  &varØ = &delxx
ELSE
  &headline =' DISPLAY statements inserted:'
  &var1 = &secxx
```

```
  &var2 = &parxx
  &var3 = &perfxx
  &var4 = &gotoxx
  &line1 ='SECTIONs    '
  &line2 ='Paragraphs  '
  &line3 ='PERFORM calls'
  &line4 ='GO TO calls '
)END
```

XDP01H PANEL

```
)ATTR
  [   TYPE(INPUT) CAPS(ON) INTENS(HIGH) COLOR(WHITE)
  $   TYPE(OUTPUT) INTENS(HIGH) COLOR(PINK)
  ]   TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
  }   TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
  {   TYPE(TEXT) INTENS(HIGH) COLOR(GREEN)
  #   TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
)BODY DEFAULT(%+_) EXPAND(//) WINDOW(6Ø,3Ø) OUTLINE(BOX)
%/-/#COBOL batch debug aid }eXtra-DisPlay /-/
+/ /}HELP SCREEN/ /
+/ /#General Information and Help selection/ /
+Command ===>_zcmd                                +
+ Enter corresponding number for further information
+
{  Option:  ===>%(1){S,G,P,A or X
+Limits
{   Start:  ===>%(2)   +
{     End:  ===>%(3)   +
+                       %(4){Exclude XDP comments
+
+This debug aid is designed to insert DISPLAY statements
+in COBOL program source code. The program must be
+recompiled before the changes take effect. The debug aid
+can insert the DISPLAY statements within a restricted range
+of lines to allow concentrated debugging and reduction
+of unnecessary display output. After and during debugging
+XDP can be used to remove (also within a restricted
+range) the previously inserted DISPLAY statements.
+For PERFORM and GO TO statements a sequence number is
+generated in the DISPLAY to show where the call
+originated.
+For examples of the DISPLAY statements enter%(5)+.
)INIT
)PROC
&ZIND=YES
&ZSEL=TRANS(&ZCMD
           1,*XPDØ1H1
           2,*XPDØ1H2
```

```
           3,*XPDØ1H3
           4,*XPDØ1H4
           5,*XPDØ1H5
           *,'?')
)END
```

XDP01H1 PANEL

```
)ATTR
  [   TYPE(INPUT) CAPS(ON) INTENS(HIGH) COLOR(WHITE)
  $   TYPE(OUTPUT) INTENS(HIGH) COLOR(PINK)
  ]   TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
  }   TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
  {   TYPE(TEXT) INTENS(HIGH) COLOR(GREEN)
  #   TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
)BODY DEFAULT(%+_) EXPAND(//) WINDOW(6Ø,3Ø) OUTLINE(BOX)
%/-/#COBOL batch debug aid }eXtra-DisPlay /-/
+/ /}HELP SCREEN/ /
+/ /#Options/ /
+
&  Option:  ===>    &S,G,P,A or X
+
+The options available are:
%S&Section-Paragraph
+  This option checks the A Margin for Section and Paragraph
+  names. If SECTION is found on the same line then SEC is
+  built into the DISPLAY statement otherwise PAR. The name
+  of the Section-Paragraph is also included in the DISPLAY.
%G&GO TO
+  The GO TO option searches within the range for GO TO
+  statements. A DISPLAY statement is built into the code
+  directly before the GO TO and contains a sequence number
+  for identification and the Destination of the GO TO.
%P&Perform
+  The Perform option searches for non inline Perform
+  statements and treats them similarly to the GO TO
+  statements in the GO TO option.
%A&All
+  The All option combines options S, G, and P in one
+  command.
+
%X&eXclude(remove)
+  The eXclude option allows the deletion of previously
+  inserted XDP Display statements.
+
+For all the options the range of lines may be specified.
)INIT
)PROC
&ZHTOP=XDPØ1H
```

```
&ZUP=XDPØ1H
&ZCONT=XDPØ1H2
)END
```

XDP01H2 PANEL

```
)ATTR
  [   TYPE(INPUT) CAPS(ON) INTENS(HIGH) COLOR(WHITE)
  $   TYPE(OUTPUT) INTENS(HIGH) COLOR(PINK)
  ]   TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
  }   TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
  {   TYPE(TEXT) INTENS(HIGH) COLOR(GREEN)
  #   TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
)BODY DEFAULT(%+_) EXPAND(//) WINDOW(6Ø,3Ø) OUTLINE(BOX)
%/-/#COBOL batch debug aid }eXtra-DisPlay /-/
+/ /}HELP SCREEN/ /
+/ /#Start/ /
+
+Limits
{   Start:  ===>%      +
+
+This is the start position of the range where XDP should
+operate. The default value is the line immediately
+following the PROCEDURE DIVISION statement. This is a
+relative line number and it is therefore recommended to
+renumber the lines prior to using XDP.
)INIT
)PROC
&ZHTOP=XDPØ1H
&ZUP=XDPØ1H1
&ZCONT=XDPØ1H3
)END
```

XDP01H3 PANEL

```
)ATTR
  [   TYPE(INPUT) CAPS(ON) INTENS(HIGH) COLOR(WHITE)
  $   TYPE(OUTPUT) INTENS(HIGH) COLOR(PINK)
  ]   TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
  }   TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
  {   TYPE(TEXT) INTENS(HIGH) COLOR(GREEN)
  #   TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
)BODY DEFAULT(%+_) EXPAND(//) WINDOW(6Ø,3Ø) OUTLINE(BOX)
%/-/#COBOL batch debug aid }eXtra-DisPlay /-/
+/ /}HELP SCREEN/ /
+/ /#End/ /
+
```

```
+Limits
+
{     End:  ===>       +
+
+This is the end position of the range where XDP should
+operate. The default value is the last line. This is a
+relative line number and it is therefore recommended to
+renumber the lines prior to using XDP. The value must
+be greater than the value for the start position.
)INIT
)PROC
&ZHTOP=XDPØ1H
&ZUP=XDPØ1H2
&ZCONT=XDPØ1H4
)END
```

XDP01H4 PANEL

```
)ATTR
  [   TYPE(INPUT) CAPS(ON) INTENS(HIGH) COLOR(WHITE)
  $   TYPE(OUTPUT) INTENS(HIGH) COLOR(PINK)
  ]   TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
  }   TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
  {   TYPE(TEXT) INTENS(HIGH) COLOR(GREEN)
  #   TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
)BODY DEFAULT(%+_) EXPAND(//) WINDOW(6Ø,3Ø) OUTLINE(BOX)
%/-/#COBOL batch debug aid }eXtra-DisPlay /-/
+/ /}HELP SCREEN/ /
+/ /#Exclude comments/ /
+
+             %(4){Exclude XDP comments
+
+Default is on, can be deactivated using "blank".
+This option performs an ISPF EXCLUDE to suppress the XDP
+comment lines. Do not delete the comment lines unless
+the DISPLAY statements are permanently required. If
+the comments are missing XDP cannot select the DISPLAY
+statements for deletion.
)INIT
)PROC
&ZHTOP=XDPØ1H
&ZUP=XDPØ1H3
&ZCONT=XDPØ1H5
)END
```

XDP01H5 PANEL

```
)ATTR
```

```
  [   TYPE(INPUT) CAPS(ON) INTENS(HIGH) COLOR(WHITE)
  $   TYPE(OUTPUT) INTENS(HIGH) COLOR(PINK)
  ]   TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
  }   TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
  {   TYPE(TEXT) INTENS(HIGH) COLOR(GREEN)
  #   TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
)BODY DEFAULT(%+_) EXPAND(//) WINDOW(75,3Ø) OUTLINE(BOX)
%/-/#COBOL batch debug aid }eXtra-DisPlay /-/
+/ /}HELP SCREEN/ /
+/ /#Examples of DISPLAY statements/ /
+
+       MAIN SECTION.
+      * -------- .XDPS * START * ---------- DO NOT CHANGE --------
+           DISPLAY "SEC ==> MAIN                                  " .
+      * -------- .XDPE * END   * ---------- DO NOT CHANGE --------
+
+       CASE-SELECT-2.
+      * -------- .XDPS * START * ---------- DO NOT CHANGE --------
+           DISPLAY "PAR ==> CASE-SELECT-2                         " .
+      * -------- .XDPE * END   * ---------- DO NOT CHANGE --------
+
+      * -------- .XDPS * START * ---------- DO NOT CHANGE --------
+           DISPLAY "GO  ==> Ø12 *  CASE-TEST-25       "
+      * -------- .XDPE * END   * ---------- DO NOT CHANGE --------
+           GO TO CASE-TEST-25
+
+      * -------- .XDPS * START * ---------- DO NOT CHANGE --------
+           DISPLAY "PER ==> ØØ2 *  MASTER-SELECT         "
+      * -------- .XDPE * END   * ---------- DO NOT CHANGE --------
+           PERFORM MASTER-SELECT
)INIT
)PROC
&ZHTOP=XDPØ1H
&ZUP=XDPØ1H4
&ZCONT=XDPØ1H
)END
```

XDP01H5 MESSAGES

```
XDPØ11  'S,G,P,A or X' .TYPE=W .WINDOW=R .ALARM=Y
,S = Sections and Paragraphs, G = GOTO contructions, P = Perform,' +
, and X = eXtract (remove)'
XDPØ12  '&lowerlim < value < &upperlim' .TYPE=W .WINDOW=R .ALARM=Y
,The Start line must be contained in the PROCEDURE DIVISION'
XDPØ13  '&lowerlim < value < &upperlim' .TYPE=W .WINDOW=R .ALARM=Y
,The End line must be contained in the PROCEDURE DIVISION'
XDPØ14'&fromcntl < value < &upperlim' .TYPE=W .WINDOW=R .ALARM=Y
,The End line must come after the start line'
```

CONCLUSION

Although the routine is quite simple it has already proved useful for several programmers within the CMS environment and now, hopefully, for a few more within the TSO world.

FINAL TIP

Due to switching between mainframe and PC and also between various code pages it is possible that the special characters used for the attribute definitions have been changed and are no longer recognizable. Please check before using the panels.

Rolf Parker
Systems Programmer (Germany)

Back-ups and offsite recovery

We use the following system to carry out full-pack back-ups for offsite recovery. It also handles the restores at the recovery site. Back-ups are taken on individual systems, then the relevant catalog information and datasets are copied/merged onto a 'onepack' system, known as 'MVSMINI'. The onepack system can then be restored (stand-alone or otherwise) offsite and the same panels, REXX, etc, used to carry out the full-pack restores. The back-ups are done using DFDSS. We have reserved 'SYS8' as the HLQ for these back-ups, and each system has a UCAT that contains only these catalog entries.

The system is 'driven' by control files in which the back-ups are arranged (logically) into 'suites', called BCKMVSA/B/etc. These back-up suites could easily be manually created/maintained, but here they are automatically generated by the 'OPSREC' utility. A sample entry for one of the back-up members would be:

```
TEST01 6000 3390 Y
```

TEST01 is the volid, 6000 is the address, 3390 is the device type, and the last field is a flag. It can be Y (yes, restore offsite), N (no, do not restore offsite), I (only initialize off, eg for storage volumes where you

are not interested in the original contents of a volume), and X (eXclude from normal restore offsite – this option causes an IEFBR14 job to be submitted so that there is a match between the number of volumes in a suite and the number of jobs submitted).

The other main control file used is for 'system affinity'. This is used to indicate which back-ups run on which system and whether or not those suites are required to be sent offsite. So, a member on system 'SYS1' might look like:

```
A B C F H K T
Y Y N Y Y Y N
```

In this BCKMVSA/B/C/F/H/K/T would all be eligible to run on SYS1, with BCKMVSA/B/F/H/K being sent offsite. Suites which exist in more than one system are flagged as being 'shared', and this simplifies the generation of the correct configuration at our recovery site. All relevant reports are e-mailed to our offsite recovery site, and thus the configurations are kept in sync.

Report processing is included that creates lists of the latest (ie GDG G0) datasets and formats them by system. This is not very sophisticated (but works) and could be replaced by tape management software's 'pull lists' or similar. There is eject processing (for STK's ACS, which could obviously easily be changed to something else.

ODDS AND ENDS

Some little utilities are used, which are described here (some are supplied, some not):

- LASTGEN (supplied) – a CLIST that returns the current generation of a GDG as a return code.
- MVSCMD (not supplied) – a program to issue an MVS command passed to it as a parameter.
- IPLMSG (not supplied) – a program to issue a WTOR displaying a message passed to it as a parameter. Called by the 'WHEN' STC, which is started when the MVSMINI system comes up, instantly showing the creation date of that system, so that the operators can verify this before continuing.

- PRTMEMS (supplied) – prints the back-up suites and each restore job's jobname. Used as a runsheet.
- SUPERSCR (not supplied) – a program that enables you to delete duplicate datasets, even if the original one is allocated.
- ADRDSSU exit ADRUENQ (supplied) – default enqueue processing will hold the VTOC for the duration of a back-up operation. If you are carrying out back-ups while systems are up (fuzzy back-ups) this can cause contention problems. If the ADRUENQ exit supplies DFDSS with an RC=4, then the VTOC will only be enqueued while the VTOC itself is being backed up.

I created a separate version of the ADRDSSU (ie DFDSS) program, which can be called if this is required:

