161

MVS update

February 2000

In this issue

MVS Update

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

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

Contributions

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

Editor
Jaime Kaminski

Disclaimer

MVS Update on-line

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

Subscriptions and back-issues

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

COBOL coding efficiency enhancements

INTRODUCTION

Over many years, I have seen areas in COBOL coding that could always benefit from some coding improvements. I have compiled a list of various hints and tips to improve both performance and readability of COBOL code. I have divided them by the appropriate COBOL program division, attempting to indicate the reason for their appropriateness. It is by no means an exhaustive or definitive list, rather includes only those tips that have constantly proven useful to me when assisting in the tuning of COBOL code. You can see that most of the suggestions are for the DATA and PROCEDURE divisions; these are usually where the most damage is done, and is consequently where the most improvement can be made.

As a standard caveat, your results may vary when implementing any of these suggestions. The differences could be attributable to the COBOL compiler (OS/VS, VS/COBOL II, COBOL for MVS and VM, COBOL/390, etc), by changes introduced in the application of standard IBM product maintenance, by the choice of compiler options, as well as by the installations compiler default options. Tips related to readability are strictly a matter of style; whatever standards for style you might choose to adopt are fine as long as those standards are adhered to uniformly. For example, your installation might choose not to allow the use of the DISPLAY verb, opting instead for WRITE statements to a file; you might prefer structured programming and thus disallow the use of either GO TO or GO TO DEPENDING ON statements.

These are very valid choices and it is not the intention of this list to contradict installation standards; if you already have standards in place, they are there because they were deemed the proper selection for the time and circumstance. Please do not take this article to your systems programmer and say “see, it says here we should...”. There is nothing more annoying to a good systems programmer than someone asking to change something because they read it in print somewhere, unless it makes good business sense to change as a result of the circumstances changing.

ENVIRONMENT DIVISION

1 Use the FILE STATUS option on the SELECT statement in the FILE-CONTROL paragraph of the INPUT-OUTPUT section to establish a status key for all I/O operations. For VSAM, specify a second status area for more detailed information, as follows:

```
SELECT INPUT-FILE
ASSIGN TO INDD
FILE STATUS IS FS-CODE VSAM-CODE.
```

and in working storage, use the following:

```
Ø3 FS-CODE         PIC X(2).
Ø3 VSAM-CODE
   Ø5 VSAM-RETCODE  PIC 9(2) COMP.
   Ø5 VSAM-COMPCODE PIC 9(1) COMP.
   Ø5 VSAM-RSNCODE  PIC 9(1) COMP.
```

2 It is preferable to create fixed-length record QSAM files rather than variable-length record QSAM files. If variable-length record QSAM files must be created, use the APPLY WRITE-ONLY clause in the I-O-CONTROL paragraph of the INPUT-OUTPUT section (or use the compiler option AWO). This will allow each output file buffer to be filled as much as possible rather than truncating the buffer when the space remaining in the buffer is smaller than the maximum record size for the file.

3 When designing VSAM KSDS files, design the key (defined by the RECORD KEY IS clause of the INPUT-OUTPUT section) within the records so that the high-order portion of the key remains relatively constant while the low-order portion varies more often. This allows for better key compression by VSAM. For example, a key of state code followed by an individual's account number would be better than an account number followed by state code.

DATA DIVISION

1 Code BLOCK CONTAINS 0 RECORDS in FD statements to allow the system to determine the blocksize at allocation time. It will use the optimal blocksize for the device the data is being stored on.

2 When coding tables in a program, accessing table entries is achieved most efficiently by using the methods below in order of descending efficiency:

– Use a literal, as follows:

```
IF ENTRY (5) = Ø THEN GO TO SEARCH-EXIT.
```

– Use an index. Index values contain the direct address of table elements. For example:

```
Ø1 TBL-INDEX SYNC USAGE IS INDEX or
Ø3 ENTRY OCCURS 1Ø TIMES INDEXED BY TBL-INDEX
```

followed by table references, as follows:

```
IF ENTRY (TBL-INDEX) = Ø THEN GO TO SEARCH-EXIT
```

– Use a subscript, which should be defined as COMP SYNC with eight or fewer digits. Subscripts contain only offsets to table elements, which must then be converted to addresses each time a table element is referenced. For example:

```
Ø3 TBL-SUB PIC S9(5) COMP SYNC.
```

When using a subscript (on tables containing less than 50 items):

– Define your table so the subscript that varies most is rightmost in the expression (with multi-level tables).

– Order the table elements in descending order of frequency of occurrence to speed up searches.

4 Group heavily accessed data items together and place them at the beginning of the WORKING STORAGE section. If you have a large amount amount of working storage, this will tend to keep the most frequently used areas resident in memory.

5 For numeric data items, whenever possible:

– Code BINARY or COMP for numeric fields, because this will allow for the most efficient arithmetic calculations. A 1-4 digit field will be require two bytes of storage, a 5-9 digit field will be require four bytes, and a 10-18 digit field will be required eight bytes, for example:

```
Ø3 DATA PIC S9(5) COMP.
```

- Code SYNC for BINARY or COMP items to align them on their proper boundaries to improve processing. This also aids in the determination of the length of records and group items, for example:

  ```
  Ø3 DATA PIC S9(5) COMP SYNC.
  ```

  A 1-4 digit field will be aligned on a halfword boundary, a 5-9 digit field will be aligned on a fullword boundary, and a 10-18 digit field will be aligned on a doubleword boundary.

- Code PACKED-DECIMAL or COMP-3 for numeric fields less than or equal to 15 digits, and define the PICTURE clause using an odd number of digits, as follows:

  ```
  Ø3 DATA PIC S9(9) COMP-3.
  ```

- Avoid using zoned decimal items when performing arithmetic, because the compiler will then need to convert these to (at least) packed decimal representation in order to perform computation, as follows:

  ```
  Ø3 DATA PIC S9(5) or
  Ø3 DATA PIC S9(5) USAGE IS DISPLAY.
  ```

- Perform arithmetic using the same types of fields (all COMP or COMP-3) and define fields with PICTURE clauses containing the same number of decimal digits; if a field is input in a different format, move the input field to a new numeric field with the same type (COMP or COMP-3) before using it for arithmetic.

- Code signs on all numeric fields, as follows:

  ```
  Ø3 DATA PIC S9(5).
  ```

- If the item will not be used in arithmetic, define the item as a DISPLAY item rather than numeric.

5 Data items used as indicators or switches should be defined as one byte fields. They should be defined as elementary items under a higher group item since each 01 level elementary item is aligned on a doubleword boundary, as follows:

```
Ø1 SWITCHES.
   Ø3 FOUND-DATA PIC X.
   Ø3 EOF-INPOUT PIC X.
```

6 Define items in WORKING-STORAGE with VALUE clauses to avoid encountering uninitialized fields, as well as to avoid the need to use the MOVE or INITIALIZE statements in the PROCEDURE division. This can also be done for table elements in VS/COBOL II and higher rather than redefining the table or using a PERFORM loop to initialize all of the table elements (see PROCEDURE DIVISION item 3 for more information).

7 When using the OCCURS DEPENDING ON (ODO) clause, define the ODO object (the field specified after ODO) as a COMP field, as follows:

```
Ø3 DATA-SECTIONS OCCURS Ø TO 5 TIMES DEPENDING IN NUM-SECTS
```

and in working storage, use the following:

```
Ø3 NUM-SECTS        PIC S9(5) COMP SYNC.
```

PROCEDURE DIVISION

1 Avoid using the ROUNDED and ON SIZE ERROR phrases on arithmetic statements.

2 Use the COMPUTE statement instead of the ADD, SUBTRACT, MULTIPLY, or DIVIDE statements when doing several arithmetic operations.

3 Avoid using the INITIALIZE statement; instead see DATA DIVISION item 7, or if necessary, use MOVE statements. If it is necessary to initialize a table via MOVE statements, initialize the first entry and perform a loop to MOVE ENTRY (1) to each subsequent entry, as follows:

```
MOVE ZEROS TO ENTRY (1).
PERFORM VARYING INDEX FROM 2 BY 1 UNTIL INDEX = n
MOVE ENTRY (1) TO ENTRY (INDEX)
END-PERFORM.
```

4 For tables containing less than 50 items, it may be more efficient to check against the current table item being pointed to before searching the table again. If the table is small, also consider ordering the table entries from the most frequently occurring element to the least. If the table is large, keep the entries in some key order and use the SEARCH ALL statement to perform a binary search.

5 The best possible table access is direct access, such as when you have a numeric month between 01 and 12, using that value to access the corresponding table entry, as below:

```
Ø1 MONTH-TABLE
   Ø3 MONTH-VALUE OCCURS 12 TIMES.
      Ø5      PIC X(8) VALUE 'JANUARY'.
      ...
      Ø5      PIC X(8) VALUE 'DECEMBER'.
```

and in the PROCEDURE division, use the following:

```
IF INPUT-REC-MONTH IS GREATER TAN  Ø AND LESS THAN 13
   MOVE MONTH-VALUE (INPUT-REC-MONTH) TO REPORT-MONTH.
```

See also item 12 below for information about checking for the valid range of index and subscript values.

5 In nested IF statements, order the test elements in descending order of frequency of occurrence to speed up searches. Preferably, use the EVALUATE statement instead to reduce coding required as well as to enhance clarity, as follows:

```
EVALUATE SEX
  WHEN 'M' PERFORM MALE-PROCESS
  WHEN 'F' PERFORM FEMALE-PROCESS
  WHEN OTHER PERFORM NO-SEX-GIVEN-PROCESS
END-EVALUATE.
```

Note that more than one element can be evaluated on the same statement, such as:

```
EVALUATE SEX ALSO AGE-RANGE
  WHEN 'M' ALSO TWENTY-RANGE PERFORM MALE-2Ø-PROCESS
  WHEN 'F' ALSO TWENTY-RANGE PERFORM FEMALE-2Ø-PROCESS
  WHEN 'M' ALSO THIRTY-RANGE PERFORM MALE-3Ø-PROCESS
  ...
  WHEN OTHER PERFORM NOT-CLASSIFIED-PROCESS
END-EVALUATE.
```

Again, order the test elements in descending order of frequency of occurrence to speed up searches.