```
//jobcard
//* DFDSSDSS                                                        *
//* ASSEMBLE AND LINK A MODIFIED VERSION OF ADRUENQ (FORCES RC=4 TO  *
//* PREVENT LONG ENQUEUE OF VTOC). NOTE THAT THIS DOES NOT UPDATE    *
//* THE REAL ADRDSSU, BUT A COPY IN BQIBI.OPSREC.ADRDSSU.            *
//ASM      EXEC PGM=ASMA90,REGION=256K,PARM='NODECK,RENT'
//SYSPRINT DD SYSOUT=*
//SYSPUNCH DD DUMMY
//SYSLIB   DD DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1   DD UNIT=3390,SPACE=(CYL,(5,1))
//SYSUT2   DD UNIT=3390,SPACE=(CYL,(5,1))
//SYSUT3   DD UNIT=3390,SPACE=(CYL,(5,1))
//SYSLIN   DD DSN=SYSG.UMOD.OBJ(ADRUENQ),DISP=SHR
//SYSIN    DD *
*********************************************************************
*** MODULE: ADRUENQ                                               ***
*** PURPOSE: WILL FORCE A RETURN CODE 4 AND PREVENT THE VTOC      ***
***          BEING ENQUEUED FOR THE DURATION OF A FULL-PACK       ***
***          DUMP, AND PREVENT POTENTIAL LOCKOUTS.                ***
***          MUST BE RE-ENTRANT.                                   ***
***          SEE "DFSMSdss Installation Exits" MANUAL.            ***
*********************************************************************
ADRUENQ  CSECT
ADRUENQ  AMODE 31
ADRUENQ  RMODE 24
         USING *,15
         LA    15,4                    SET RC=4
         BR    14
         END
//LINK     EXEC  PGM=IEWL,REGION=256K,
//         PARM='LET,LIST,NCAL,XREF,RENT,AC=1,AMODE=31',
//         COND=(7,LT)
```

```
//SYSPRINT DD SYSOUT=*
//OBJECT   DD DSN=SYSG.UMOD.OBJ,DISP=SHR
//SYS1LK   DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSUT1   DD UNIT=339Ø,SPACE=(TRK,(2Ø,5))
//SYSLMOD  DD DSN=BQIBI.OPSREC.ADRDSSU,DISP=SHR
//SYSLIN   DD *
 INCLUDE OBJECT(ADRUENQ)
 INCLUDE SYS1LK(ADRDSSU)
 ENTRY   ADRDSSU
 NAME    ADRDSSU(R)
```

ISPF MESSAGES MEMBER

```
MBKP17ØA .ALARM=YES
'>>> Back-ups can only be run on SYS1, SYS2 and SYS3...'
MBKP17ØB .ALARM=YES
'>>> Invalid back-up suite selected - nothing run...'
MBKP17ØC .ALARM=YES
'>>> PF3 pressed - nothing selected...'
MBKP17ØD .ALARM=YES
'>>> Cannot select entry &BKU.) - no volume is allocated to it...'
MBKP17ØE .ALARM=YES
'>>> PF3 pressed - no &SLTP selected for &SUITE....'
MBKP17ØF .ALARM=YES
'>>> Error including &SLTP skeleton &SKELETN - contact SYSTEMS...'
MBKP17ØG .ALARM=YES
'>>> Job &BKJBNM submitted for "BKUPSITE Build Processing"...'
MBKP17ØH .ALARM=YES
'>>> Job &BKJBNM (&DISVOL) already running - Submit "ALL" refused...'
MBKP17ØI .ALARM=YES
'>>> Job &BKJBNM (&DISVOL) already on Input Queue - Submit "ALL" cannot
&MSGX'
MBKP17ØJ .ALARM=YES
'>>> You have selected a non-existant back-up suite (&suite)...'
MBKP17ØK .ALARM=YES
'>>> Target/source volid CANNOT match, and target MUST be a spare
volume.'
MBKP17ØL .ALARM=YES
'>>> PF3 pressed - RESTORE processing terminated...'
MBKP17ØM .ALARM=YES
'>>> Not running at BKUPSITE - cannot perform Recovery Functions...'
MBKP17ØN .ALARM=YES
'>>> &BKJBNM &RUNTYPE submitted for "&SUITE"...'
MBKP17ØO .ALARM=YES
'>>> You cannot run BCKMVSO - use "BKPS" Option 6 to back up ONEPAC'
MBKP17ØP .ALARM=YES
'>>> Job &BKJBNM is already running - nothing submitted..'
MBKP17ØQ .ALARM=YES
'>>> Job &BKJBNM is already on the Input Queue - nothing submitted...'
MBKP17ØR .ALARM=YES
```

```
'>>> Job &BKJBNM submitted to &CMLSTP....'
MBKP17ØS .ALARM=YES
'>>> Job &BKJBNM submitted to refresh and back-up "ONEPAC" datasets...'
MBKP17ØT .ALARM=YES
'&bkmsg'
MBKP17ØU .ALARM=YES
'>>> Please enter "MVSx", where "x" is the Back-up Suite Suffix...'
MBKP17ØV .ALARM=YES
'>>> Please enter a valid CAPid- ØØØ:ØØ, ØØØ:Ø1, ØØ1:ØØ or ØØ1:Ø1'
```

REXX EXECS AND CLISTS

BKPS

```
/* ---------------------------------------------------------- */
/* REXX "BKPS"                                                */
/* Driver for DFDSS Full Pack Back-ups and Restores.          */
/* ---------------------------------------------------------- */
/* ---------------------------------------------------------- */
/*  See where we're running. If we are on "MVSMINI" then we are */
/*  at the recovery site - we will have to request the ID of    */
/*  the system we want to recover first.                       */
/* ---------------------------------------------------------- */
Address "TSO"
"WHEREAMI"                                  /* Where are we then? */
retc = rc                                   /* initialize vars... */
sys = ""
presel = ""
savsys = ""
atcomd = ""
debug = "N"
jobpref = "£GOI"                           /* Jobname prefix     */
Address "ISPEXEC"
If retc =  4 Then sys = "SYS1"              /* SYS1               */
If retc =  8 Then sys = "SYS2"              /* SYS2               */
If retc = 2Ø Then sys = "SYS3"              /* SYS3               */
If retc = 99 Then Do                        /* MINI...            */
  savsys = "MINI"
  atcomd = ">>RECOVERY<<"
  "ISPEXEC DISPLAY PANEL(POPBK17D)" /* ..so see who we're recovering */
  If rc = 8 Then                            /* PF3 = quit...      */
    Return
  presel = "R"                              /* Set for recovery   */
End
If sys = "" Then Do                         /* Unknown system     */
  "ISPEXEC SETMSG MSG(MBKP17ØA)"
  Return
End
/* ---------------------------------------------------------- */
/*  This is where the back-up suites and control files live. The*/
```

```
/*  various members have been generated by the "EOPSREC" cmd.   */
/* ------------------------------------------------------------ */
bkpdsn = "BQIBIØ6.OPSREC.BACKUPS"
contrl = "BQIBIØ6.OPSREC.CONTROL"
/* ------------------------------------------------------------ */
/*  MAXVOLS:                                                    */
/*  The current MAXIMUM number of back-ups that can be contained*/
/*  in a single SUITE. Note that if this value is increased     */
/*  then the scrollable panels will also need to be updated.    */
/* ------------------------------------------------------------ */
maxvols = 6Ø                              /* Current MAX table entries*/
sellist = "ABCDEFGHIJKLMNOPQRSTUVWXYZØ123"
sellist = sellist||sellist
                                          /* ...and jobname suffixes  */
actual_suites = ""            /* List of suites that actually exist */
usrid = USERID()                                      /* For NOTIFY */
authgrp = "N"
group = Left(usrid,5)
If group = "BQIBI" | group = "BQIOS" Then
  authgrp = "Y"
/* ------------------------------------------------------------ */
/*  Ensure the back-up dataset exists - if not, inform user...  */
/* ------------------------------------------------------------ */
Address "TSO"
bkpstat = SYSDSN("'"bkpdsn"'")
ctlstat = SYSDSN("'"contrl"'")
Address "ISPEXEC"
If bkpstat = "DATASET NOT FOUND" Then Do /* Dataset doesn't exist... */
  Say " "
  Say "The Back-up Suites dataset is missing - contact Systems..."
  Say " "
  Return                                  /* Go back to main menu     */
End
If ctlstat = "DATASET NOT FOUND" Then Do /* Dataset doesn't exist... */
  Say " "
  Say "The Control Data dataset is missing - contact Systems..."
  Say " "
  Return                                  /* Go back to main menu     */
End
/* ------------------------------------------------------------ */
/*  Read in the control record for the system we're running on  */
/*  (contains the list of back-up suites that are valid for THIS*/
/*  system)...                                                  */
/* ------------------------------------------------------------ */
this_systems_backups = ""
Address "TSO"
ctl = contrl"(@"sys")"
"ALLOC FI(TEMP1) DA('"ctl"') SHR"
"EXECIO * DISKR TEMP1 (Stem okbkups. FINIS"
"FREE FI(TEMP1)"
If okbkups.Ø = Ø Then Do
```

```
  Say " "
  Say "The 'System Identification Record' for system "sys" cannot"
  Say "be located..... Please contact Systems IMMEDIATELY."
  Say " "
  Return
End
this_systems_backups = okbkups.1
/* ------------------------------------------------------------ */
/*  Get the names of all of the back-up suites from 'bkpdsn'... */
/* ------------------------------------------------------------ */
Call FIND_BACKUPS
/* ------------------------------------------------------------ */
/*  This is the main control section of this program.           */
/* ------------------------------------------------------------ */
Do Forever
  opsel = presel
  "ISPEXEC DISPLAY PANEL(POPBK171)"
  If RC = 8 Then Leave
  Select
    When opsel = '1' Then Do                    /* Run a back-up     */
      jbtp = "B"
      sltp = "Backup"
      pnl = "POPBK172"                          /* Back-up selection */
      skeletn = "SOPBK17Ø"                      /* Back-up skeleton  */
      Call DO_BACKUPS_AND_RESTORES
    End
    When opsel = '2' Then Do                    /* Run a restore     */
      jbtp = "R"
      sltp = "Restore"
      pnl = "POPBK177"                          /* Restore selection */
      skeletn = "SOPBK171"                      /* Restore skeleton  */
      Call DO_BACKUPS_AND_RESTORES
    End
    When opsel = '3' Then                       /* Call SDSF         */
      Call DISPLAY_STATUS
    When opsel = '4' Then                       /* Display back-up   */
      Call DISPLAY_SUITE                        /* 'suite' contents  */
    When opsel = '5' Then                       /* Create/recreate   */
      Call CREATE_STANDALONE                    /* IPLable cart      */
    When opsel = '6' Then                       /* Refresh cats, etc */
      Call REFRESH_ONEPAC                       /* on ONEPAC system  */
    When opsel = '7' Then                       /* Create listings   */
      Call RECOVERY_PRINT                       /* of back-ups       */
    When opsel = '8' Then                       /* Mass cart eject   */
      Call EJECT_CARTS                          /* from the library  */
    When opsel = '9' Then                       /* Eject some carts  */
      Call PARTIAL_EJECT                        /* from the library  */
    When opsel = 'R' Then Do                    /* Recovery selected */
      If savsys ¬= "MINI" Then             /* Not running at BKUPSITE*/
        "ISPEXEC SETMSG MSG(MBKP17ØM)"     /* Can't do recovery      */
      Else
```