6 Use a single OPEN statement to open multiple files; use a single CLOSE statement to close multiple files. This also has the advantage of writing an end-of-file mark in otherwise empty output files. If you have defined file status fields for your files, however, it might be easier to code separate OPEN and CLOSE statements to check their status after each operation. In either

case, you should also check the file status field after READ and WRITE statements for acceptable values.

7 For numeric field MOVEs, define the receiving field type, size and sign equal to the sending field. For character field MOVEs, define the receiving field size equal to the sending field.

8 Use the static form of the CALL statement instead of the dynamic form where possible, as follows:

```
CALL 'SUBPROG' USING PARM-FIELD.
```

Use the compiler option NODYNAM instead of DYNAM. This reduces overhead by assuming that subprograms invoked have been statically linked into the COBOL run unit. Use the RETURN-CODE special register to check the return code from any called subprogram as well as to set a return code for your program prior to issuing a GOBACK.

9 Analyze loops to keep them to the minimum number of instructions. For example, when applying a discount to a group of summed data, sum the data first and then apply the discount rather than applying the discount to each item and then summing the result.

10 Use in-line PERFORM statements followed by END-PERFORM statements when a paragraph is PERFORMed from one place, rather than out-of-line, to allow the COBOL optimizer to optimize code more efficiently.

11 On IF statements, check for continuous ranges of values instead of checking for each value individually, or if a field is greater than one character use condition-names, as follows:

```
IF LIT = 'A' THRU 'Z' ... or
88 COND1 VALUES ARE '1Ø' THRU '44'
IF COND1 THEN ...
```

12 Check for out-of-range conditions on indexes or subscripts before using them to address a table, or PERFORM a paragraph using the UNTIL option checking for the index or subscript exceeding the maximum number of table entries. For a multi-dimensional table, check all indexes or subscripts. It is probably a good documentation idea to code the maximum number of occurrences in an elementary item somewhere near the table

definition. This value can then be used for range checking, as follows:

```
Ø1 MONTH-TABLE
   Ø3 MONTH-RANGE PIC S9(5) COMP SYNC VALUE 12.
   Ø3 MONTH-VALUE OCCURS 12 TIMES.
      Ø5      PIC X(8) VALUE 'JANUARY'.
      ...
      Ø5      PIC X(8) VALUE 'DECEMBER'.
```

and in the PROCEDURE division, as follows:

```
IF INPUT-REC-MONTH IS NUMERIC AND
   INPUT-REC-MONTH IS LESS THAN MONTH-RANGE
      MOVE MONTH-VALUE (INPUT-REC-MONTH) TO REPORT-MONTH.
```

Do not use the compiler option SSRANGE for index or subscript checking except for development testing, because this will cause range checking to be performed every single time an index or subscript is referenced in your code.

13 When sorting data is required, an external JCL SORT is more efficient than a COBOL SORT statement. If data manipulation needs to be done along with sorting, A COBOL SORT statement might be considered. When using the COBOL SORT statement:

– Use either the USING or GIVING options (but not both) as well as the compiler option FASTSRT. This will allow the sort utility to perform at least some I/O operations, which it can do more efficiently than the QSAM I/O used by COBOL.

– Check the SORT-RETURN special register for the return code from the sort operation.

– Alternatively, see ‘Other Considerations’ below to execute a JCL sort using COBOL programs as sort exit routines.

14 When returning control, always use the GOBACK statement rather than the STOP RUN or EXIT PROGRAM statements. This will return control back to previous caller instead of terminating the entire COBOL run unit.

15 When branching to different routines based on the value of a numeric data item, use the GO TO DEPENDING ON statement rather than a series of IF or nested IF statements, as follows:

```
GO TO A B C D E F .... DEPENDING ON STATE-CODE.
GO TO NO-STATE-CODE-FOUND.
```

Preferably, use the EVALUATE statement (see item 16).

16 When branching to different routines based on the value of a character data item, use the EVALUATE statement rather than a series of IF or nested IF statements, as follows:

```
EVALUATE STATE-CODE
     WHEN 'NJ' PERFORM NJ-ROUTINE
     WHEN 'NY' PERFORM NY-ROUTINE
     WHEN OTHER PERFORM STANDARD-ROUTINE.
```

Code the WHEN clauses checking for the most common conditions first.

17 Use the WRITE statement rather than DISPLAY to output information, or use:

```
DISPLAY 'text' UPON name
```

where name is either SYSOUT or a name defined in the SPECIAL-NAMES paragraph of the CONFIGURATION section of the ENVIRONMENT division.

18 Check the validity of numeric data items before using them in arithmetic, as follows:

```
IF INPUT-DATA IS NUMERIC ...
```

Use the compiler option NUMPROC(PFD) rather than NUMPROC(NOPFD) or NUMPROC(MIG) so extra code will not be generated to fix invalid signs.

19 Check the validity of alphabetic data items, as well as for containing upper or lower case data, as follows:

```
IF INPUT-DATA IS ALPHABETIC ....
IF INPUT-DATA IS ALPHABETIC-UPPER ....
IF INPUT-DATA IS ALPHABETIC-LOWER ....
```

20 Check for the availability of COBOL provided intrinsic or callable functions before you attempt to code functions yourself. This can include arithmetic functions (such as square root, minimum/maximum/mean values, and trigonometric calculations), date and time functions (such as date/time conversion, day of week, and GMT time), as well as alphabetic functions (such as lowercase, reverse, and uppercase).

21 Use scope delimiters to improve readability of code wherever possible. These include IF/END-IF, PERFORM/END-PERFORM, SEARCH/END-SEARCH and EVALUATE/END-EVALUATE, to name a few.

OTHER CONSIDERATIONS

1 For highest sort performance, execute an external JCL sort and modify the sort processing by coding exit routines in COBOL. This will allow the sort utility to perform all the I/O operations instead of the QSAM I/O used by COBOL. While the SORT utility allows for several different exit routines, the two most common are called the E15 exit and the E35 exit. The E15 and the E35 exits are the only two SORT exits that can be written in COBOL.

An E15 exit is a COBOL program that gets control during the input phase of the SORT step, equivalent to a COBOL INPUT procedure. An E35 exit is a COBOL program that gets control during the output phase of the SORT step, equivalent to a COBOL OUTPUT procedure. In the example below, the COBOL E15 exit routine is called each time a record is read from the SORTIN file, and once at end-of-file. Communication between the SORT utility and the COBOL exit takes place in the LINKAGE SECTION of the COBOL program.

For fixed-length records, three fields must be defined in the LINKAGE SECTION. This example assumes that the record length of the SORTIN file is 100 bytes. Assuming the usual other required division and sections are present, the code would be as follows:

```
LINKAGE SECTION.
Ø1 EXIT-STATUS             PIC 9(8) COMP.
   88 FIRST-RECORD         VALUE Ø.
   88 NOT-FIRST-RECORD     VALUE 4.
   88 END-OF-FILE          VALUE 8.
Ø1 ORIGINAL-RECORD.
   Ø5 RECORD-TYPE          PIC X.
      88 GOOD-RECORD       VALUE '1'.
      88 CHANGE-RECORD     VALUE '2'.
      88 SKIP-RECORD       VALUE '3'.
   Ø5 FILLER               PIC X(99).
```

```
Ø1 MODIFIED-RECORD.
   Ø5 FILLER                PIC X(9Ø).
   Ø5 CHANGE-DATA           PIC X(1Ø).

 PROCEDURE DIVISION USING EXIT-STATUS
     ORIGINAL-RECORD MODIFIED-RECORD.

 IF END-OF-FILE
    DISPLAY 'END OF PROCESSING' UPON SYSOUT
    MOVE 8 TO RETURN-CODE
    GOBACK.

 IF FIRST-RECORD
    DISPLAY 'START OF PROCESSING' UPON SYSOUT.

 IF GOOD-RECORD
    MOVE Ø TO RETURN-CODE
 ELSE
    IF CHANGE-RECORD
       MOVE ORIGINAL-RECORD TO MODIFIED-RECORD
       MOVE 'NEW DATA' TO CHANGE-DATA
       MOVE 2Ø TO RETURN-CODE
    ELSE
       IF SKIP-RECORD
          MOVE 4 TO RETURN-CODE
       ELSE
          DISPLAY 'UNKNOWN RECORD TYPE' UPON SYSOUT
          MOVE 16 TO RETURN-CODE.

 GOBACK.
```

– Each time control is passed to the COBOL program, the SORT will pass a value in the EXIT-STATUS field; a value of 0 indicates that this is the first record in the SORTIN file, a value of 4 indicates that this is a record other than the first record in the SORTIN file, and a value of 8 indicates that we are at end-of-file in the SORTIN file.

– The record from the SORTIN file will be passed to the program in the field called ORIGINAL-RECORD. If the program needs to change the record before being sorted, it should place the changed record in the field called MODIFIED-RECORD.

– The program should always pass a value in RETURN-CODE back to the SORT (RETURN-CODE is a reserved word in COBOL and should not be defined in the program). The acceptable values for RETURN-CODE are:

0 – Accept the original record from SORTIN file.

4 – Delete the current record from the SORTIN file.

8 – Do not return to this exit routine (signifies end of the input phase).

12 – Insert a record into the SORT file ahead of the record which has just been passed (not illustrated in the program).

16 – End the SORT program with a completion code of 16 (signifies a serious error condition).

20 – Replace the current record with a record supplied by the exit.

– The JCL to invoke the sort and the COBOL exit would appear as follows:

```
//*-------------------------------------------------------*
//** EXAMPLE OF SORT JCL USING AN EXIT ROUTINE
//*-------------------------------------------------------*
//STEPØØ1  EXEC PGM=SORT,PARM=PRINT121
//EXITLIB  DD DISP=SHR,DSN=LOAD.LIB
//SORTIN   DD DISP=SHR,DSN=INPUT.FILE
//SORTOUT  DD DISP=OLD,DSN=OUTPUT.FILE
//SYSOUT   DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *
 SORT FIELDS=(192,3,A,4Ø,9,A),FORMAT=CH
 MODS E15=(COBXIT15,Ø,EXITLIB,C)
//
```

Refer to your sort utility manual for the definition and meanings of the fields on the MODS control statement.

Systems Programmer (USA) © Xephon 2000

SMP/E registered dataset archive utility

INTRODUCTION

In many shops that I have worked in, there are usually several systems programmers who at one time or another need to utilize SMP/E to apply maintenance to a system. As is usually the case when multiple people perform the same task, each person will do it somewhat differently. This can sometimes lead to problems with the integrity of the SMP/E datasets. We have developed a simple utility program that can be used to facilitate the save process for the datasets that are registered within the Consolidated Software Inventory (CSI). Since we are reading the CSI each time to determine which datasets must be backed up, we always pick-up any changes that may have occured in the CSI since the last back-up.

The program logic is fairly simple. We are passed the name of the base CSI. From it, we are looking for other CSI datasets that are defined, as well as all the DDDEF definitions that are in the CSI. From that, we generate the necessary dump control statements for DFSMSdss. You could generate the JCL and control statements for the utility that you prefer to use. We use the system dynamic allocation facilities to gain access to all of the CSI datasets.

I have also included the source for $ESAPRO, $ESAEPI, and $ESASTG. These are three macros that we use for general program development. The program was developed and executed in a DFSMS 1.3 environment.