```
        Call BKUPSITE_RESTORES                  /* Offsite restores  */
    End
  End                                           /* Select            */
End
Return                                         /* Bye bye for now...  */
End
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
DO_BACKUPS_AND_RESTORES:
/* ================================================================ */
/*  This routine does the following:                                */
/*  a) Requests the suite required and checks whether it is         */
/*meant to run on this system; can be overridden if 'authorized'*/
/*  b) Extracts the volumes within the requested suite.             */
/*  c) Enables the user to request individual back-ups or the       */
/*     entire suite, as necessary,                                  */
/*     OR                                                           */
/*     Enables the user to request individual restores.             */
/*  d) Calls the routine to submit the job(s).                      */
/* ================================================================ */
asuite = ""
Do Forever
  "ISPEXEC DISPLAY PANEL(POPBK175)"         /* Show available suites */
  If rc = 8 Then Do                         /* No suite = PF3 hit    */
    "ISPEXEC SETMSG MSG(MBKP170C)"          /* Show nothing selected */
    Return
  End
  Interpret "SUITE = BCKMVS"asuite          /* Get name of suite     */
  If Pos(suite,actual_suites) ¬= Ø Then     /* Make sure it exists   */
    Leave                                   /* Yes..drop thru        */
  Else
    "ISPEXEC SETMSG MSG(MBKP170J)"          /* No...display message  */
End                                         /* Do Forever            */
/* ---------------------------------------------------------------- */
/*  Check that the suite selected is valid to be run on THIS        */
/*  system... If not then we can still have the chance to go        */
/*  ahead and run the back-ups anyway (Exceptional Circumstance)*/
/*  NOTE: That only BQIBI and BQIOS users get the option to run */
/*        back-ups on a system where they are not normally run. */
/* ---------------------------------------------------------------- */
If asuite = "O" Then Do                     /* Can't run 'BCKMVSO'.. */
  "ISPEXEC SETMSG MSG(MBKP170O)"            /* ..as its 'special'..  */
  Return
End
bkms1 = "------ Press ENTER or PF3 to Return ------"
If authgrp = "Y" Then
  bkms1 = "Press ENTER to Continue, or PF3 to Cancel."
If Pos(asuite,this_systems_backups) = Ø Then Do
  "ISPEXEC DISPLAY PANEL(POPBK174)"
  If authgrp ¬= "Y" Then Do                /* Non-authorized user    */
    "ISPEXEC SETMSG MSG(MBKP170B)"
    Return
```

```
  End
  If rc = 8 Then Do                        /* PF3 by authorized user */
    "ISPEXEC SETMSG MSG(MBKP170C)"
    Return
  End
End
/* ------------------------------------------------------------ */
/*  The next call will extract a list of the volumes that are   */
/*  in the member 'suite'...                                    */
/* ------------------------------------------------------------ */
Call EXTRACT_VOLUMES
/* ------------------------------------------------------------ */
/*  Display the list of volumes to be backed up or restored for */
/*  this suite and allow selection of ALL (back-ups) or indiv-  */
/*  idual volumes (back-ups and restores).                      */
/* ------------------------------------------------------------ */
bku = ""
jobnm = jobpref||Right(suite,1)jbtp        /* Suffix "B" or "R"      */
subbed = "N"
dosub = "Y"
Do Forever
  zcmd = ""
  "ISPEXEC DISPLAY PANEL("pnl")"           /* Volume selection panel */
  If rc = 8 Then Do                        /* RC8 = PF3, so leave    */
    If subbed = "N" Then                   /* If nothing was subbed, */
      "ISPEXEC SETMSG MSG(MBKP170E)"       /* show message...        */
    Leave                                  /* Go back to previous pnl*/
  End
  /* ------------------------------------------------------------ */
  /*  If it's a restore, double check that the details are correct!!!*/
  /*  Also, ensure that the target volid and original volid are     */
  /*  different (a restore MUST be done to a spare volume).         */
  /* ------------------------------------------------------------ */
  If opsel = '2' Then Do                   /* Restore                */
    Interpret 'volid = bkvl'bku            /* Volid to be restored   */
    "ISPEXEC DISPLAY PANEL(POPBK17A)"      /* Now double check its OK*/
    If rc = 8 Then Do                      /* PF3 - don't submit job */
      "ISPEXEC SETMSG MSG(MBKP170L)"       /* ...show message...     */
      dosub = "N"                          /* ...and set flag...     */
    End
    Else Do
      If volid = btgtvl Then Do            /* If equal, disallow...  */
        "ISPEXEC SETMSG MSG(MBKP170K)"     /* ...show message...     */
        dosub = "N"                        /* ...and set flag...     */
      End
      If Left(btgtvl,2) <> "MV" Then Do    /* Target MUST be a spare,*/
        "ISPEXEC SETMSG MSG(MBKP170K)"     /* otherwise refuse.      */
        dosub = "N"
      End
    End
  End
```

```
  bku = Strip(bku)
  If dosub ¬= "N" Then
    Call SUB_JOB
End                                        /* Do Forever             */
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
DISPLAY_STATUS:
/* ====================== Option 3 ========================== */
/*  Use SDSF to display the status of backup or restore jobs... */
/* ========================================================== */
suite = ""
sltp = "Status"
Do Forever
  "ISPEXEC DISPLAY PANEL(POPBK175)"         /* Go get suite name     */
  If rc = 8 Then Do                         /* RC8 = PF3 so,         */
    "ISPEXEC SETMSG MSG(MBKP17ØC)"          /* ...set message        */
    Return                                  /* ...and go back        */
  End
  Interpret "SUITE = BCKMVS"asuite          /* Get name of suite     */
  If Pos(suite,actual_suites) ¬= Ø Then     /* Make sure it exists   */
    Leave                                   /* Yes..drop thru        */
  Else
    "ISPEXEC SETMSG MSG(MBKP17ØJ)"          /* No...display message  */
End                                         /* Do Forever            */
prefx = jobpref||asuite"B*"
If savsys = "MINI" Then                        /* If at BKUPSITE...  */
  prefx = "RSMVS"||asuite"*"                   /* ...different jobnm */
"ISPEXEC SELECT PGM(ISFISP) PARM(PRE "prefx")"
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
DISPLAY_SUITE:
/* ====================== Option 4 ========================== */
/*  Display the contents of the various back-up/restore suites  */
/* ========================================================== */
Do Forever
  Do Forever
    asuite = ""
    sltp = "Display"
    "ISPEXEC DISPLAY PANEL(POPBK175)"       /* Show available suites */
    If rc = 8 Then                          /* PF3 - go back         */
      Return
    resp = asuite
    Interpret "SUITE = BCKMVS"asuite        /* Get name of suite     */
    If Pos(suite,actual_suites) ¬= Ø Then   /* Make sure it exists   */
      Leave                                 /* Yes..drop thru        */
    Else
      "ISPEXEC SETMSG MSG(MBKP17ØJ)"        /* No...display message  */
  End                                       /* Do Forever            */
/* ---------------------------------------------------------- */
/*  The next call will extract a list of the volumes that are   */
/*  in the member 'suite'...                                    */
```

```
/* ---------------------------------------------------------- */
  Do a = 1 to maxvols                          /* Make sure we get a */
    Interpret 'bkvl'a' = ""'                   /* nice clean display */
    Interpret 'bkad'a' = ""'
    Interpret 'bkdv'a' = ""'
  End
  Call EXTRACT_VOLUMES
/* ---------------------------------------------------------- */
/*  Display the list of volumes in the selected suite.        */
/*  Note that the user has the option on the panel to elect to  */
/*  print the suite contents. If this is chosen, we will submit */
/*  a job to do this...                                       */
/* ---------------------------------------------------------- */
  bkjbnm = jobpref||Right(suite,1)
  Do Forever
    "ISPEXEC DISPLAY PANEL(POPBK178)"      /* Show volumes involved  */
    If rc = 8 Then                         /* RC8 = PF3, so leave    */
      Leave
  End                                          /* Do Forever         */
End                                            /* Do Forever         */
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
FIND_BACKUPS:
/* ========================================================== */
/*  Get a list of all the back-up suites found in the dataset   */
/*  specified in 'bkpdsn', then display them so they can        */
/*  be selected...                                            */
/* ========================================================== */
Address "TSO"
"DROPBUF"
x = outtrap("out.",100,noconcat)
"LISTDS '"bkpdsn"' MEMBERS"
/* ---------------------------------------------------------- */
/*  Build up fields for panel... These are:                   */
/*   BKnn           Eyecatcher (just a '*')                   */
/*   SUITEnn        Name of the suite (eg 'BCKMVSA')          */
/*   ACTUAL_SUITES  Table of actual suite names found         */
/*   BKnnATTR       Flag set so that suites that are meant to   */
/*                  run on THIS system are highlighted        */
/* ---------------------------------------------------------- */
xx = 0
Do a = 7 to out.0                /* First 6 lines contain headings */
  xx = xx + 1
  member = Strip(out.a)                        /* Get member name      */
  If member = "COMDBKPS" Then Do               /* Ignore 'special' ones*/
    xx = xx - 1
    Iterate
  End
  Interpret "BK"xx" = '   *'"                  /* Set panel variables  */
  Interpret "SUITE"xx" = MEMBER"               /*  "    "       "      */
  suite = Right(member,1)                      /* Get suite id         */
```

```
  If suite = "@" Then Do                    /* Ignore 'special' ones*/
    xx = xx - 1
    Iterate
  End
  actual_suites = actual_suites' 'member    /* Build list of suites */
                                            /*  that actually exist */
  If Pos(suite,this_systems_backups) ¬= Ø Then
    Interpret "BK"xx"ATTR = 'HI'"           /* Hilite if for our sys*/
  Else
    Interpret "BK"xx"ATTR = 'LO'"           /* Else low intensity   */
End
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
EXTRACT_VOLUMES:
/* ============================================================== */
/*  Get the list of volumes found in the member the user has    */
/*  selected (this is specified in variable 'suite')...         */
/* ============================================================== */
Address "TSO"
dsn = bkpdsn"("suite")"
"ALLOC FI(TEMP1) DA('"dsn"') SHR"
"EXECIO * DISKR TEMP1 (Stem line. FINIS"
"FREE FI(TEMP1)"
b = Ø
Do a = 1 to maxvols
  Interpret 'bkvl'a' = "*NONE*"'         /* Pre-init panel variables */
  Interpret 'B'a' = 'a'||")"'
End
Do a = 1 to line.Ø
  Parse Upper Var line.a bkvol bkadd bkdev bkcom .
  b = b + 1                                 /* Count of vols to back up */
  Interpret 'bkvl'a' = bkvol'               /* Assign to variables      */
  Interpret 'bkad'a' = bkadd'               /*   "    "      "          */
  Interpret 'bkdv'a' = bkdev'               /*   "    "      "          */
  Interpret 'bkcm'a' = bkcom'               /*   "    "      "          */
End
volcnt = b                                  /* Save count of backup vols*/
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
SUB_JOB:
/* ============================================================== */
/*  Routine to submit the job(s)...                             */
/* ============================================================== */
/* -------------------------------------------------------------- */
/*  Create the back-up job(s) in 'ZTEMPF' dataset (if doing ALL */
/*  then all jobs are created at once and subb'ed together).    */
/*  Before submitting we call 'PURGE_OUTPUT' to remove any      */
/*  residual output from previous jobs and also to check to     */
/*  see whether there are already any jobs with the same name   */
/*  either running or on the Input Queue; if any are found      */
/*  then we will not submit another job.                        */
```

```
/* ---------------------------------------------------------- */
"ISPEXEC FTOPEN TEMP"                          /* Open ZTEMPF        */
If bku = "ALL" Then Do                         /* ALL back-ups       */
  bb = Ø
  Do a = 1 to volcnt
    suffix = Right('Ø'a,2)                     /* Create job suffix  */
    bkjbnm = jobnm||Right('Ø'a,2)              /* Create jobname     */
    Interpret 'disvol = bkvl'a                /* Set up for Tailoring */
    Call PURGE_OUTPUT                          /* Purge old output   */
    If errflag ¬= "NO" Then Do                 /* Duplicate job, etc */
      Address "ISPEXEC"
      If errflag = "RUN" Then Do               /* Job running: set   */
        "ISPEXEC SETMSG MSG(MBKP17ØH)"         /*  message and leave */
        "ISPEXEC FTCLOSE"                      /* Close temp file    */
        Return
      End
      If errflag = "ONQ" Then Do               /* Job on queue: set  */
        msgx = "be processed until it is removed..."
        "ISPEXEC SETMSG MSG(MBKP17ØI)"         /*  message and leave */
        "ISPEXEC FTCLOSE"                      /* Close temp file    */
        Return
      End
    End
    Interpret 'disadr = bkad'a
    If Left(disadr,1) = "Ø" Then disadr = "/"Right(disadr,3)
    Interpret 'disdev = bkdv'a
    Call BKUPSITE_Bit     /* May need different skeleton at BKUPSITE */
/* ---------------------------------------------------------- */
/*  Get skeleton and sub job if all OK...                     */
/* ---------------------------------------------------------- */
    "ISPEXEC FTINCL "skeletn                   /* Do File Tailoring  */
    If rc ¬= Ø Then Do                         /* Problems? Set a    */
      "ISPEXEC SETMSG MSG(MBKP17ØF)"           /*  message and leave */
      Return
    End
  End
End
Else Do                                        /* Single back-up     */
  a = bku                                      /* Get back-up number */
  Interpret 'volid = bkvl'a                    /* Get volid          */
  bkjbnm = jobnm||Right('Ø'a,2)                /* Create jobname     */
  Interpret 'disvol = bkvl'a                  /* Set up for Tailoring */
  Call PURGE_OUTPUT                            /* Purge old output   */
  If errflag ¬= "NO" Then Do                   /* Duplicate job, etc */
    Address "ISPEXEC"
    If errflag = "RUN" Then Do                 /* Job running: set   */
      "ISPEXEC SETMSG MSG(MBKP17ØP)"           /*  message and leave */
      "ISPEXEC FTCLOSE"                        /* Close temp file    */
      Return
    End
    If errflag = "ONQ" Then Do                 /* Job on queue: set  */
```

```
      "ISPEXEC SETMSG MSG(MBKP170Q)"           /*  message and leave */
      "ISPEXEC FTCLOSE"                        /* Close temp file    */
      Return
    End
  End
  Else Do
    Interpret 'disadr = bkad'a
    If Left(disadr,1) = "Ø" Then disadr = Right(disadr,3)
    Interpret 'disdev = bkdv'a
    Call BKUPSITE_Bit     /* May need different skeleton at BKUPSITE */
    "ISPEXEC FTINCL "skeletn                   /* File Tailoring     */
    If rc ¬= Ø Then Do                         /* Problems? Set a    */
      "ISPEXEC SETMSG MSG(MBKP170F)"           /*  message and leave */
      Return
    End
  End
End
"ISPEXEC FTCLOSE"                              /* Close temp file    */
Call SUBMIT_RTN                                /* Actually submit it */
subbed = "Y"
runtype = "(ALL backups)"
If bku ¬= "ALL" Then
  runtype = "("volid" backup)"
Else
  bkjbnm = ""                              /* Reset for following msg */
If opsel = "2" Then
  runtype = "("volid" restore)"
If opsel = "R" Then Do
  If bku = "ALL" Then
    runtype = "(ALL restores)"
  Else
    runtype = "("volid" restore)"
End
Address "ISPEXEC"
"ISPEXEC SETMSG MSG(MBKP170N)"
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
BKUPSITE_BIT:
/* ------------------- BKUPSITE ----------------------------- */
/*  Set up values for restores to be substituted into skeleton. */
/*  Note that the TARGET volumes at BKUPSITE should be already  */
/*  initialized with a volid of 'SI'ccuu where 'SI' is a prefix */
/*  used by BKUPSITE and ccuu is OUR address (the volumes will  */
/*  be attached to the guest machines as OUR addresses).        */
/* ------------------------------------------------------------ */
If savsys = "MINI" Then Do                /* If at BKUPSITE...      */
  skeletn = "SOPBK173"                    /* Restore skeleton       */
  Interpret 'comflag = bkcm'a             /* Get restore flag       */
  If comflag = "X" Then                   /* Excluded from restore? */
    skeletn = "SOPBK177"                  /* Yes - sub a BR14 job   */
  Interpret 'comunit = bkad'a             /* Address to restore to  */
```

```
  comdev = disdev                          /* Devtype to restore to  */
  comvol = "SI"comunit                     /* Volid to restore to    */
  comunit = disadr                         /* Address to restore to  */
  If Left(comunit,1) = "/" Then            /* Remove leading '/'     */
    comunit = Right(comunit,3)
  Interpret 'comflag = bkcm'a              /* Restore or initialize? */
  If comflag = "I" Then                    /* Just initialize so...  */
    skeletn = "SOPBK174"                   /* ...include init skel   */
End
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
PURGE_OUTPUT:
/* ================================================================ */
/*  As the 'STATUS' command will also report on any old output  */
/*  still on the Output Queue, each time we sub a job we will   */
/*  first purge any output remaining with our jobname. At the   */
/*  same time we'll check to make sure there's not already any  */
/*  jobs on the Input Queue with the same name. If this is the  */
/*  case, or if the job is in fact already running, then we'll  */
/*  return the following in 'errflag':                          */
/*            ONQ    Back-up job already on Input Queue         */
/*            RUN    Back-up job already running                */
/*  MODIFICATION: Instead of purging the old output, we will    */
/*  use the '£TO' command to requeue it to the output queue.    */
/* ================================================================ */
errflag = "NO"
x = Outtrap("out.",1Ø,"NOCONCAT")              /* Trap output        */
Address "TSO"
/*---------------------------------------------------------------- */
/* The expected responses from the 'STATUS' command are:           */
/*  IKJ56192I JOB jobname(Jnnn) ON OUTPUT QUEUE  (possibly multiple) */
/*  IKJ56197I JOB jobname(Jnnn) WAITING FOR EXECUTION              */
/*     or                                                          */
/*  IKJ56197I JOB jobname(Jnnn) WAITING FOR EXECUTION, IN HOLD STAT  */
/*  IKJ562Ø2I JOB jobname NOT FOUND                                */
/*  IKJ56211I JOB jobname(Jnnn) EXECUTING                          */
/*---------------------------------------------------------------- */
"STATUS "bkjbnm                                /* Inquire on job     */
x = Outtrap("OFF")                             /* Set OUTTRAP off    */
Do c = 1 to out.Ø
  Queue out.c
  Parse PULL msgid . stjbnm .
  If msgid = "IKJ56197I" Then Do               /* If on input queue  */
    errflag = "ONQ"                            /* ...set the flag... */
    Leave                                      /* ...and get out...  */
  End
  If msgid = "IKJ56211I" Then Do               /* If already running */
    errflag = "RUN"                            /* ...set the flag... */
    Leave                                      /* ...and get out...  */
  End
  If msgid = "IKJ56192I" Then                  /* If on output queue */
```

```
    "ESACMD2 '£TOJOBQ,JM="bkjbnm",ALL,Q=2,NDISP=WRITE'" /* Re-queue  */
End
Address "ISPEXEC"
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
CREATE_STANDALONE:
/* ====================== Option 5 ========================== */
/*  Create an IPLable stand-alone DFDSS cartridge (requires an  */
/*  'NL' cart).                                                 */
/* ============================================================ */
If sys <> "SYS1" Then Do
  bkmsg = "Please run the 'Create Stand-alone Cart' Option on MVSSYS1"
  "ISPEXEC SETMSG MSG(MBKP17ØT)"
  Return
End
wklyjb = "S/A CART"
jobdesc = "Creation of an IPLable Stand-alone DFDSS Cart"
"ISPEXEC DISPLAY PANEL(POPBK17G)"
If rc = 8 Then Do                          /* PF3 hit - don't sub   */
  "ISPEXEC SETMSG MSG(MBKP17ØC)"           /* Show nothing selected */
  Return
End
bkjbnm = usrid"S"                             /* Create jobname     */
"ISPEXEC FTOPEN TEMP"                         /* Open ZTEMPF        */
"ISPEXEC FTINCL SOPBK176"                     /* File Tailoring     */
"ISPEXEC FTCLOSE"                             /* Close temp file    */
"ISPEXEC VGET (ZTEMPF)"                       /* Get temp filename  */
If debug = "Y" Then
  "ISPEXEC EDIT DATASET('"ztempf"')"          /* Debugging          */
Call SUBMIT_RTN
bkmsg = ">>> Job "bkjbnm" submitted to create Stand-alone DFDSS cart."
bkmsg = bkmsg" NOTE that a scratch 'NL' cart will be required outside"
bkmsg = bkmsg" the ACS."
"ISPEXEC SETMSG MSG(MBKP17ØT)"                /* Say its submitted  */
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
REFRESH_ONEPAC:
/* ====================== Option 6 ========================== */
/*  Submit the job to refresh the back-up catalogs and other    */
/*  datasets for recovery at BKUPSITE...                        */
/* ============================================================ */
If sys <> "SYS1" Then Do
  bkmsg = "Please run 'Generate BKUPSITE Restores' on MVSSYS1"
  "ISPEXEC SETMSG MSG(MBKP17ØT)"
  Return
End
wklyjb = "GENREST"
jobdesc = "Generate BKUPSITE Restores/Backup 'ONEPAC'"
"ISPEXEC DISPLAY PANEL(POPBK17G)"
If rc = 8 Then Do                          /* PF3 hit - don't sub   */
  "ISPEXEC SETMSG MSG(MBKP17ØC)"           /* Show nothing selected */
```

```
  Return
End
bkjbnm = usrid"R"                              /* Create jobname     */
blddat = Date()
"ISPEXEC FTOPEN TEMP"                          /* Open ZTEMPF        */
"ISPEXEC FTINCL SOPBK172"                      /* File Tailoring     */
"ISPEXEC FTCLOSE"                              /* Close temp file    */
"ISPEXEC VGET (ZTEMPF)"                        /* Get temp filename  */
If debug = "Y" Then
  "ISPEXEC EDIT DATASET('"ztempf"')"           /* Debugging          */
Call SUBMIT_RTN
"ISPEXEC SETMSG MSG(MBKP17ØS)"                 /* Say its submitted  */
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
RECOVERY_PRINT:
/* ====================== Option 7 ========================== */
/*  Submit the job to print the listings for BKUPSITE...        */
/*  First see if they want the lot or just a subset.            */
/* ========================================================== */
If sys <> "SYS1" Then Do
  bkmsg = "Please run the 'BKUPSITE Print' Option on MVSSYS1"
  "ISPEXEC SETMSG MSG(MBKP17ØT)"
  Return
End
wklyjb = "COMDPRNT"
jobdesc = "Create BKUPSITE Print"
"ISPEXEC DISPLAY PANEL(POPBK17G)"
If rc = 8 Then Do                           /* PF3 hit - don't sub   */
  "ISPEXEC SETMSG MSG(MBKP17ØC)"            /* Show nothing selected */
  Return
End
cmlstp = "produce FULL BKUPSITE Listings"      /* Listing header     */
outmbr = "ALL"                                 /* Cart eject member  */
bkjbnm = usrid"P"                              /* Create jobname     */
"ISPEXEC FTOPEN TEMP"                          /* Open ZTEMPF        */
"ISPEXEC FTINCL SOPBK175"                      /* File Tailoring     */
"ISPEXEC FTCLOSE"                              /* Close temp file    */
"ISPEXEC VGET (ZTEMPF)"                        /* Get temp filename  */
If debug = "Y" Then
  "ISPEXEC EDIT DATASET('"ztempf"')"           /* Debugging          */
Call SUBMIT_RTN
"ISPEXEC SETMSG MSG(MBKP17ØR)"                 /* Say its submitted  */
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
EJECT_CARTS:
/* ====================== Option 8 ========================== */
/*  Submit the job to eject the carts from the ACS. The carts   */
/*  are in a member in 'BQIBI.OPSREC.EJECT', which has been     */
/*  created using Option '7' (BKUPSITE Listings). These could   */
/*  be all of the carts for a week which need to go to BKUPSITE */
/*  or an individual suite.                                     */
```

```
/* ========================================================== */
If sys <> "SYS1" Then Do
  bkmsg = "Please run the 'Eject BKUPSITE carts' Option on MVSSYS1"
  "ISPEXEC SETMSG MSG(MBKP17ØT)"
  Return
End
wklyjb = "EJECTCRT"
jobdesc = "Eject BKUPSITE Carts"
"ISPEXEC DISPLAY PANEL(POPBK17G)"
If rc = 8 Then Do                          /* PF3 hit - don't sub   */
  "ISPEXEC SETMSG MSG(MBKP17ØC)"           /* Show nothing selected */
  Return
End
cmlstp = "eject ALL carts..."
bkjbnm = usrid"E"                             /* Create jobname     */
"ISPEXEC FTOPEN TEMP"                         /* Open ZTEMPF        */
"ISPEXEC FTINCL SOPBK178"                     /* File Tailoring     */
"ISPEXEC FTCLOSE"                             /* Close temp file    */
"ISPEXEC VGET (ZTEMPF)"                       /* Get temp filename  */
If debug = "Y" Then
  "ISPEXEC EDIT DATASET('"ztempf"')"          /* Debugging          */
Call SUBMIT_RTN
"ISPEXEC SETMSG MSG(MBKP17ØR)"                /* Say its submitted  */
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
PARTIAL_EJECT:
/* ====================== Option 9 ========================== */
/*  Submit the job to eject just the carts for a selected suite */
/*  rather than all current back-ups...                         */
/* ========================================================== */
If sys <> "SYS1" Then Do
  bkmsg = "Please run the 'Selective Eject' Option on MVSSYS1"
  "ISPEXEC SETMSG MSG(MBKP17ØT)"
  Return
End
cmls = ""
cmprm = ""
"ISPEXEC DISPLAY PANEL(POPBK17F)"
If rc = 8 Then Do                          /* PF3 hit - don't sub   */
  "ISPEXEC SETMSG MSG(MBKP17ØC)"           /* Show nothing selected */
  Return
End
cmlstp = "eject BCK"cmls" carts"              /* Listing header     */
cmprm = "BCK"cmls                             /* Parm for skeleton  */
bkjbnm = usrid"E"                             /* Create jobname     */
"ISPEXEC FTOPEN TEMP"                         /* Open ZTEMPF        */
"ISPEXEC FTINCL SOPBK179"                     /* File Tailoring     */
"ISPEXEC FTCLOSE"                             /* Close temp file    */
"ISPEXEC VGET (ZTEMPF)"                       /* Get temp filename  */
If debug = "Y" Then
  "ISPEXEC EDIT DATASET('"ztempf"')"          /* Debugging          */
```