ARKSMPDS

```
         TITLE 'A R K S M P D S — ARCHIVE SMP/E CONTROLLED DATASETS TO+
               TAPE'
*—+—+—+—+—+—+—+—+—+—+—+—+—+—*
*                                                                      *
* CSECT   : ARKSMPDS                                                   *
*                                                                      *
* MODULE  : ARKSMPDS                                                   *
*                                                                      *
* DESC    : ARKSMPDS IS A UTILITY PROGRAM THAT CAN BE USED TO CREATE  *
*           A BACKUP OF DATASETS THAT ARE DEFINED IN A SMP CSI        *
*           DATASET.  IT CAN BE USEFUL TO EXECUTE THIS UTILITY PRIOR  *
*           TO EXECUTING A SMP APPLY OR ACCEPT FUNCTION.              *
*                                                                     *
* MACROS  : $ESAPRO $ESAEPI $ESASTG ACB RPL OPEN CLOSE                *
* DSECTS  : DCBD                                                      *
* CALL(S) : N/A                                                       *
* INPUT   : N/A                                                       *
* OUTPUT  : DFDSS CONTROL STATEMENTS                                  *
* PLIST   : NAME OF THE BASE CSI                                      *
*                                                                     *
*---+---+---+---+---+---+---+---+---+---+---+---+---+---*
         EJECT
ARKSMPDS $ESAPRO R12,AM=31,RM=24
*
*---+---+---+---+---+---+---+---+---+---+---+---+---+---*
* MAKE SURE THAT WE WERE PASSED SOMETHING IN THE PARM FIELD           *
*---+---+---+---+---+---+---+---+---+---+---+---+---+---*
*
         MVC   RET_CODE,RC0010          PRESET RETURN CODE
         LR    R2,R1                    COPY PARM POINTER
         LTR   R2,R2                    Q. ANY PARMS PASSED
         BZ    LABL9999                 A. NO
         LR    R2,R1                    PICK UP THE PARM POINTER
         L     R2,0(R2)                 LET'S PICK THEM UP
         XC    RET_CODE,RET_CODE        CLEAR RET_CODE
         LH    R3,0(R2)                 PUT THE LENGTH IN REG 3
         STH   R3,TUNIT3+4              STORE IT IN TEXT UNIT
         BCTR  R3,0                     DECREMENT FOR THE MOVE
*
*---+---+---+---+---+---+---+---+---+---+---+---+---+---*
* MOVE THE CSI NAME FOR THE DYNAMIC ALLOCATE                          *
*---+---+---+---+---+---+---+---+---+---+---+---+---+---*
*
         B     *+10                     BRANCH AROUND
         MVC   TUNIT3+6(*-*),2(R2)      MOVE THE DSN
         EX    R3,*-6                   EXECUTE PREVIOUS INSTRUCTION
         BAL   R10,ALLOCCSI             GO ALLOCATE THE GLOBAL
         OPEN  (WORKFL1,(OUTPUT))
*
LABL0000 DS    0H
*
         MVC   DYNRPL(CSIRPLL),CSIRPL  MOVE IN MODEL RPL
         MVC   DYNXLS(CSIXLSL),CSIXLS  MOVE IN MODEL EXLST
         LA    R11,DYNRPL               GET @(RPL)
         USING IFGRPL,R11               LET ASSEMBLER KNOW
         LA    R14,DSNTAB               GET @(DLIB AND TARGET CSI'S TBL)
         LA    R15,44*25                GET THE LENGTH
         ICM   R1,B'1111',=XL4'40000000' PUT IN SPACE PAD
         MVCL  R14,R0                   NOW INIT THE TABLE
*
```

```
         LA    R1,CSIACB                GET @(ACB)
         ST    R1,RPLDACB               STORE IT IN RPL
         LA    R1,CSIREC                GET @(BUFFER AREA)
         ST    R1,RPLAREA               STORE IT IN RPL
         LA    R1,143                   GET SIZE OF BUFFER
         ST    R1,RPLBUFL               STORE IT IN RPL
         LA    R15,DYNXLS               GET ADDRESS OF EXIT LIST
         MODCB ACB=CSIACB,EXLST=(R15)
         OPEN  (CSIACB)
         LTR   R15,R15                  Q. OPEN UP OK
         BZ    LABL0075                 A. YES, PROCEED
*
* NEED CODE HERE FOR ERROR
*
LABL0075 DS    0H
*
         LA    R7,DSNTAB-FORTY6         POINT R7 BEFORE TABLE
         MVC   DSNADDRS+4,FULL46        STORE ENTRY SIZE
         LA    R8,DSNTAB                POINT R8 AT TABLE
         ST    R8,DSNADDRS              SAVE IT
         LA    R15,DYNXLS               GET ADDRESS OF DYNAMIC EXLIST
         MODCB EXLST=(R15),EODAD=LABL0300
*
LABL0100 DS    0H
*
         GET   RPL=(R11)
         LTR   R15,R15                  Q. READ OK
         BZ    LABL0200                 A. YES, PROCEED
*
* NEED CODE HERE FOR ERROR
*
LABL0200 DS    0H
         CLC   GBLCHK1,CSIREC           Q. ZONE TYPE RECORD
         BH    LABL0100                 A. NO, HAVN'T MADE IT TO ZONES
         BL    LABL0300                 A. NO, WE'RE PAST THE ZONES
*
LABL0210 DS    0H
*
         CLC   GBLCHK2,CSIREC+12        Q. FURTHER INVESTIGATION
         BNE   LABL0100                 A. WE DON'T WANT THIS RECORD
         CLC   GBLCHK3,CSIREC+24        Q. DLIB
         BE    LABL0225                 A. YES, WE'LL TAKE IT
         CLC   GBLCHK4,CSIREC+24        Q. TARGET
         BNE   LABL0100                 A. NO, ON TO NEXT RECORD
*
LABL0225 DS    0H
*
         MVC   OUTBUF,PRIMOBUF          PRIME UP THE OUTPUT BUFFER
         LA    R4,CSIREC                POINT AT INPUT BUFFER
         L     R5,RPLRLEN               GET THE LENGTH
         AR    R5,R4
         LA    R4,32(,R4)               POINT AT DATASET NAME
         SR    R5,R4                    COMPUTE THE LENGTH
         BCTR  R5,Ø                     DECREMENT FOR THE MVC
         EX    R5,MOVEDSN               GO MOVE THE DSN
         LA    R7,FORTY6(,R7)           INCREMENT IN THE TABLE
         ST    R7,DSNADDRS+8            SAVE ADDRESS
         STH   R5,Ø(R7)                 SAVE LENGTH
         EX    R5,SAVEDSN               SAVE DSN IN TABLE
         PUT   WORKFL1,OUTBUF
         B     LABLØ1ØØ
*
LABLØ3ØØ DS    ØH
*
         CLC   GBLCHK5,CSIREC           Q. IN THE ZONE WE WANT
         BL    LABL1ØØØ                 A. NO
         CLC   GBLCHK6,CSIREC+12        Q. IS IT A DDDEF
         BNE   LABLØ1ØØ                 A. NO
         MVC   OUTBUF,PRIMOBUF          PRIME UP THE OUTPUT BUFFER
         LA    R4,CSIREC                POINT AT INPUT BUFFER
         L     R5,RPLRLEN               GET THE LENGTH
         AR    R5,R4                    COMPUTE THE LENGTH
         LA    R4,16(,R4)               POINT AT DATASET NAME
         SR    R5,R4                    COMPUTE THE LENGTH
         BCTR  R5,Ø                     DECREMENT FOR THE MVC
         EX    R5,MOVEDSN               GO MOVE THE DSN
         PUT   WORKFL1,OUTBUF
         B     LABLØ1ØØ                 GO DO ANOTHER
*
LABL1ØØØ DS    ØH
*
         CLOSE (CSIACB)
         BAL   R1Ø,DALOCCSI             GO DEALLOCATE THE GLOBAL CSI
         LM    R7,R9,DSNADDRS           PICK UP TABLE OF OTHER CSIS
*
LABL11ØØ DS    ØH
*
         LH    R5,Ø(R7)                 GET LENGTH OF THE DSN
         B     *+1Ø                     BRANCH AROUND THE MOVE
         MVC   TUNIT3+6(*-*),2(R7)      TARGET OF THE EX
         EX    R5,*-6                   NOW MOVE THE DSN
         LA    R5,1(,R5)                INCREMENT LENGTH BY 1
         STH   R5,TUNIT3+4              PUT IT IN THE TEXT UNIT
         BAL   R1Ø,ALLOCCSI             GO ALLOCATE THE DATASET
         OPEN  (CSIACB)
         MVC   GBLCHK7,MDLCHK7          MOVE IN MODEL CHECK FIELD
         LA    R15,DYNXLS               GET ADDRESS OF DYNAMIC EXLIST
         MODCB EXLST=(R15),EODAD=LABL2ØØØ
*
LABL1125 DS    ØH
*
         GET   RPL=(R11)
         CLC   GBLCHK8,CSIREC          Q. ZONE RECORD
         BH    LABL1125                A. NO, HAVEN'T REACHED ZONES
         BL    LABL1225                A. NO, PAST THE ZONE RECORDS
         MVC   GBLCHK7+1(1),CSIREC+5   SAVE THE ZONE NUMBER
         B     LABL1125                GO READ ANOTHER RECORD
*
LABL12ØØ DS    ØH
*
         GET   RPL=(R11)
*
LABL1225 DS    ØH
*
         CLC   GBLCHK7,CSIREC          Q. ARE WE IN A VALID ZONE
         BL    LABL2ØØØ                A. NO
         CLC   GBLCHK6,CSIREC+12       Q. DDDEF
         BNE   LABL12ØØ                A. NO
         MVC   OUTBUF,PRIMOBUF         PRIME UP THE OUTPUT BUFFER
         LA    R4,CSIREC               GET @(INPUT BUFFER)
         L     R5,RPLRLEN              GET LENGTH OF RECORD READ
         AR    R5,R4                   POINT AT END OF THE BUFFER
         LA    R4,16(,R4)              POINT AT START OF DSN
         SR    R5,R4                   CALCULATE DSN LENGTH
         BCTR  R5,Ø                    DECREMENT FOR THE MVC
         EX    R5,MOVEDSN              MOVE DSN TO OUTPUT BUFFER
         PUT   WORKFL1,OUTBUF
         B     LABL12ØØ                GO GET ANOTHER RECORD
*
LABL2ØØØ DS    ØH
*
         CLOSE (CSIACB)
         BAL   R1Ø,DALOCCSI            GO DEALLOCATE CSI
         BXLE  R7,R8,LABL11ØØ          GET NEXT CSI
*
LABL3ØØØ DS    ØH
*
         CLOSE (WORKFL1)
*
LABL31ØØ DS    ØH
*
         LA    R1,PL1                  GET @(PLIST FOR SORT)
         LINK  EP=SORT
         OPEN  (WORKFL2,(INPUT),WORKFL3,(OUTPUT))
*
LABL41ØØ DS    ØH
*
         PUT   WORKFL3,DFDSS1          WRITE DFDSS CONTROL STATEMENT
         PUT   WORKFL3,DFDSS2          WRITE DFDSS CONTROL STATEMENT
*
LABL42ØØ DS    ØH
*                                      LOOP THROUGH INPUT
         GET   WORKFL2,OUTBUF
         PUT   WORKFL3,OUTBUF
         B     LABL42ØØ
*
LABL43ØØ DS    ØH
*
         PUT   WORKFL3,DFDSS3          WRITE DFDSS CONTROL STATEMENT
         PUT   WORKFL3,DFDSS4          WRITE DFDSS CONTROL STATEMENT
         PUT   WORKFL3,DFDSS5          WRITE DFDSS CONTROL STATEMENT
         PUT   WORKFL3,DFDSS6          WRITE DFDSS CONTROL STATEMENT
         CLOSE (WORKFL2,,WORKFL3)
         XC    RET_CODE,RET_CODE       CLEAR THE RETURN CODE
         B     LABL9999                EXIT THE PROGRAM
         TITLE 'A R K S M P D S — DYNAMIC ALLOCATE OF A CSI DATASET'
*
ALLOCCSI DS    ØH
*
         LA    R3,DALSTATS             STATUS SETTING
         STH   R3,TUNIT1               STORE IN TEXT UNIT 1
         MVC   TUNIT1+2,=X'ØØØ1'       INDICATE 1 PARM
         MVC   TUNIT1+4,=X'ØØØ1'       INDICATE LENGTH OF THE PARM
         MVI   TUNIT1+6,X'Ø8'          INDICATE SHR STATUS
         LA    R3,DALDDNAM             DDNAME
         STH   R3,TUNIT2               STORE IN TEXT UNIT 2
         MVC   TUNIT2+2,=X'ØØØ1'       INDICATE 1 PARM
         MVC   TUNIT2+4,=X'ØØØ3'       INDICATE LENGTH OF PARM
         MVC   TUNIT2+6(3),=C'CSI'     MOVE IN DDNAME
         LA    R3,DALDSNAM             INDICATE DATASET NAME
         STH   R3,TUNIT3               STORE IN TEXT UNIT 3
         MVC   TUNIT3+2,=X'ØØØ1'       INDICATE 1 PARM
         LA    R3,TUNIT1               GET @(TEXT UNIT)
         ST    R3,T991                 PUT IT ON THE LIST
         LA    R3,TUNIT2               GET @(TEXT UNIT)
         ST    R3,T992                 PUT IT ON THE LIST
         LA    R3,TUNIT3               GET @(TEXT UNIT)
         ST    R3,T993                 PUT IT ON THE LIST
         OI    T993,X'8Ø'              SET HIGH ORDER BIT ON
         LA    R4,PTR99+4              GET @(RB)
         USING S99RB,R4                LET ASSEMBLER KNOW
         LA    R3,T991                 GET @(FIRST TEXT UNIT)
         ST    R3,S99TXTPP             PUT IT IN THE RB
         MVC   S99RBLN,=AL1(S99RBEND-S99RB) INDICATE THE LENGTH
         MVI   S99VERB,S99VRBAL        SPECIFY ALLOCATE
         ST    R4,PTR99                POINT AT RB
         OI    PTR99,X'8Ø'             HIGH ORDER BIT ON
         DROP  R4                      TELL ASSEMBLER
         LA    R1,PTR99                POINT AT SVC99 INFO
         SVC   99                      LET'S DO IT
         BR    R1Ø                     RETURN
         TITLE 'A R K S M P D S — DYNAMIC DEALLOCATE OF CSI DATASET'
*
DALOCCSI DS    ØH
```

```
*
         LA    R3,DALDDNAM             DDNAME
         STH   R3,TUNIT2               STORE IN TEXT UNIT 2
         MVC   TUNIT2+2,=X'ØØØ1'       INDICATE 1 PARM
         MVC   TUNIT2+4,=X'ØØØ3'       INDICATE LENGTH OF PARM
         MVC   TUNIT2+6(3),=C'CSI'     MOVE IN DDNAME
         LA    R3,TUNIT2               POINT TO TEXT UNIT
         ST    R3,T992                 SAVE IT
         OI    T992,X'8Ø'              INDICATE LAST
         LA    R4,PTR99+4              PICK UP THE POINTER
         USING S99RB,R4                LET THE ASSEMBLER KNOW
         LA    R3,T992                 PICK IT UP
         ST    R3,S99TXTPP             NOW SAVE IT
         MVC   S99RBLN,=AL1(S99RBEND-S99RB) PICK UP THE LENGTH
         MVI   S99VERB,S99VRBUN        PICK UP THE ACTION
         LA    R3,S99RB                POINTER TO REQUEST
         ST    R3,PTR99                SAVE IT
         OI    PTR99,X'8Ø'             HIGH ORDER BIT ON
         DROP  R4
         LA    R1,PTR99                SET REGISTER 1 UP
         SVC   99                      LET'S EXECUTE IT
         BR    R1Ø
*
LABL9999 DS    ØH
*
         $ESAEPI RET_CODE
         TITLE 'A R K S M P D S — LITERALS AND WORK FIELDS'
FORTY6   EQU   46                      SIZE OF DSN
FULL46   DC    A(FORTY6)
MOVEDSN  MVC   OUTBUF+23(*-*),Ø(R4)
SAVEDSN  MVC   2(*-*,R7),Ø(R4)
GBLCHK1  DC    XL4'ØØØØØ3ØØ'
GBLCHK2  DC    XL2'2Ø1Ø'               DLIB/TARGET ZONE RECORD
GBLCHK3  DC    CL8'DLIB    '
GBLCHK4  DC    CL8'TARGET  '
GBLCHK5  DC    XL4'ØØØØØ6Ø5'
GBLCHK6  DC    XL2'Ø67Ø'               DDDEF RECORD
MDLCHK7  DC    XL4'ØØØ1Ø6Ø5'           DDDEF IN TGT/DLIB CSI
GBLCHK8  DC    XL4'ØØØØØ2ØØ'           ZONE ID IN TGT/DLIB
PRIMOBUF DC    CL7Ø'                                                   '
         DC    CL1Ø'- '
DFDSS1   DC    CL7Ø'  DUMP OUTDD(BACKUP)                               '
         DC    CL1Ø'- '
DFDSS2   DC    CL7Ø'       DATASET(INCLUDE(                            '
         DC    CL1Ø'- '
DFDSS3   DC    CL7Ø'                      ))                           '
         DC    CL1Ø'- '
DFDSS4   DC    CL7Ø'       TOL(ENQF)                                   '
         DC    CL1Ø'- '
DFDSS5   DC    CL7Ø'       OPT(3)                                      '
         DC    CL1Ø'- '
```

```
DFDSS6   DC    CL80'      CANCEL                                    '
PL1      DC    A(CTLST)
PL2      DC    F'-1'
CTLST    DS    0H
         DC    AL2(CTL2-CTL1)
CTL1     DC    C' SORT FIELDS=(24,44,CH,A)'
         DC    C' RECORD TYPE=F,LENGTH=80'
         DC    C' SUM FIELDS=NONE'
CTL2     EQU   *
         TITLE 'A R K S M P D S — ACB FOR THE CSI'
CSIACB   ACB   DDNAME=CSI,MACRF=(KEY,IN,SEQ)
         TITLE 'A R K S M P D S — RPL FOR THE CSI'
CSIRPL   RPL   ACB=CSIACB,                                              +
               AREA=CSIRPL,                                             +
               AREALEN=143                                              +
               RECLEN=143,                                              +
               MF=L
CSIRPLL  EQU   *-CSIRPL
CSIXLS   EXLST EODAD=0
CSIXLSL  EQU   *-CSIXLS
         TITLE 'A R K S M P D S — DATA CONTROL BLOCKS'
WORKFL1  DCB   DDNAME=WORKFIL1,DSORG=PS,MACRF=PM
WORKFL2  DCB   DDNAME=WORKFIL1,DSORG=PS,MACRF=GM,EODAD=LABL4300
WORKFL3  DCB   DDNAME=WORKFIL3,DSORG=PS,MACRF=PM
         $ESASTG
RET_CODE DS    F
CSIREC   DS    CL143                   INPUT RECORD
OUTBUF   DS    CL80                    OUTPUT BUFFER
GBLCHK7  DS    XL4                     USED TO FIND
DSNADDRS DS    3F                      USED FOR TABLE ACCESS
DSNTAB   DS    (FORTY6*25)X            TABLE OF CSI DSN'S
DYNRPL   DS    0F                      ALIGN RPL ON FULLWORD
         DS    (CSIRPLL)X
DYNXLS   DS    0F                      ALIGN EXITLIST ON FULLWORD
         DS    (CSIXLSL)X
PTR99    DS    F                       POINTER TO SVC 99 PLIST
         DS    (S99RBEND-S99RB)X       ALLOW SPACE FOR SVC 99 RB
T991     DS    F                       POINTER TO FIRST TEST UNIT
T992     DS    F                       POINTER TO SECOND TEXT UNIT
T993     DS    F                       POINTER TO THIRD TEXT UNIT
TUNIT1   DS    Y
         DS    XL2
         DS    XL2                     PARM LENGTH
         DS    X                       STATUS INDICATOR
TUNIT2   DS    Y
         DS    XL2
         DS    XL2                     PARM LENGTH
         DS    XL3                     DDNAME
TUNIT3   DS    Y
         DS    XL2
         DS    XL2                     PARM LENGTH
         DS    XL44                    DATASET LENGTH
         TITLE 'A R K S M P D S — MAP OUT THE RPL'
         IFGRPL DSECT=YES,AM=VSAM
         TITLE 'A R K S M P D S — SVC99 INFORMATION'
         IEFZB4DØ
         IEFZB4D2
         END    ARKSMPDS
```