```
Call SUBMIT_RTN
"ISPEXEC SETMSG MSG(MBKP17ØR)"                  /* Say its submitted  */
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
SUBMIT_RTN:
Address "ISPEXEC"
"ISPEXEC VGET (ZTEMPF)"                         /* Get temp filename  */
If debug = "Y" Then
  "ISPEXEC EDIT DATASET('"ztempf"')"            /* Debugging...       */
Address "TSO"
"SUBMIT '"ztempf"'"
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
BKUPSITE_RESTORES:
/* ====================== Option R ========================== */
/*  Display the selection panel to enable the Ops to submit the */
/*  relevant recovery jobs at BKUPSITE.                         */
/* ============================================================ */
/*  This routine does the following:                            */
/*  a) Requests the suite required and checks whether it is     */
/*     meant to run on this system (can be overridden by Ops).  */
/*  b) Extracts the volumes within the requested suite.         */
/*  c) Enables the user to request individual restores or the   */
/*     entire suite, as necessary                               */
/*     OR                                                       */
/*     Enables the user to request individual restores.         */
/*  d) Calls the routine to submit the job(s).                  */
/* ============================================================ */
asuite = ""
sltp = "Recovery"
Do Forever
  "ISPEXEC DISPLAY PANEL(POPBK175)"         /* Show available suites */
  If rc = 8 Then Do                         /* No suite = PF3 hit    */
    "ISPEXEC SETMSG MSG(MBKP17ØC)"          /* Show nothing selected */
    Return
  End
  Interpret "SUITE = BCKMVS"asuite          /* Get name of suite     */
  If Pos(suite,actual_suites) ¬= Ø Then     /* Make sure it exists   */
    Leave                                   /* Yes..drop thru        */
  Else
    "ISPEXEC SETMSG MSG(MBKP17ØJ)"          /* No...display message  */
End                                         /* Do Forever            */
/* ---------------------------------------------------------- */
/*  Check that the suite selected is valid to be run on this    */
/*  system... If not then we can still have the chance to go    */
/*  ahead and run the restore anyway (Exceptional Circumstance) */
/*  NOTE: That only specific users get the option to run the    */
/*        restores on a system where they are not normally run. */
/* ---------------------------------------------------------- */
bkms1 = "------ Press ENTER / PF3 to Return ------"
If authgrp = "Y" Then
```

```
  bkms1 = "Press ENTER to Continue, or PF3 to Cancel."
If Pos(asuite,this_systems_backups) = Ø Then Do
  "ISPEXEC DISPLAY PANEL(POPBK174)"
  If authgrp ¬= "Y" Then Do
    "ISPEXEC SETMSG MSG(MBKP17ØB)"
    Return
  End
  If rc = 8 Then Do
    "ISPEXEC SETMSG MSG(MBKP17ØC)"
    Return
  End
End
/* ------------------------------------------------------------ */
/*  The next call will extract a list of the volumes that are   */
/*  in the member 'suite'...                                    */
/* ------------------------------------------------------------ */
Call EXTRACT_VOLUMES
/* ------------------------------------------------------------ */
/*  Display the list of volumes to be restored for this suite   */
/*  and allow selection of ALL restores or individual volumes.  */
/* ------------------------------------------------------------ */
bkjbnm = "RSMVS"||Right(suite,1)          /* eg RSMVSA              */
jobnm = bkjbnm
subbed = "N"
dosub = "Y"
Do Forever
  bku = ""
  zcmd = ""
  bgn = "Ø"
  "ISPEXEC DISPLAY PANEL(POPBK17E)"       /* Volume selection panel */
  If rc = 8 Then Do                       /* RC8 = PF3, so leave    */
    If subbed = "N" Then                  /* If nothing was subbed, */
      "ISPEXEC SETMSG MSG(MBKP17ØE)"      /* show message...        */
    Leave                                 /* Go back to previous pnl*/
  End
  bku = Strip(bku)
  If dosub ¬= "N" Then
    Call SUB_JOB
End                                       /* Do Forever             */
Return
/* ------------------------------------------------------------ */
/* REXX "DRMSG1"                                                */
/* SEND A MESSAGE TO A TSO USER IF A D/R BACKUP FAILS           */
/* ------------------------------------------------------------ */
  PARSE UPPER ARG JOBN SUITE VOLID
  ADDRESS "TSO"
  "SEND '==> "JOBN" FAILED (SUITE: "SUITE" VOLID: "VOLID" <==')",
  U(BQIBIØ6)"
/* ------------------------------------------------------------ */
/* REXX "EJMSG"                                                 */
/* Send a msg to the user who subbed a failed cart eject job */
```

```
/* ---------------------------------------------------- */
Parse Upper Arg user jobn .
ADDRESS "TSO"
"SEND '==> Job "jobn" (cart eject run) has failed. <==',
      U("user")"
Return
```

FAILMAST

```
/* ---------------------------------------------------- */
/* REXX "FAILMAST"                                      */
/* Produce disk/cart listings for BKUPSITE.             */
/* ---------------------------------------------------- */
/* ==================================================== */
/*            The following listings are produced:      */
/*            - All disks for MVSSYS1 (sorted by address) */
/*            - All disks for MVSSYS2 (sorted by address) */
/*            - All disks for MVSSYS3 (sorted by address) */
/*            - All disks (sorted by address within suite) */
/*            As the SYS1/SYS2/SYS3 listings are sorted in a */
/*            different order to the 'all' listing, the input */
/*            (allocated in the JCL) will be different.  */
/*            Output is to DDNAMEs DASDLST (for the three */
/*            listings by system) and DASDALL (for the 'all' */
/*            listing).                                 */
/*                                                      */
/*            Run in batch (in skeleton SOPBK175).      */
/* ==================================================== */
Address "TSO"
blank1 = "     "
/* ---------------------------------------------------- */
/*  This is where the "COMDBKPS" member (that contains a list */
/*  of all of the disks that are backed up and whose tapes go */
/*  to BKUPSITE) is held.                               */
/* ---------------------------------------------------- */
bkpdsn = "BQIBI.OPSREC.BACKUPS"
bkpstat = SYSDSN("'"bkpdsn"'")
/* ---------------------------------------------------- */
/*  This is where the "RECDSKM" Control File lives.     */
/* ---------------------------------------------------- */
contrl = "BQIBIØ6.OPSREC.CONTROL"
ctlstat = SYSDSN("'"contrl"'")
/* ---------------------------------------------------- */
/*  Make sure the datasets exist...                     */
/* ---------------------------------------------------- */
If bkpstat = "DATASET NOT FOUND" Then Do /* Dataset doesn't exist... */
  Say ">>>>>"
  Say ">>>>> BACKUPS DATASET "bkpdsn" NOT FOUND..."
  Say ">>>>>"
  Return (8)                          /* End...          */
End
```

```
If ctlstat = "DATASET NOT FOUND" Then Do /* Dataset doesn't exist... */
  Say ">>>>>"
  Say ">>>>> CONTROL DATASET "contrl" NOT FOUND..."
  Say ">>>>>"
  Return (8)                              /* End...                  */
End
/* ---------------------------------------------------------- */
/*  Get the Control Record for each system...                 */
/* ---------------------------------------------------------- */
sys = "SYS1"                              /* Get SYS1 backups        */
Call GET_CONTROL
If ctlok ¬= "Y" Then
  Return (8)
Else Do
  SYS1_backups = okbkups.1
  SYS1_restore = okbkups.2
End
sys = "SYS3"                              /* Get SYS3 backups        */
Call GET_CONTROL
If ctlok ¬= "Y" Then
  Return (8)
Else Do
  SYS3_backups = okbkups.1
  SYS3_restore = okbkups.2
End
sys = "SYS2"                              /* Get SYS2 backups        */
Call GET_CONTROL
If ctlok ¬= "Y" Then
  Return (8)
Else Do
  SYS2_backups = okbkups.1
  SYS2_restore = okbkups.2
End
SYS1_dasd_list. = ""                      /* Init. SYS1 stem var     */
SYS1cnt = Ø                               /*       and its count     */
SYS3_dasd_list. = ""                      /* Init. SYS3 stem var     */
SYS3cnt = Ø                               /*       and its count     */
SYS2_dasd_list. = ""                      /* Init. SYS2 stem var     */
SYS2cnt = Ø                               /*       and its count     */
/* ---------------------------------------------------------- */
/*  Read in the back-up info sorted by address...             */
/* ---------------------------------------------------------- */
"EXECIO * DISKR COMDBKP1 (Stem comdbkp1. FINIS"
If comdbkp1.Ø = Ø Then Do
  Say ">>>>>"
  Say ">>>>> NO COMDBKP1 RECORDS FOUND..."
  Say ">>>>>"
  Return (8)                              /* End...                  */
End
/* ---------------------------------------------------------- */
/*  Read in the back-up info sorted by address within suite...  */
```

```
/* ---------------------------------------------------------- */
"EXECIO * DISKR COMDBKP2 (Stem comdbkp2. FINIS"
If comdbkp2.Ø = Ø Then Do
  Say ">>>>>"
  Say ">>>>> NO COMDBKP2 RECORDS FOUND..."
  Say ">>>>>"
  Return (8)                          /* End...                  */
End
/* ----------------------------------------------------------0-- */
/*  Loop thru the back-ups and spit out the details according   */
/*  to which system/device type they're for...                  */
/* ---------------------------------------------------------- */
Do a = 1 to comdbkp1.Ø
  Parse Upper Var comdbkp1.a bkvol bkadd bkdev bksuite bkcom .
  If bkcom = "X" Then                 /* BKUPSITE flag = 'X'...   */
    Iterate                           /* ...not normally restored */
  bit = Right(bksuite,1)              /* Get suite suffix         */
  thesys = ""
  If Pos(bit,SYS1_backups) ¬= Ø Then
    thesys = "SYS1"
  If Pos(bit,SYS2_backups) ¬= Ø Then
    thesys = "SYS2"
  If Pos(bit,SYS3_backups) ¬= Ø Then
    thesys = "SYS3"
  /* Check for suites that are in BOTH camps (eg SYSRES vols) */
  /* If so, mark these as being SHARED so that the disks will   */
  /* automatically defined for all guests at BKUPSITE.          */
  If Pos(bit,SYS1_backups) ¬= Ø Then
    If Pos(bit,SYS3_backups) ¬= Ø Then
      thesys = "COMN"
  If thesys = "" Then Do
    Say ">>>>>"
    Say ">>>>> UNKNOWN BACKUP SUITE ENCOUNTERED: '"bksuite"'"
    Say ">>>>>"
/*  Return (8)  To terminate if this happens */
  End
  If thesys ¬= "COMN" Then Do
    Interpret thesys"cnt = "thesys"cnt + 1"       /* Bump its count   */
    Interpret thesys"_dasd_list."thesys"cnt = bkadd bkvol bkdev bksuite"
  End
  /* For "COMMON" suites, write out to "SYS1", "SYS2" AND "SYS3"...   */
  If thesys = "COMN" Then Do
    bksuite = bksuite"      (COMMON)"
    thesys = "SYS1"
    Interpret thesys"cnt = "thesys"cnt + 1"       /* Bump its count   */
    Interpret thesys"_dasd_list."thesys"cnt = bkadd bkvol bkdev bksuite"
    thesys = "SYS3"
    Interpret thesys"cnt = "thesys"cnt + 1"       /* Bump its count   */
    Interpret thesys"_dasd_list."thesys"cnt = bkadd bkvol bkdev bksuite"
    thesys = "SYS2"
    Interpret thesys"cnt = "thesys"cnt + 1"       /* Bump its count   */
```