$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
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
```

$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
```

$ESAEPI MACRO

```
         MACRO
         $ESASTG
.*********************************************************************
.*
.*       THIS MACRO IS USED IN CONJUNCTION WITH THE $ESAEPI AND $ESAPRO
.*       MACROS.  IT PROVIDES A Q TYPE ADDRESS CONSTANT WHICH WILL CON-
.*       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
.*       .       .      .
.*       .       .      .
.*       .       .      .
.*
.*********************************************************************
RCØØØØ   DC    F'Ø'                         USED TO SET RETURN CODES
RCØØØ4   DC    F'4'                         USED TO SET RETURN CODES
RCØØØ8   DC    F'8'                         USED TO SET RETURN CODES
RCØØØC   DC    F'12'                        USED TO SET RETURN CODES
RCØØ1Ø   DC    F'16'                        USED TO SET RETURN CODES
QDSECT   DC    Q(DSECT)                     DEFINE A QCON
QLENGTH  CXD                                LET ASM CALCULATE THE LENGTH
DSECT    DSECT
         DS    18F                          SET ASIDE REGISTER SAVE AREA
         MEND
```

Enterprise Data Technologies (USA) © Xephon 2000

Converting TPUTs to PUTLINEs

INTRODUCTION

REXX has many inherent routines to perform tasks necessary for programming, but when it comes to system tasks, it is necessary to write some Assembler and get the results by trapping (OUTTRAP). I have seen many systems programmers trapping the Assembler program output in REXX and use it for some other purpose. But here comes the problem; many of the routines might contain TPUTs instead of PUTLINE in the Assembler code, making it impossible to trap the output under REXX. An easy way to get rid of this problem is to convert all the TPUTs to PUTLINEs. But PUTLINE's parameter list is not as simple as TPUT, the programmer has to supply more info like I/O parameter list or CPPL details and also other data. It is a complicated task for a beginner in Assembler language to understand about CPPL and other details. Many entry-level programmers never use PUTLINE and use TPUT. So I thought of writing a simple Assembler macro to make life easier. With this macro, they do not even have to look at a complex PUTLINE!

The MCPUTLN macro is a multipurpose macro, which facilitates three different types of invocation based on the type of call. The macro has to be called once at the beginning of the program, right after setting up addressability. It is also necessary to call this macro at the end of the program, probably after data declaration. Once you have fulfilled this condition, you could call the macro any number of times in the body of the program. It sends a line, 80 characters in length, initialized with blanks before moving the actual contents.

There are three types of calls – 1.SETUP, 2. DISP and 3.DECL. In the beginning of the program, you need to call like this:

```
MCPUTLN SETUP
```

In the body of the program a string can be displayed by calling (any number of times):

```
MCPUTLN DISP,str,len
```

After variable declaration, another mandatory call:

```
    MCPUTLN DECL
```

In an ISPF editor, you may issue this command to convert your TPUTs to MCPUTLNs

```
    CHANGE 'TPUTxx' 'MCPUTLN DISP,'
```

In this case ‘x’ is blank space, it depends on the number of blanks you leave between TPUT and its operand in the original source.

Now you are set ready to trap output displayed by the program you wrote. I wrote a simple Assembler program that displays IPL VOLSER and UCB using TPUT and I used this macro to convert that into a trappable version (PUTLINE).

It traces IPL VOLSER (system residence volume) from CVT and displays it using TPUT.

Remember, this macro assumes your program is a command processor, meaning it should have been either STEPLIBed or linklisted, and it has to be called like a command processor. Otherwise, PUTLINE may not work.

IPLINFTP

```
IPLINFTP CSECT
         SAVE (14,12)                   SAVE REGISTERS
         BALR  R12,Ø
         USING *,R12
         LA    R2,SAVEAREA
         ST    R2,8(,R13)
         ST    R13,SAVEAREA+4
         LR    R13,R2
         L     R2,16
         L     R2,X'3Ø'(R2)
         MVC   VSER(6),X'1C'(R2)
         MVC   VDEV(3),X'D'(R2)
         TPUT  VSER,6
         TPUT  VDEV,3
         L     R13,SAVEAREA+4
         RETURN (14,12),RC=Ø
         YREGS
SAVEAREA DS   18F
VSER     DS   CL6
VDEV     DS   CL3
         END
```

IPLINFXP

```
IPLINFXP CSECT
         SAVE (14,12)                   SAVE REGISTERS
         BALR  R12,Ø
         USING *,R12
         LA    R2,SAVEAREA
         ST    R2,8(,R13)
         ST    R13,SAVEAREA+4
         LR    R13,R2
         MCPUTLN SETUP
         L     R2,16
         L     R2,X'3Ø'(R2)
         MVC   VSER(6),X'1C'(R2)
         MVC   VDEV(3),X'D'(R2)
         MCPUTLN DISP,VSER,6
         MCPUTLN DISP,VDEV,3
*        TPUT  VSER,6
*        TPUT  VDEV,3
         L     R13,SAVEAREA+4
         RETURN (14,12),RC=Ø
         YREGS
SAVEAREA DS   18F
VSER     DS   CL6
VDEV     DS   CL3
         MCPUTLN DECL
         END
```

TRAPIPL

```
/* REXX */
address 'TSO'
a=outtrap(in.)
'iplinfxp'
a=outtrap(off)
do i=1 to in.Ø
   say in.i
end
return
```

MCPUTLN

```
         MACRO
         MCPUTLN  &CALTYP,&STR1,&LEN1
         LCLA  &MAXLEN,&MINLEN
&MINLEN  SETA  1
&MAXLEN  SETA  8Ø
         AIF ('&CALTYP' EQ 'SETUP').PLSETP
         AIF ('&CALTYP' EQ 'DISP').PLCALL
         AIF ('&CALTYP' EQ 'DECL').PLFOOT
.ERR1    MNOTE 12,'INCORRECT CALL TO MCPUTLN..USE SETUP/DISP/DECL'
         MEXIT
.PLSETP  ANOP
```

```
         AIF (N'&SYSLIST NE 1).ERRSETP
         ST    R2,SAVER2
         ST    R3,SAVER3
         USING CPPL,R1
         USING IOPL,R3
         LA    R3,AIOPL
         L     R2,CPPLUPT
         ST    R2,IOPLUPT
         L     R2,CPPLECT
         ST    R2,IOPLECT
         LA    R2,AECB
         ST    R2,IOPLECB
         DROP  R1
         DROP  R3
         L     R2,SAVER2
         L     R3,SAVER3
         MEXIT
.ERRSETP MNOTE 12,'SETUP DOES NOT NEED ADDITIONAL PARAMETERS'
         MEXIT
.PLCALL  ANOP
         AIF (T'&STR1 NE 'C').ERR2
         AIF (T'&LEN1 NE 'N').ERR2
         AIF ('&LEN1' GT '&MAXLEN').ERR3
         AIF ('&LEN1' LT '&MINLEN').ERR3
         AIF (N'&SYSLIST NE 3).ERR4
         MVC   APUTMSG(8Ø),BLKLINE
         MVC   APUTMSG(&LEN1),&STR1
         PUTLINE PARM=PUTMFL,OUTPUT=(APUTHDR,SINGLE,DATA),MF=(E,AIOPL)
         MEXIT
.ERR2    MNOTE 12,'LABEL/LENGTH INVALID, CHECK 2/3RD PARMS'
         MEXIT
.ERR3    MNOTE 12,'INVALID LENGTH.. MUST BE IN BETWEEN 1 AND 8Ø'
         MEXIT
.ERR4    MNOTE 12,'NEEDS 3 POSITIONAL PARAMETERS FOR THIS CALL'
         MEXIT
.PLFOOT  ANOP
         AIF (N'&SYSLIST NE 1).ERRFOOT
AECB     DS    F'Ø'
APUTHDR  DC    H'84'
         DC    H'Ø'
APUTMSG  DS    CL8Ø
AIOPL    DS    4F'Ø'
BLKLINE  DC    8ØCL1' '
SAVER2   DS    F
SAVER3   DS    F
PUTMFL   PUTLINE MF=L
         IKJCPPL
         IKJIOPL
         MEXIT
.ERRFOOT MNOTE 12,'DECL DOES NOT NEED ADDITIONAL PARAMETERS'
.MEND    ANOP
         MEND
```

Shridar Nelliyappan Manivel
Systems Programmer © Xephon 2000

Creating a one-pack OS/390 system

INTRODUCTION

MVS Update Issue 45 (June 1990) contained an article entitled *Creating an MVS mini system* and this was followed up in Issue 49 (October 1990) by an article entitled *Creating an MVS mini system – further comments*. Our installation successfully built a mini-system based on MVS/XA 2.2.3 using these articles.

Since then we have expanded this process to re-create a one-pack system every time we upgraded our version/release of MVS or, its later metamorphosis, OS/390. Each upgrade of the operating system generally involved some changes to the procedure, eg dataset size increases, additional datasets, new/updated PARMLIB members, etc. Our one-pack system is now based on an OS/390 2.6.0 system and the procedure has changed quite significantly from the published article in 1990. The original article stated that 'At IPL time you must expect many messages and replies because of the lack of some system files or PARMLIB members.' We decided to minimize these as we feel time is of the essence in a recovery scenario, there may be an absence of experienced staff and/or the required documentation.

Given that we are now almost ten years down the line, and that we have created one-pack systems for MVS/XA 2.2.3, MVS/ESA 4.2.0, MVS/ESA 4.3.0, OS/390 1.2.0, OS/390 1.3.0 and now OS/390 2.6.0 I think it would helpful to pass on our experiences and job-suite to other installations who may be interested in having a similar recovery system.

The system is extremely useful in cases where changes have been made to the production system with the result that it will no longer IPL. It is then a simple case of IPLing the one-pack system, backing out the offending changes, then successfully (hopefully!) IPLing the production system. If required, the one-pack system can be used to run various DFSMSdss volume restores at a time, thus avoiding the slow, single-threaded DFSMS Stand-Alone Services restore.

It will be of no surprise to you that this system resides on one volume of DASD and it utilizes approximately 750 cylinders of 3390 DASD (which equates to approx 600GB). It can comfortably be dumped on to a 3490 cartridge and it is useful if it is preceded by the DFSMS Stand-Alone Service program. This allows the one-pack system to be restored immediately after IPLing the Stand-Alone Restore program from the 3490 unit. The one-pack system can then be IPLed with minimal effort. Our installation keeps a couple of offsite copies of this tape to facilitate disaster recovery.

SYSTEM CREATION

The various jobs will create an IPLable one-pack system containing JES2, VTAM, TSO, ISPF, and SDSF. We have not included our security product, ACF2, in the one-pack system becausewe want to minimize security issues in the event of a disaster recovery.

Please remember that some of these jobs may need customization to meet your installation requirements. In particular you will almost certainly have to tailor the CONSOLxx and JESPRM00 of SYS1.PARMLIB. These jobs, which should be run in the following order, are fairly self-explanatory and contain comments describing their function. If necessary any further clarification will be inserted after the JCL.

INITVOL

```
//U13ØØMIN JOB ØØ6Ø1Ø3Ø1Ø,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//         CLASS=X,NOTIFY=&SYSUID,REGION=6M
//*
//*****************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                      *
//*   MEM: INITVOL                                                *
//*   DOC: INITIALIZES AN OFFLINE VOLUME AS OS6MIN                *
//*****************************************************************
//*
//INITVOL  EXEC PGM=ICKDSF
//*
//SYSPRINT  DD  SYSOUT=*
//SYSIN     DD  *
  INIT UNITADDRESS(ØFB3) VERIFY(FB3) VOLID(OS6MIN) -
       VTOC(Ø,1Ø,25) INDEX(Ø,1,9)
/*
//
```

IPLTEXT

```
//U13ØØMIN JOB ØØ6Ø1Ø3Ø1Ø,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//         CLASS=X,NOTIFY=&SYSUID,REGION=6M
//*
//*******************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                        *
//*   MEM: IPLTEXT                                                  *
//*   DOC: ADDS IPL TEXT TO OS6MIN VOLUME                           *
//*******************************************************************
//*
//IPLTEXT EXEC PGM=ICKDSF
//*
//SYSPRINT DD  SYSOUT=*
//IPLVOL   DD  DISP=SHR,VOL=SER=OS6MIN,UNIT=339Ø
//IPLTEXT  DD  DSN=SYS1.SAMPLIB(IPLRECS),DISP=SHR
//         DD  DSN=SYS1.SAMPLIB(IEAIPLØØ),DISP=SHR
//SYSIN    DD  *
  REFORMAT DDNAME(IPLVOL)                                     -
           IPLDD(IPLTEXT)                                     -
           NOVERIFY                                           -
           BOOTSTRAP   /* IPLRECS OF IPLTEXT DD WILL SUPPLY IT */
/*
//
```

ALLOCSYS

```
//U13ØØMIN JOB ØØ6Ø1Ø3Ø1Ø,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//         CLASS=X,NOTIFY=&SYSUID,REGION=6M
//*
//*******************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                        *
//*   MEM: ALLOCSYS                                                 *
//*   DOC: CREATES PARMLIB, PROCLIB ETC FOR MINI SYSTEM             *
//*        THESE ARE RENAMED BY MEMBER RENAMSYS                     *
//*******************************************************************
//*
//ALLOC   EXEC PGM=IEFBR14
//*
//SYSPRINT DD  SYSOUT=*
//BRODCAST DD  DSN=MINI.BRODCAST,DISP=(NEW,KEEP),VOL=SER=OS6MIN,
//*            DCB=(RECFM=F,BLKSIZE=129,LRECL=129,DSORG=PS),
//             SPACE=(CYL,(1),,CONTIG),UNIT=339Ø
//HASPACE  DD  DSN=MINI.HASPACE,DISP=(NEW,KEEP),VOL=SER=OS6MIN,
//             SPACE=(CYL,5Ø,,CONTIG),DCB=(DSORG=PS),UNIT=339Ø
//HASPCKPT DD  DSN=MINI.HASPCKPT,DISP=(NEW,KEEP),VOL=SER=OS6MIN,
//             DCB=(RECFM=F,BLKSIZE=4Ø96,LRECL=4Ø96,DSORG=PS),
//             SPACE=(CYL,6,,CONTIG),UNIT=339Ø
//LOGREC   DD  DSN=MINI.LOGREC,DISP=(NEW,KEEP),VOL=SER=OS6MIN,
//*            DCB=(RECFM=U,BLKSIZE=1944,DSORG=PS),
//             SPACE=(CYL,(1),,CONTIG),UNIT=339Ø
```

```
//PARMLIB  DD  DSN=MINI.PARMLIB,DISP=(NEW,KEEP),VOL=SER=OS6MIN,
//             DCB=(RECFM=FB,BLKSIZE=6160,LRECL=80,DSORG=PO),
//             SPACE=(CYL,(1,1,25)),UNIT=3390
//PROCLIB  DD  DSN=MINI.PROCLIB,DISP=(NEW,KEEP),VOL=SER=OS6MIN,
//             DCB=(RECFM=FB,BLKSIZE=6160,LRECL=80,DSORG=PO),
//             SPACE=(CYL,(1,1,25)),UNIT=3390
//UADS     DD  DSN=MINI.UADS,DISP=(NEW,KEEP),VOL=SER=OS6MIN,
//             DCB=(RECFM=FB,BLKSIZE=6160,LRECL=80,DSORG=PO),
//             SPACE=(TRK,(5,1,5)),UNIT=3390
//
```

RENAMSYS

```
//U1300MIN JOB 0060103010,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//         CLASS=X,NOTIFY=&SYSUID,REGION=6M
//*
//*******************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                        *
//*   MEM: RENAMSYS                                                 *
//*   DOC: RENAMES DATASETS CREATED BY MEMBER ALLOCSYS.             *
//*******************************************************************
//*
//RENAME  EXEC PGM=IEHPROGM
//*
//DD1      DD  VOL=SER=OS6MIN,DISP=OLD,UNIT=3390
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  RENAME       DSNAME=MINI.BRODCAST,                                  +
               NEWNAME=SYS1.BRODCAST,                                 +
               VOL=3390=OS6MIN
  RENAME       DSNAME=MINI.HASPACE,                                   +
               NEWNAME=SYS1.HASPACE,                                  +
               VOL=3390=OS6MIN
  RENAME       DSNAME=MINI.HASPCKPT,                                  +
               NEWNAME=SYS1.HASPCKPT,                                 +
               VOL=3390=OS6MIN
  RENAME       DSNAME=MINI.LOGREC,                                    +
               NEWNAME=SYS1.LOGREC,                                   +
               VOL=3390=OS6MIN
  RENAME       DSNAME=MINI.PARMLIB,                                   +
               NEWNAME=SYS1.PARMLIB,                                  +
               VOL=3390=OS6MIN
  RENAME       DSNAME=MINI.PROCLIB,                                   +
               NEWNAME=SYS1.PROCLIB,                                  +
               VOL=3390=OS6MIN
  RENAME       DSNAME=MINI.UADS,                                      +
               NEWNAME=SYS1.UADS,                                     +
               VOL=3390=OS6MIN
/*
//
```