```
    Interpret thesys"_dasd_list."thesys"cnt = bkadd bkvol bkdev bksuite"
  End
End
/* ---------------------------------------------------------- */
/*  Now read the records back in for each system and produce  */
/*  the "DASDLIST" listings for each system...                */
/*  *NOTE* The format of the "EXECIO" command on MVS means that */
/*         the "Stem" parameter causes a number to be appended  */
/*         to the Stem variable name. Because of this we will   */
/*         set the variable "xyz1" to our data, but we specify  */
/*         "Stem xyz" on the "EXECIO" Command. I was confused   */
/*         so I thought I'd put this note here... (GC).         */
/* ---------------------------------------------------------- */
/* ---------------------------------------------------------- */
/*  MVSSYS1 disks...                                          */
/* ---------------------------------------------------------- */
today = Date()
xyz1 = "1 Production System 1 Disks Taken to BKUPSITE (MVSSYS1): "today
"EXECIO 1 DISKW DASDLSTF (Stem xyz"
xyz1 = "+ Production System 1 Disks Taken to BKUPSITE (MVSSYS1): "today
"EXECIO 1 DISKW DASDLSTF (Stem xyz"
"EXECIO 1 DISKW DASDLSTF (Stem xyz"
SYS18 = Ø; SYS19 = Ø
Do a = 1 to SYS1cnt
  Parse Upper Var SYS1_dasd_list.a bkadd bkvol bkdev bksuite
  If bkdev = "338Ø" Then                /* For 338Øs...            */
    SYS18 = SYS18 + 1
  If bkdev = "339Ø" Then                /* For 339Ø...             */
    SYS19 = SYS19 + 1
  lin1 = " "bkadd"     "bkvol"     "bkdev"          "bksuite
  "EXECIO 1 DISKW DASDLSTF (Stem lin"    /* Print details          */
End
"EXECIO 1 DISKW DASDLSTF (Stem blank"
lin1 = " 338Øs = "SYS18"   339Øs = "SYS19
"EXECIO 1 DISKW DASDLSTF (Stem lin"
/* ---------------------------------------------------------- */
/*  MVSSYS2 disks...                                          */
/* ---------------------------------------------------------- */
xyz1 = "1 Production System 2 Disks Taken to BKUPSITE (MVSSYS2): "today
"EXECIO 1 DISKW DASDLSTC (Stem xyz"
xyz1 = "+ Production System 2 Disks Taken to BKUPSITE (MVSSYS2): "today
"EXECIO 1 DISKW DASDLSTC (Stem xyz"
"EXECIO 1 DISKW DASDLSTC (Stem xyz"
SYS28 = Ø; SYS29 = Ø
Do a = 1 to SYS2cnt
  Parse Upper Var SYS2_dasd_list.a bkadd bkvol bkdev bksuite
  If bkdev = "338Ø" Then                /* For 338Øs...            */
    SYS28 = SYS28 + 1
  If bkdev = "339Ø" Then                /* For 339Ø...             */
    SYS29 = SYS29 + 1
  lin1 = " "bkadd"     "bkvol"     "bkdev"          "bksuite
```

```
  "EXECIO 1 DISKW DASDLSTC (Stem lin"    /* Print details            */
End
"EXECIO 1 DISKW DASDLSTC (Stem blank"
lin1 = " 338Øs = "SYS28"   339Øs = "SYS29
"EXECIO 1 DISKW DASDLSTC (Stem lin"
/* ---------------------------------------------------------- */
/*  MVSSYS3 disks...                                          */
/* ---------------------------------------------------------- */
xyz1 = "1 Production System 3 Disks Taken to BKUPSITE (MVSSYS3): "today
"EXECIO 1 DISKW DASDLSTP (Stem xyz"
xyz1 = "+ Production System 3 Disks Taken to BKUPSITE (MVSSYS3): "today
"EXECIO 1 DISKW DASDLSTP (Stem xyz"
"EXECIO 1 DISKW DASDLSTP (Stem xyz"
SYS38 = Ø; SYS39 = Ø
Do a = 1 to SYS3cnt
  Parse Upper Var SYS3_dasd_list.a bkadd bkvol bkdev bksuite
  If bkdev = "338Ø" Then                /* For 338Øs...             */
    SYS38 = SYS38 + 1
  If bkdev = "339Ø" Then                /* For 339Ø...              */
    SYS39 = SYS39 + 1
  lin1 = " "bkadd"     "bkvol"     "bkdev"          "bksuite
  "EXECIO 1 DISKW DASDLSTP (Stem lin"    /* Print details            */
End

"EXECIO 1 DISKW DASDLSTP (Stem blank"
lin1 = " 338Øs = "SYS38"   339Øs = "SYS39
"EXECIO 1 DISKW DASDLSTP (Stem lin"
/* ---------------------------------------------------------- */
/*  ALL disks...                                              */
/* ---------------------------------------------------------- */
xyz1 = "1 ALL your site name DISKS TAKEN TO BKUPSITE: "today
"EXECIO 1 DISKW DASDALL (Stem xyz"
xyz1 = "+ ALL your site name DISKS TAKEN TO BKUPSITE: "today
"EXECIO 1 DISKW DASDALL (Stem xyz"
"EXECIO 1 DISKW DASDALL (Stem xyz"
all8 = Ø; all9 = Ø
Do a = 1 to comdbkp2.Ø
  Parse Upper Var comdbkp2.a bkvol bkadd bkdev bksuite bkcom . . bkshr .
  /* Make sure that suites that don't go to BKUPSITE aren't included */
  /* in the 'FULL' listing that's created...                       */
  bit = Right(bksuite,1)                /* Get suite suffix         */
  found = "N"
  xx = Wordpos(bit,SYS1_backups)        /* SYS1 back-ups...         */
  If xx ¬= Ø Then Do
    found = "Y"
    bkup = Subword(SYS1_restore,xx,1)   /* Suite goes to BKUPSITE?  */
    If bkup = "N" Then                  /* ...no - skip it          */
      Iterate
  End
  xx = Wordpos(bit,SYS2_backups)        /* SYS2 back-ups...         */
  If xx ¬= Ø Then Do
```

```
    found = "Y"
    bkup = Subword(SYS2_restore,xx,1)     /* Suite goes to BKUPSITE? */
    If bkup = "N" Then                    /* ...no - skip it         */
      Iterate
  End
  xx = Wordpos(bit,SYS3_backups)          /* SYS3 back-ups...        */
  If xx ¬= Ø Then Do
    found = "Y"
    bkup = Subword(SYS3_restore,xx,1)     /* Suite goes to BKUPSITE? */
    If bkup = "N" Then                    /* ...no - skip it         */
      Iterate
  End
  If found = "N" Then                     /* Not SYS1, SYS2 or SYS3, */
    Iterate                               /* ...so ignore...         */
  If bkdev = "338Ø" Then                 /* For 338Øs...            */
    all8 = all8 + 1
  If bkdev = "339Ø" Then                 /* For 339Ø...             */
    all9 = all9 + 1
  xx = " "
  If bkcom = "X" Then
    xx = "     Not normally restored..."
  If bkshr = "Y" Then
    xx = "     PLEASE MARK AS *SHARED*"
  lin1 = " "bkadd"     "bkvol"     "bkdev"          "bksuite"     "xx
  "EXECIO 1 DISKW DASDALL (Stem lin"     /* Print details           */
End
"EXECIO 1 DISKW DASDALL (Stem blank"
lin1 = " 338Øs = "all8"   339Øs = "all9
"EXECIO 1 DISKW DASDALL (Stem lin"
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
GET_CONTROL:
/* ========================================================== */
/*  Read in the control record for each requested system ID.  */
/*  The control record shows what back-ups are run on each    */
/*  system.                                                   */
/* ========================================================== */
ctlok = "Y"
ctl = contrl"(@"sys")"
this_systems_backups = ""
Address "TSO"
ctl = contrl"(@"sys")"
"ALLOC FI(TEMP1) DA('"ctl"') SHR"
"EXECIO * DISKR TEMP1 (Stem okbkups. FINIS"
"FREE FI(TEMP1)"
If okbkups.Ø = Ø Then Do
  Say ">>>>>"
  Say ">>>>> CONTROL MEMBER '@"sys"' NOT FOUND..."
  Say ">>>>>"
  ctlok = "N"                             /* End...                  */
End
```

```
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
PRINT_LINE:
/* ============================================================= */
/*  Move the volid to the printline. If the printline is full   */
/*  (1Ø volids across the page) print the line...               */
/* ============================================================= */
prtlin1 = prtlin1||bkadd"   "            /* Move volid to prtline    */
slots = slots + 1                        /* Entries across line      */
If slots = 1Ø Then Do                    /* Line full?               */
  "EXECIO 1 DISKW DASDLST (Stem prtlin"  /* ...write it out          */
  slots = Ø                              /* ...reset count           */
  prtlin1 = " "                          /* ...reset line            */
End
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
PRINT_HDNG:
/* ============================================================= */
/*  Print a heading for requested device type (5 blank lines,   */
/*  a heading line and another blank line)...                   */
/* ============================================================= */
head1 = "                                "hdng
"EXECIO 1 DISKW DASDLST (Stem blank"
"EXECIO 1 DISKW DASDLST (Stem blank"
"EXECIO 1 DISKW DASDLST (Stem blank"
"EXECIO 1 DISKW DASDLST (Stem blank"
"EXECIO 1 DISKW DASDLST (Stem blank"
"EXECIO 1 DISKW DASDLST (Stem head"
"EXECIO 1 DISKW DASDLST (Stem blank"
Return
```

GENLIST1

```
/* ---------------------------------------------------- */
/* REXX "GENLIST1"                                       */
/* Generate listings for BKUPSITE...                     */
/* ---------------------------------------------------- */
/* ============================================================ */
/*           Read the output from a 'LISTC LVL(SYS8) GDG'       */
/*           request and produce a list of GDG Base entries.    */
/*           This info is passed to "GENLIST2".                 */
/*           Optional parm: suite name (eg 'BCKMVS1') if        */
/*           only a subset is required.                         */
/* ============================================================ */
Address "TSO"
Parse Upper Arg suite .
/* ------------------------------------------------------------ */
/*  Read in the entire listing...                               */
/* ------------------------------------------------------------ */
"EXECIO * DISKR LISTING (Stem listc. FINIS"
```

```
/* ---------------------------------------------------------- */
/*  Process the entries, extracting out just the GDG Bases.     */
/*  For each of these, call "LASTGEN" to determine the latest   */
/*  generation; write out an explicit "LISTC" command for this. */
/* ---------------------------------------------------------- */
Do a = 1 to listc.Ø
  If Substr(listc.a,2,11) = "GDG BASE --" Then Do  /* GDG BAse entry */
    dsn = Substr(listc.a,18,24)                    /* Extract dsn    */
    If suite <> "" Then Do                         /* Suite specified*/
      If Substr(listc.a,28,7) <> suite Then        /* If no match... */
        Iterate                                    /* ...then skip   */
    End
    "%LASTGEN "dsn
    lastcc = rc
    If lastcc = 55555 Then                         /* RC 55555 means */
      Iterate                                      /* no generations */
    goovoo = ".G"Right("ØØØ"lastcc,4)"VØØ"         /* Generation no. */
    dsn = dsn||goovoo
    Queue " LISTC ENT("dsn") VOL"                  /* Queue the info */
    "EXECIO 1 DISKW OUTPUT"                        /* ...write it out*/
  End
End
Return
```

BKPS

```
/* ---------------------------------------------------------- */
/* REXX "BKPS"                                                */
/* Generate listings for BKUPSITE...                          */
/* ---------------------------------------------------------- */
/* ========================================================== */
/*           Read the output "GENLIST1" (this is output from  */
/*           an IDCAMS "LISTC ENT(...) VOL" request for each  */
/*           dataset we want to report on. The output is:     */
/*           1/ A list of back-up GDG Base names and their    */
/*              associated volumes. There will be carriage    */
/*              control in column 1 and each 'new' suite      */
/*              will start on  a new page.                    */
/*           2/ A list of just the required cartridge numbers. */
/*              This will be sorted and read in by 'GENLIST3' */
/*              to create a) a listing with 1Ø volids across  */
/*              the page and b) a listing to be used by the   */
/*              'Eject cartridges' option.                    */
/* ========================================================== */
Address "TSO"
/* ---------------------------------------------------------- */
/*  Read in the entire listing...                             */
/* ---------------------------------------------------------- */
listc. = ""
last_suite = ""
```