COPYLIB

```
//U13ØØMIN JOB ØØ6Ø1Ø3Ø1Ø,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//          CLASS=X,NOTIFY=&SYSUID,REGION=6M,TIME=144Ø
//*
//*********************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                          *
//*   MEM: COPYLIB                                                    *
//*   DOC: DFDSS COPIES THE EXISTING SYSTEM MAIN LIBRARIES TO         *
//*        THE OS6MIN VOLUME                                          *
//*********************************************************************
//*
//COPY    EXEC PGM=ADRDSSU  /*,PARM='TYPRUN=NORUN'
//*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  COPY DS(INCLUDE(SYS1.CMDLIB, -
                 SYS1.CSSLIB, -
                 SYS1.HELP, -
                 SYS1.IBM.PARMLIB, -
                 SYS1.IBM.PROCLIB, -
                 SYS1.IMAGELIB, -
                 SYS1.LINKLIB, -
                 SYS1.LPALIB, -
                 SYS1.MIGLIB, -
                 SYS1.NUCLEUS, -
                 SYS1.SCBDCLST, -
                 SYS1.SCBD*ENU, -
                 SYS1.SHASLINK, -
                 SYS1.SHASMIG, -
                 SYS1.SISTCLIB, -
                 SYS1.SVCLIB, -
                 SYS1.VTAMLST, -
                 SYS1.VTAMLIB, -
                 SYS2.VTAMLIB, -
                 CPAC.CMDPROC, -
                 CPAC.ISPPLIB -
                 CPAC.LINKLIB -
                 SDSF.HASPINDX, -
                 SDSF.SISF*, -
                 ISPF.SISP*) -
         EXCLUDE(ISPF.SISPALIB, -
                 ISPF.SISPGENU, -
                 ISPF.SISPGMLI, -
                 ISPF.SISPGUI, -
                 ISPF.SISPJSRV, -
                 ISPF.SISPMACS, -
                 ISPF.SISPVENU, -
                 SDSF.SISFSRC)) -
  ODY(OS6MIN) TOL(ENQF) WAIT(Ø,Ø) SHR
/*
//
```

Note 1 – SYS1.IMAGELIB is not required but it avoids JES2 issuing messages:

- HASP871 WARNING - SYS1.IMAGELIB FAILED TO OPEN
- HASP872 REPLY ‘Y’ TO CONTINUE INITIALIZATION, ‘N’ TO TERMINATE
- ‘R’ TO RETRY.

Note 2 – update CPAC.CMDPROC(ISPPDF) on OS6MIN to ensure that only the SYSPROC, SYSEXEC, ISPPROF, ISPMLIB, ISPPLIB, ISPSLIB, ISPTLIB and ISPTABL DDnames are allocated, and that their concatenations include only datasets which exist on the one-pack system.

Note 3 – update SYS1.VTAMLST(ATCCON00) on OS6MIN to remove major nodes that are not required on the one-pack system eg CICSAPPL, etc.

CATDEF

```
//U1300MIN JOB 0060103010,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//         CLASS=X,NOTIFY=&SYSUID,REGION=6M
//*
//*******************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                        *
//*   MEM: CATDEF                                                   *
//*   DOC: DEFINES A USER CATALOG ON OS6MIN WHICH WILL LATER        *
//*        BECOME THE MINI SYSTEM MASTER CATALOG.                   *
//*******************************************************************
//*
//DEFINE  EXEC PGM=IDCAMS
//*
//SYSPRINT DD  SYSOUT=*
//VOL      DD  UNIT=3390,VOL=SER=OS6MIN,DISP=OLD
//SYSIN    DD  *
  DEFINE  USERCATALOG -
          (NAME(CATALOG.MVSICFM.VOS6MIN) -
          CYLINDERS(1 1) -
          VOLUME(OS6MIN) -
          FREESPACE(10 10) -
          RECORDSIZE(4086 4086) -
          ICFCATALOG -
          FILE(VOL)) -
  CATALOG(CATALOG.MVSICFM.VOS6CT1)
/*
//
```

CATBLD

```
//U13ØØMIN JOB ØØ6Ø1Ø3Ø1Ø,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//         CLASS=X,NOTIFY=&SYSUID,REGION=6M
//*
//*******************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                        *
//*   MEM: CATBLD                                                   *
//*   DOC: CATALOGS REQUIRED SYS1, SYS2, SDSF, ISPF AND CPAC        *
//*        DATA SETS IN OS6MIN MASTER CATALOG.                      *
//*******************************************************************
//*
//DEFINE  EXEC PGM=IDCAMS
//*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
 /*****************************************/
 /*   SYS1 DATASETS FOLLOW .......        */
 /*****************************************/
  DEFINE NONVSAM(NAME(SYS1.BRODCAST) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.CMDLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.CSSLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.HASPACE) -
             VOLUME(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.HASPCKPT) -
             VOLUME(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.HELP) -
             VOLUME(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.IBM.PARMLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.IBM.PROCLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.IMAGELIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.LINKLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.LOGREC) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.LPALIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.MIGLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.NUCLEUS) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.PARMLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.PROCLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.SCBDCLST) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.SCBDHENU) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.SCBDMENU) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.SCBDPENU) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.SCBDTENU) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.SHASLINK) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.SHASMIG) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.SISTCLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.SVCLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.UADS) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.VTAMLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(SYS1.VTAMLST) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
```

```
/*****************************************/
/*   SYS2 DATASETS FOLLOW .......        */
/*****************************************/
 DEFINE NONVSAM(NAME(SYS2.VTAMLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
/*****************************************/
/*   SDSF DATASETS FOLLOW .......        */
/*****************************************/
 DEFINE NONVSAM(NAME(SDSF.HASPINDX) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
 DEFINE NONVSAM(NAME(SDSF.SISFEXEC) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
 DEFINE NONVSAM(NAME(SDSF.SISFJCL) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
 DEFINE NONVSAM(NAME(SDSF.SISFLINK) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
 DEFINE NONVSAM(NAME(SDSF.SISFLOAD) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
 DEFINE NONVSAM(NAME(SDSF.SISFLPA) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
 DEFINE NONVSAM(NAME(SDSF.SISFMLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
 DEFINE NONVSAM(NAME(SDSF.SISFPLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
 DEFINE NONVSAM(NAME(SDSF.SISFSLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
 DEFINE NONVSAM(NAME(SDSF.SISFTLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
/*****************************************/
/*   ISPF DATASETS FOLLOW .......        */
/*****************************************/
 DEFINE NONVSAM(NAME(ISPF.SISPCLIB) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
 DEFINE NONVSAM(NAME(ISPF.SISPEXEC) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
 DEFINE NONVSAM(NAME(ISPF.SISPHELP) -
             VOL(OS6MIN) DEVT(339Ø)) -
             CATALOG(CATALOG.MVSICFM.VOS6MIN)
 DEFINE NONVSAM(NAME(ISPF.SISPLOAD) -
               VOL(OS6MIN) DEVT(339Ø)) -
               CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(ISPF.SISPLPA) -
               VOL(OS6MIN) DEVT(339Ø)) -
               CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(ISPF.SISPMENU) -
               VOL(OS6MIN) DEVT(339Ø)) -
               CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(ISPF.SISPPENU) -
               VOL(OS6MIN) DEVT(339Ø)) -
               CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(ISPF.SISPSAMP) -
               VOL(OS6MIN) DEVT(339Ø)) -
               CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(ISPF.SISPSASC) -
               VOL(OS6MIN) DEVT(339Ø)) -
               CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(ISPF.SISPSENU) -
               VOL(OS6MIN) DEVT(339Ø)) -
               CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(ISPF.SISPSLIB) -
               VOL(OS6MIN) DEVT(339Ø)) -
               CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(ISPF.SISPTENU) -
               VOL(OS6MIN) DEVT(339Ø)) -
               CATALOG(CATALOG.MVSICFM.VOS6MIN)
 /******************************************/
 /*   CPAC DATASETS FOLLOW .......         */
 /******************************************/
  DEFINE NONVSAM(NAME(CPAC.CMDPROC) -
               VOL(OS6MIN) DEVT(339Ø)) -
               CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(CPAC.ISPPLIB) -
               VOL(OS6MIN) DEVT(339Ø)) -
               CATALOG(CATALOG.MVSICFM.VOS6MIN)
  DEFINE NONVSAM(NAME(CPAC.LINKLIB) -
               VOL(OS6MIN) DEVT(339Ø)) -
               CATALOG(CATALOG.MVSICFM.VOS6MIN)
/*
//
```

PAGEDEF

```
//U13ØØMIN JOB ØØ6Ø1Ø3Ø1Ø,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//         CLASS=X,NOTIFY=&SYSUID,REGION=6M
//*
//*******************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                        *
//*   MEM: PAGEDEF                                                  *
//*   DOC: DEFINES PLPA, COMMON AND LOCAL PAGE DATASETS             *
//*******************************************************************
```

```
//*
//DEFINE  EXEC PGM=IDCAMS
//*
//STEPCAT  DD  DSN=CATALOG.MVSICFM.VOS6MIN,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//PAGEVOL  DD  UNIT=339Ø,VOL=SER=OS6MIN,DISP=SHR
//SYSIN    DD  *
  DEFINE PAGESPACE(NAME(PAGE.VOS6MIN.PLPA) -
                    CYLINDERS(1) -
                    FILE(PAGEVOL) -
                    VOLUMES(OS6MIN)) -
         CATALOG(CATALOG.MVSICFM.VOS6MIN)

  DEFINE PAGESPACE(NAME(PAGE.VOS6MIN.COMMON) -
                    CYLINDERS(125) -
                    FILE(PAGEVOL) -
                    VOLUMES(OS6MIN)) -
         CATALOG(CATALOG.MVSICFM.VOS6MIN)

  DEFINE PAGESPACE(NAME(PAGE.VOS6MIN.LOCALM) -
                    CYLINDERS(1ØØ) -
                    FILE(PAGEVOL) -
                    VOLUMES(OS6MIN)) -
         CATALOG(CATALOG.MVSICFM.VOS6MIN)
/*
//
```