```
gdg_flag = ""
"EXECIO * DISKR LISTING (Stem listc. FINIS"
Do a = 1 to listc.Ø
  If Substr(listc.a,2,12) = "GDG BASE --" Then Do
    gdg_flag = "Y"
    Iterate
  End
  If Substr(listc.a,2,2Ø) = "NONVSAM ------ SYS8" Then Do
    gdg_flag = "N"
    xx = Substr(listc.a,18,33)
    ww = xx
    Call WRITE_HDR
    Iterate
  End
  If Substr(listc.a,38,9) = "CREATION-" Then Do
    If gdg_flag = "Y" Then
      Iterate           /* Don't print details for GDG Base entries */
    xx = "   "Substr(listc.a,38,24)
    Call WRITE_RCD
    Iterate
  End
  If Substr(listc.a,9,9) = "VOLSER--" Then Do
    yy = Substr(listc.a,27,6)
    xx = "   "yy
    If DATATYPE(yy) <> "NUM" Then Do
      Say ">>>-----------------------------------------"
      Say ">>> Invalid volid detected: "yy
      Say ">>> DSN was:" ww
      Say ">>>          Record DROPPED!"
      Say ">>>-----------------------------------------"
      "SEND '==> GENLIST2 has detected an invalid volid: "yy"',
            U(BQIBIØ6)"
    End
    Else Do
      Call WRITE_RCD
      Call WRITE_VOL
    End
    Iterate
  End
End
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
WRITE_HDR:
cc = "Ø"                                  /* Default dbl space    */
Suite = Left(xx,17)                       /* Extract 'suite' bit  */
If suite <> last_suite Then Do            /* New suite = new page */
  cc = "1"
  last_suite = suite
End
Queue cc||xx
"EXECIO 1 DISKW OUTPUT"                   /* Write suite name     */
```

```
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
WRITE_RCD:
cc = " "                                  /* Single space         */
Queue cc||xx
"EXECIO 1 DISKW OUTPUT"                   /* Write listing data   */
Return
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
WRITE_VOL:
Queue xx
"EXECIO 1 DISKW VOLIDS"                   /* Write volid          */
Return
```

GENLIST3

```
/* ------------------------------------------------------- */
/* REXX "GENLIST3"                                          */
/* Generate listings for BKUPSITE...                        */
/* ------------------------------------------------------- */
/* ======================================================= */
/*           Read a list of volids (sorted into order) and  */
/*           create a listing of them, 1Ø across the page.  */
/*           Also create a list of carts that can be used as*/
/*           input into an 'SLUADMIN' job to eject the carts.*/
/*           The routine MAY be passed a suite name, which, if*/
/*           present is used in the listing header, and a CAPid*/
/*           to be placed in the eject statements.          */
/* ======================================================= */
/* Trace ir */
Address "TSO"
Parse Upper Arg suite cap .
hdr = "ALL"
If suite <> "" Then
  hdr = suite
capid = "ØØØ:ØØ"
If cap <> "" Then
  capid = cap
/* ------------------------------------------------------- */
/*  Read in the entire list of volids...                   */
/* ------------------------------------------------------- */
vols. = ""
vols_this_line = Ø
totvol = Ø
dt = Date()" "Time()
Queue "1     List of "hdr" cartridges for BKUPSITE   (created "dt")"
Queue "+     List of "hdr" cartridges for BKUPSITE   (created "dt")"
Queue "+     List of "hdr" cartridges for BKUPSITE   (created "dt")"
"EXECIO * DISKR VOLIDS (Stem vols. FINIS"
/* ------------------------------------------------------- */
/*  Print them 1Ø-across-the-page...                       */
```

```
/* ---------------------------------------------------------- */
line = "Ø"                                 /* Start with a double space */
Do a = 1 to vols.Ø
  totvol = totvol + 1
  vol = Strip(vols.a)
  line = line||vol"  "
  vols_this_line = vols_this_line + 1
  If vols_this_line = 1Ø Then Do
    Queue line
    line = " "
    vols_this_line = Ø
  End
End
If vols_this_line <> Ø Then
  Queue line
"EXECIO * DISKW OUTPUT"                        /* Write out listing    */
Queue "ØTotal carts: "totvol
"EXECIO * DISKW OUTPUT (FINIS"                 /* Write out total      */
/* ---------------------------------------------------------- */
/*  Now create the eject jobs... (one for each 'range' of carts */
/*  we find).                                                   */
/* ---------------------------------------------------------- */
Queue "  EJECT VOLSER( -"
"EXECIO 1 DISKW EJECTS"                        /* Write out header     */
Do a = 1 to vols.Ø
  vol = Strip(vols.a)
  If a <> vols.Ø Then
    vol = "    "vol", -"
  Else
    vol = "    "vol")   CAP("capid")"
  Queue vol
End
"EXECIO * DISKW EJECTS"                        /* Write out volume recs*/
Return
```

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

Grant Carson
Systems Programmer (UK)

DFHSM back-up control dataset audit routine – part 2

This month we conclude the program, which has been designed to produce a short audit report of the HSM back-up control dataset.


```
         TITLE 'BCDSINVT - PRINT LINE SUBROUTINE'
*----+----+----+----+----+----+----+----+----+----+----+---+----+----*
* $EDTPL IS A SIMPLE SUBROUTINE THAT CAN BE USED TO MOVE DATA TO A    *
* PRINT BUFFER, AND THEN OUTPUT THE BUFFER.                           *
* PLIST  +Ø  A(MESSAGE OR DATA TO PRINT)                              *
*        +4  A(BUFFER TO PUT THE MESSAGE IN)                          *
*        +8  A(DCB OF THE PRINT FILE)                                 *
*        +12 A(LINE COUNTER FIELD)                                    *
*        +16 A(MAXIMUM NUMBER OF LINES ON THE PAGE)                   *
* NOTES: BOTH THE MESSAGE OR DATA LINE TO BE PRINTED AND THE BUFFER   *
*        THAT IT USED TO PRINT FROM MUST BE LAID OUT IN THE FOLLOWING *
*        FORMAT.  THIS ROUTINE IS INCLUDED AT LINKAGE TIME            *
*                                                                     *
*        +Ø  2 BYTES CONTAINING THE LENGTH OF THE MESSAGE.            *
*        +2  1 BYTE FOOR THE CARRIAGE CONTROL.                        *
*        +3  THE MESSAGE OR DATA BEGINS HERE.                         *
*----+----+----+----+----+----+----+----+----+----+----+----+----+----*
$EDTPL   CSECT                        START THE CONTROL SECTION
$EDTPL   AMODE 31                     START THE CONTROL SECTION
$EDTPL   RMODE ANY                    START THE CONTROL SECTION
         BAKR  R14,Ø                  PUSH ONTO THE STACK
         LAE   R12,Ø(R15,Ø)           SET UP OUR BASE
         USING $EDTPL,R12             LET THE ASSEMBLER KNOW
         EREG  R1,R1                  REFRESH REGISTER 1
         LR    R2,R1                  GET CONTENTS
         L     R3,Ø(R2)               POINT TO DATA BUFFER
         L     R4,4(R2)               POINT AT THE PRINT BUFFER
         XR    R15,R15                CLEAR REG 15
         LR    R1,R15                 CLEAR REG 1
         ICM   R15,B'ØØ11',Ø(R3)      GET THE LENGTH
         ICM   R1,B'ØØ11',Ø(R4)       GET THE LENGTH
         CR    R15,R1                 COMPARE THE LENGTHS
         BH    $EDTPL1Ø               ERROR, EXIT ROUTINE
         ICM   R15,B'1ØØØ',=XL4'4ØØØØØØØ' MOVE IN THE PADDING VALUE
         LA    R14,2(,R3)             POINT TO THE MESSAGE
         LA    RØ,2(,R4)              POINT TO THE BUFFER
         MVCL  RØ,R14                 MOVE THE MESSAGE
         L     R3,12(R2)              POINT TO THE LINE COUNTER
         ICM   R4,B'1111',Ø(R3)       GET THE COUNTER VALUE
         LA    R4,1(,R4)              INCREMENT LINE COUNTER
         L     R5,16(R2)              POINT TO MAX LINES VALUE
         ICM   R5,B'1111',Ø(R5)       PUT MAX LINES VALUE IN R5
         CR    R4,R5                  Q. COUNTER = TO MAX?
         BNE   $EDTPLØ5               A. NO
         L     R5,4(R2)               POINT TO OUTPUT BUFFER
         MVI   2(R5),C'1'             MOVE IN CC FOR TOP OF PAGE
         LA    R4,1                   SET COUNTER TO 1
$EDTPLØ5 DS    ØH
         STCM  R4,B'1111',Ø(R3)       SAVE THE COUNTER
         L     R4,4(R2)               REFRESH POINTER TO PRINT BUF.
```

```
         LA    R15,2(,R4)              GET PAST THE LENGTH FIELD
         L     R14,8(R2)               PICK UP THE DCB ADDRESS
         PUT   (R14),(R15)
$EDTPL1Ø DS    ØH
         PR                            RETURN TO THE CALLER
         LTORG                         PLACE FOR THE LITERALS
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         DROP  R12                     TELL THE ASSEMBLER
         END   $EDTPL                  END THE CSECT
```

SOURCE FOR THE $ESAPRO MACRO

```
         MACRO
&LABEL   $ESAPRO &AM=31,&RM=ANY,&MODE=P
.*********************************************************************
.*       THIS MACRO WILL PROVIDE ENTRY LINKAGE AND OPTIONALLY
.*       MULTIPLE BASE REGISTERS.  TO USE THIS MACRO, YOU NEED TO
.*       ALSO USE THE $ESASTG MACRO.  THE $ESASTG DEFINES THE SYMBOL
.*       QLENGTH WHICH OCCURS IN THE CODE THAT &ESAPRO GENERATES.
.*       IF YOU DO NOT CODE ANY OPERANDS, THEN REGISTER 12 WILL BE
.*       USED AS THE BASE.  IF YOU CODE MULTIPLE SYMBOLS, THEN THEY
.*       WILL BE USED AS THE BASE REGISTERS.
.*       EXAMPLES:
.*              SECTNAME $ESAPRO          = REG 12 BASE
.*              SECTNAME $ESAPRO 5        = REG 5 BASE
.*              SECTNAME $ESAPRO R1Ø,R11  = REGS 1Ø AND 11 ARE BASES
.*********************************************************************
         LCLA  &AA,&AB,&AC
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
```

```
R6        EQU   6
R7        EQU   7
R8        EQU   8
R9        EQU   9
R1Ø       EQU   1Ø
RA        EQU   1Ø
R11       EQU   11
RB        EQU   11
R12       EQU   12
RC        EQU   12
R13       EQU   13
RD        EQU   13
R14       EQU   14
RE        EQU   14
R15       EQU   15
RF        EQU   15
          SPACE 1
ARØ       EQU   Ø
AR1       EQU   1
AR2       EQU   2
AR3       EQU   3
AR4       EQU   4
AR5       EQU   5
AR6       EQU   6
AR7       EQU   7
AR8       EQU   8
AR9       EQU   9
AR1Ø      EQU   1Ø
ARA       EQU   1Ø
AR11      EQU   11
ARB       EQU   11
AR12      EQU   12
ARC       EQU   12
AR13      EQU   13
ARD       EQU   13
AR14      EQU   14
ARE       EQU   14
AR15      EQU   15
ARF       EQU   15
*
FPRØ      EQU   Ø
FPR2      EQU   2
FPR4      EQU   4
FPR6      EQU   6
&LABEL    CSECT
&LABEL    AMODE &AM
&LABEL    RMODE &RM
          SYSSTATE ASCENV=&MODE          SET THE ENVIRONMENT
          B     $$$$EYEC-*(R15)          BRANCH AROUND EYECATCHER
          DC    AL1(($$$$EYEC-*)-1)      EYECATCHER LENGTH
```

```
         DC    CL8'&LABEL'             MODULE ID
         DC    CL3' - '
         DC    CL8'&SYSDATE'           ASSEMBLY DATE
         DC    CL3' - '
         DC    CL8'&SYSTIME'           ASSEMBLY TIME
         DC    CL3'   '                FILLER
$$$$F1SA DC    CL4'F1SA'               USED FOR STACK OPERATIONS
$$$$4Ø96 DC    F'4Ø96'                 USED TO ADJUST BASE REGS
$$$$EYEC DS    ØH
         BAKR  R14,Ø                   SAVE GPRS AND ARS ON THE STACK
         AIF   (N'&SYSLIST EQ Ø).USER12
         LAE   &SYSLIST(1),Ø(R15,Ø)    LOAD OUR BASE REG
         USING &LABEL,&SYSLIST(1)      LET THE ASSEMBLER KNOW
         AGO   .GNBASE
.USER12  ANOP
         MNOTE *,'NO BASE REG SPECIFIED, REGISTER 12 USED'
         LAE   R12,Ø(R15,Ø)            LOAD OUR BASE REG
         USING &LABEL,R12              LET THE ASSEMBLER KNOW
         AGO   .STGOB
.GNBASE  ANOP
         AIF   (N'&SYSLIST LE 1).STGOB
&AA      SETA  2
&AC      SETA  4Ø96
.GNBASE1 ANOP
*
         AIF   (&AA GT N'&SYSLIST).STGOB
&AB      SETA  &AA-1
         LR    &SYSLIST(&AA),&SYSLIST(&AB) GET INITIAL BASE
         A     &SYSLIST(&AA),$$$$4Ø96      ADJUST NEXT BASE
         USING &LABEL+&AC,&SYSLIST(&AA)    LET THE ASSEMBLER KNOW
&AA      SETA  &AA+1
&AC      SETA  &AC+4Ø96
         AGO   .GNBASE1
.STGOB   ANOP
         L     RØ,QLENGTH              GET THE DSECT LENGTH
         STORAGE OBTAIN,LENGTH=(RØ),LOC=(RES,ANY)
         LR    R15,R1                  GET @(OBTAINED AREA)
         L     R13,QDSECT              GET DISPLACEMENT INTO AREA
         LA    R13,Ø(R13,R15)          GET @(OBTAINED AREA)
         LR    RØ,R13                  SET REG Ø = REG 13
         L     R1,QLENGTH              GET THE LENGTH OF THE AREA
         XR    R15,R15                 CLEAR REG 5
         MVCL  RØ,R14                  INTIALIZE THE AREA
         MVC   4(4,R13),$$$$F1SA       INDICATE STACK USAGE
         USING DSECT,R13               INFORM ASSEMBLER OF BASE
.MEND    ANOP
         EREG  R1,R1                   RESTORE REGISTER 1
         MEND
```

SOURCE FOR THE $ESAEPI MACRO

```
         MACRO
         $ESAEPI
.*************************************************************************
.*       THIS MACRO WILL PROVIDE EXIT LINKAGE. IT WILL FREE THE
.*       STORAGE AREA THAT WAS ACQUIRED BY THE $ESAPRO MACRO.  YOU
.*       CAN OPTIONALLY PASS IT A RETURN CODE VALUE.  THIS VALUE IS
.*       EITHER THE LABEL OF A FULL WORD IN STORAGE, OR IT IS A REG-
.*       ISTER. AS WITH THE $ESAPRO MACRO, YOU NEED TO USE THE $ESASTG
.*       MACRO.  THE SYMBOL QLENGTH WHICH OCCURS IN THE CODE THAT IS
.*       GENERATED BY THIS MACRO IS DEFINED BY $ESASTG
.*       EXAMPLES:
.*              $ESAEPI          = NO RETURN CODE SPECIFIED
.*              $ESAEPI (R5)     = RETURN CODE IS IN REG 5
.*              $ESAEPI RETCODE  = RETURN CODE IS IN THE FULLWORD AT
.*                                 RETCODE
.*************************************************************************
         AIF   (N'&SYSLIST EQ Ø).STGFRE
         AIF   ('&SYSLIST(1)'(1,1) EQ '(').REGRC
         L     R2,&SYSLIST(1)          GET RETURN CODE VALUE
         AGO   .STGFRE
.REGRC   ANOP
         LR    R2,&SYSLIST(1,1)        GET RETURN CODE VALUE
.STGFRE  ANOP
         L     RØ,QLENGTH              GET THE DSECT LENGTH
         STORAGE RELEASE,LENGTH=(RØ),ADDR=(R13)
         AIF   (N'&SYSLIST NE Ø).SETRC
         XR    R15,R15                 CLEAR THE RETURN CODE
         AGO   .MEND
.SETRC   ANOP
         LR    R15,R2                  SET THE RETURN CODE
.MEND    ANOP
         PR                            RETURN TO CALLER
* FOR ADDRESSABILITY PURPOSES
         LTORG
         MEND
```

SOURCE FOR THE $ESASTG

```
         MACRO
         $ESASTG
.*************************************************************************
.*       THIS MACRO IS USED IN CONJUNCTION WITH THE $ESAEPI AND $ESAPRO
.*       MACROS.  IT PROVIDES A Q TYPE ADDRESS CONSTANT WHICH WILL CON-
.*       TAIN THE LENGTH OF THE DSECT.  A REGISTER SAVE AREA ID
.*       PROVIDED AS WELL.
.*       EXAMPLES:
.*              $ESASTG
```

```
.*      XXX     DC    F          = DEFINE ADDITIONAL STORAGE AREA
.*      YYY     DC    XL255
.*       .       .      .
.*       .       .      .
.*********************************************************************
##BIT0   EQU   128                     SET BIT 0 ON
##BIT1   EQU   64                      SET BIT 1 ON
##BIT2   EQU   32                      SET BIT 2 ON
##BIT3   EQU   16                      SET BIT 3 ON
##BIT4   EQU   8                       SET BIT 4 ON
##BIT5   EQU   4                       SET BIT 5 ON
##BIT6   EQU   2                       SET BIT 6 ON
##BIT7   EQU   1                       SET BIT 7 ON
        SPACE 1
RC0000   DC    F'0'                    USED TO SET RETURN CODES
RC0004   DC    F'4'                    USED TO SET RETURN CODES
RC0008   DC    F'8'                    USED TO SET RETURN CODES
RC000C   DC    F'12'                   USED TO SET RETURN CODES
RC0010   DC    F'16'                   USED TO SET RETURN CODES
        SPACE 1
QDSECT   DC    Q(DSECT)                DEFINE A QCON
QLENGTH  CXD                           LET ASM CALCULATE THE LENGTH
DSECT    DSECT
         DS    18F                     SET ASIDE REGISTER SAVE AREA
RET_CODE DS    F                       HOLDER FOR THE CURRENT RC
BASE_24  DS    F                       BASE ADDRESS FOR 24 BIT STORAGE
BASE_24L DS    F                       LENGTH OF 24 BIT STORAGE
BASE_24P DS    F                       SUBPOOL NUMBER
BASE_31  DS    F                       BASE ADDRESS FOR 31 BIT STORAGE
BASE_31L DS    F                       LENGTH OF 31 BIT STORAGE
BASE_31P DS    F                       SUBPOOL NUMBER
         MEND
```

SOURCE FOR THE $EDTML MACRO

```
         MACRO
.*********************************************************************
.*       THIS MACRO IS DESIGNED TO BE USED WITH A STANDARD MESSAGES   *
.*       CSECT.  YOU PROVIDE THE MESSAGE NUMBER THAT YOU WANT TO LO-  *
.*       CATE, AND THE MACRO WILL RETURN THE ADDRESS OF THE MESSAGE   *
.*       IF IT IS IN THE TABLE.                                       *
.*       INPUT: CONSISTS OF THREE PARAMETERS, THE MESSAGE NUMBER, THE *
.*              REGISTER OR FULLWORD TO PLACE THE MESSAGE ADDRESS     *
.*              INTO, AND THE ADDRESS OF THE MESSAGE TABLE CSECT.  IF *
.*              THE MESSAGE IS NOT FOUND IN THE TABLE, HIGH VALUES    *
.*              ARE RETURNED.                                         *
.*                                                                    *
.*       EXAMPLE: $EDTML   MSG#,(REGISTER),MSG. TABLE CSECT ADDR      *
.*       EXAMPLE: $EDTML   MSG#,FIELD,MSG. TABLE CSECT ADDR           *
```

```
.*                                                                   *
.*********************************************************************
         $EDTML
         LCLC  &LBL1,&LBL2,&LBL3
         LCLC  &MSGNO,&RVAL,&MSGTBL
.*********************************************************************
.* SEE HOW MANY PARMS WE HAVE.  WE MUST HAVE EXACTLY THREE FIELDS TO  *
.* CONVERT THE DATE.                                                 *
.*********************************************************************
         AIF   (N'&SYSLIST EQ 3).MT4
         MNOTE 12,'$EDTML ERROR - YOU MUST PROVIDE THREE PARAMETERS'
         AGO   .MEND
.MT4     ANOP
.*********************************************************************
.* GO AHEAD AND CREATE THE LABELS WE WILL NEED.                      *
.*********************************************************************
&LBL1    SETC  'LB1'.'&SYSNDX'
&LBL2    SETC  'LB2'.'&SYSNDX'
&LBL3    SETC  'LB3'.'&SYSNDX'
.*********************************************************************
.* PICK UP THE PARMS AND ASSIGN THEM TO LOCAL VARIABLES.             *
.*********************************************************************
&MSGNO   SETC  '&SYSLIST(1)'
&RVAL    SETC  '&SYSLIST(2)'
&MSGTBL  SETC  '&SYSLIST(3)'
         ICM   R1,B'1111',&MSGTBL       GET ADDRESS OF MESSAGES
         ICM   R14,B'1111',Ø(R1)        GET ENTRY SIZE
         ICM   R15,B'1111',4(R1)        GET ADDRESS OF MESSAGE TABLE
         LA    R1,8(,R1)                POINT AT FIRST MESSAGE
         ICM   RØ,B'1111',=XL4'FFFFFFFF' SET TO HIGH VALUES
&LBL1    DS    ØH
         CLC   Ø(1,R1),=AL1(&MSGNO)     CHECK MESSAGE NUMBER
         BE    &LBL2                    FOUND THE MESSAGE
         LA    R1,Ø(R14,R1)             BUMP THE ADDRESS
         BCT   R15,&LBL1                DECREMENT THROUGH ALL MESSAGES
         B     &LBL3                    ERROR, MESSAGE NOT IN TABLE
&LBL2    DS    ØH
         L     R1,1(R1)                 BUMP IT UP BY ONE
         LR    RØ,R1                    POINT TO THE MESSAGE
&LBL3    DS    ØH
         AIF   ('&RVAL'(1,1) EQ '(').MT5
         STCM  RØ,B'1111',&RVAL         PUT ADDR. IN REQUESTED AREA
         AGO   .MEND
.MT5     ANOP
&RVAL    SETC  '&RVAL'(2,K'&RVAL-2)
         LR    &RVAL,RØ                 PUT ADDR. IN REQUESTED REG.
.MEND    ANOP
         MEND                     EXIT
```

MVS news

IBM has announced z/OS V1.4, which is said to increase scalability, may reduce administration costs, and simplifies configuration with support for IPv6.

It allows less error-prone management of z/OS Unix identities for users and groups, extends the value of msys for Setup with self-configuring advances and exploitation enhancements to the msys for Setup framework, and there's assistance with OS/390 to z/OS migration.

z/OS V1.4 makes use of eLiza technologies to self-optimize sysplex performance with Workload Manager (WLM) balancing of batch initiators across systems in a sysplex, achieve more granular performance reporting with better self-optimization of WebSphere Application Server, and enhance Security Server PKI, which gives an improved Digital Certificate Management solution on the z/OS platform.

There are more self-configuring capabilities with new Web-based wizards for z/OS Intelligent Resource Director and IBM eServer Security Planner and it's said to be easier to add systems to a sysplex in JES3 environments.

The new release has more tools, including those which simplify configuration, renumbering support, and application compatibility with new IPv6 support; enable clock synchronization between clients and servers with a new TCP/IP daemon supporting SNTP; simplify configuration and improve diagnosis capability and serviceability in SNA networks with Enterprise Extender (EE) and SNA enhancements; and provide additional configuration and definitional flexibility with TN3270 enhancements.

Security has been enhanced with improved cryptographic services through SSL and Security Server LDAP and firewall technologies.

Application support is improved with the ability to decompose or compose code that comes from another code page using Unicode Normalization Service, there is increased flexibility and reliability in a sysplex with distributed Byte Range Lock Manager (BRLM), and there are additional recovery options for choosing which system takes over file system mounts when the current mount owner leaves the sysplex.

For further information contact your local IBM representative.
URL: http://www.ibm.com/servers/eserver/zseries/zos/downloads/.

* * *

SDS has announced Version 5.1.0 of Vital Signs VisionNet (VSV), its TCP/IP network performance monitor for OS/390 and z/OS.

The new release provides Web browser access to performance data for every resource on a network. The VSV Web Server gets performance data from the mainframe database and delivers it, via secure intranet, to standard Web browsers.

VSV's browser-based graphic interface provides summary and detailed performance reports for TCP/IP, telnet, FTP, sockets, CSM, VTAM, and NCP.

For further information contact:
Software Diversified Services, 5155 East River Road, Suite 411, Minneapolis, MN 55421-1025, USA.
Tel: (763) 571 9000.