IODFDEF

```
//U13ØØMIN JOB ØØ6Ø1Ø3Ø1Ø,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//         CLASS=X,NOTIFY=&SYSUID,REGION=6M
//*
//*********************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                          *
//*   MEM: IODFDEF                                                    *
//*   DOC: CREATES SYSØ.IODFØØ DATASET                                *
//*********************************************************************
//*
//*    NB CHANGE LINE 15 TO THE DRIVING SYSTEM'S IODF DATA SET NAME
//*
//IODFDEF EXEC PGM=IDCAMS
//*
//STEPCAT  DD  DSN=CATALOG.MVSICFM.VOS6MIN,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//IN       DD  DSN=SYSØ.IODF58,DISP=SHR
//VOL      DD  UNIT=339Ø,VOL=SER=OS6MIN,DISP=OLD
//SYSIN    DD  *
  DELETE (SYSØ.IODFØØ.CLUSTER) CATALOG(CATALOG.MVSICFM.VOS6MIN) -
     FILE(VOL)
  SET LASTCC = Ø
  SET MAXCC = Ø
  DEFINE CLUSTER(NAME(SYSØ.IODFØØ.CLUSTER) -
                 VOLUMES(OS6MIN) -
                 CYLINDERS(1Ø) -
                 FILE(VOL) -
                 LINEAR) -
            DATA(NAME(SYSØ.IODFØØ)) -
         CATALOG(CATALOG.MVSICFM.VOS6MIN)

  IF LASTCC = Ø THEN +
    DEFINE PATH(NAME(MINISYS.SYSØ.IODFØØ.CLUSTER) -
                PATHENTRY(SYSØ.IODFØØ.CLUSTER)) -
        CATALOG(CATALOG.MVSICFM.VOS6MIN)

  IF LASTCC = Ø THEN +
    DEFINE ALIAS(NAME(MINISYS) -
                 RELATE(CATALOG.MVSICFM.VOS6MIN)) -
         CATALOG(CATALOG.MVSICFM.VOS6CT1)

  IF LASTCC = Ø THEN +
    REPRO INFILE(IN) OUTDATASET(MINISYS.SYSØ.IODFØØ.CLUSTER)

  DELETE MINISYS ALIAS
  DELETE MINISYS.SYSØ.IODFØØ.CLUSTER PATH -
         CATALOG(CATALOG.MVSICFM.VOS6MIN)
/*
//
```

Note 1 – change the IN DDname DSN= parameter to your driving system's IODF dataset name.

Note 2 – change CATALOG.MVSICFM.VOS6CT1 to your driving system's master catalog name.

Note 3 – the DEFINE PATH and DEFINE ALIAS statements allow the newly created SYS0.IODF00 to be accessed from the driving system.

Note 4 – creating SYS0.IODF00, instead of SYS0.IODFnn, means no changes are required to the IODF statement in the SYS1.PARMLIB LOADxx members whenever the system is (re)built from a new IODF.

CATPTR

```
//U13ØØMIN JOB ØØ6Ø1Ø3Ø1Ø,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//         CLASS=X,NOTIFY=&SYSUID,REGION=6M
//*
//*******************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                        *
//*   MEM: CATPTR                                                   *
//*   DOC: REPLACE MEMBER 'SYS1.NUCLEUS(SYSCATLG)' TO FACILITATE    *
//*        THE USE OF AN EXISTING ICF MASTER CATALOG WITH A NEWLY     *
//*        CREATED IPL VOLUME.                                        *
//*******************************************************************
//*
//* NOTE: THE FORMAT OF SYSCATLG IN SYS1.NUCLEUS HAS CHANGED FOR
//*       MVS/DFP VERSION 3. THE NEW FORMAT OF SYSCATLG IS:
//*       VVVVVVTALLNNNNNN.....
//*       VVVVVV    = VOLSER
//*       T         = CATALOG TYPE
//*                   1 = ICF CATALOG
//*                   2 = ICF CATALOG WITH SYS% TO SYS1 CONVERSION
//*                       ACTIVE
//*       A         = NUMBER OF MLA ALIAS LEVELS (1-4)
//*                   DEFAULT IS 1
//*       LL        = CAS SERVICE TASK LOWER LIMIT
//*                   DEFAULT=X'3C', MINIMUM=X'18', MAXIMUM=X'C8'
//*       NNNNNN... = CATALOG NAME (UP TO 44 CHARACTERS)
//*
//REPLACE EXEC PGM=IEBDG
//*
//SYSPRINT DD  SYSOUT=*
//NUCLEUS  DD  DSN=SYS1.NUCLEUS(SYSCATLG),DISP=SHR,
//             UNIT=339Ø,VOL=SER=OS6MIN
//SYSIN    DD  *
  DSD OUTPUT=(NUCLEUS)
  FD NAME=VOL,
     LENGTH=Ø6,
     STARTLOC=Ø1,
     FILL=X'4Ø',
     PICTURE=6,'OS6MIN' <—— VERIFY/CHANGE VOLSER OF CATALOG
  FD NAME=CATTYPE,
     LENGTH=Ø1,
     STARTLOC=Ø7,
     FILL=X'F1'    <—— ICF CATALOG
  FD NAME=ALIAS,
     LENGTH=Ø1,
     STARTLOC=Ø8,
     FILL=X'F1'    <——NUMBER OF MLA ALIAS LEVELS (DEFAULT=1)
  FD NAME=CASLL,
     LENGTH=Ø2,
     STARTLOC=Ø9,
     FILL=X'4Ø',
     PICTURE=2,'3C' <——CAS SERVICE TASK LOWER LIMIT (DEFAULT=6Ø)
  FD NAME=CAT,
     LENGTH=44,
     STARTLOC=11,
     FILL=X'4Ø',
     PICTURE=44,'CATALOG.MVSICFM.VOS6MIN                    '
  FD NAME=FIL,
     LENGTH=26,
     STARTLOC=55,
     FILL=X'4Ø'
  CREATE QUANTITY=1,FILL=X'ØØ',NAME=(VOL,CATTYPE,ALIAS,CASLL,CAT,FIL)
/*
//
```

Note 1 – the details supplied in this member will only be used if there is no SYSCAT statement supplied in the SYS1.PARMLIB(LOADxx) member.

PARMUPDT

```
//U13ØØMIN JOB ØØ6Ø1Ø3Ø1Ø,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//         CLASS=X,NOTIFY=&SYSUID,REGION=6M
//*
//*******************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                        *
//*   MEM: PARMUPDT                                                 *
//*   DOC: CREATES THE REQUIRED PARMLIB MEMBERS                     *
//*******************************************************************
//*
//UPDATE  EXEC PGM=IEBUPDTE,PARM='NEW'
//*
//SYSPRINT DD  SYSOUT=*
//SYSUT2   DD  DSN=SYS1.PARMLIB,DISP=SHR,
//             UNIT=339Ø,VOL=SER=OS6MIN
//SYSIN    DD  DATA,DLM='@@'
./ ADD    NAME=ALLOCØØ
 TIOT SIZE(64)              /* SIZE OF THE TASK I/O TABLE           */
./ ADD    NAME=CLOCKØØ
OPERATOR NOPROMPT
TIMEZONE E.ØØ.ØØ.ØØ
ETRMODE  NO
ETRZONE  NO
ETRDELTA 1
./ ADD    NAME=COFVLFØØ
CLASS NAME(CSVLLA)
      EMAJ(LLA)
./ ADD    NAME=COMMNDØØ
COM='S VLF,SUB=MSTR'
COM='S NET'
COM='S SDSF'
./ ADD    NAME=CONSOLDM
INIT     CMDDELIM(;)
         MLIM(9999)
         MONITOR(DSNAME)
         MMS(NO)
         PFK(ØØ)
         RLIM(1Ø)
         UEXIT(N)
         AMRF(N)
/*                                                                 */
```

```
DEFAULT ROUTCODE(ALL)
/*                                                                */
CONSOLE DEVNUM(BAØ) ALTERNATE(BA1)                /* MCONS - BAØ    */
        ROUTCODE(1-1Ø,12-128)
        PFKTAB(PFKTAB1)
        AUTH(MASTER)
        NAME(XDEVEXT2)
        UNIT(327Ø-X)
        MONITOR(JOBNAMES-T)
        RBUF(1Ø)
        CON(N) SEG(39) DEL(R) RNUM(19) RTME(1) MFORM(J,T) AREA(NONE)
/*                                                                */
CONSOLE DEVNUM(BA1) ALTERNATE(EAØ)                /* MCONS - BA1    */
        ROUTCODE(1-1Ø,12-128)
        PFKTAB(PFKTAB1)
        AUTH(MASTER)
        NAME(XDEVSEC)
        UNIT(327Ø-X)
        MONITOR(JOBNAMES-T)
        RBUF(1Ø)
        CON(N) SEG(39) DEL(R) RNUM(19) RTME(1) MFORM(J,T) AREA(NONE)
/*                                                                */
CONSOLE DEVNUM(EAØ) ALTERNATE(EA1)                /* MCONS - EAØ    */
        ROUTCODE(1-1Ø,12-128)
        PFKTAB(PFKTAB1)
        AUTH(MASTER)
        NAME(XDEVEXT1)
        UNIT(327Ø-X)
        MONITOR(JOBNAMES-T)
        RBUF(1Ø)
        CON(N) SEG(39) DEL(R) RNUM(19) RTME(1) MFORM(J,T) AREA(NONE)
/*                                                                */
CONSOLE DEVNUM(EA1) ALTERNATE(BA3)                /* MCONS - EA1    */
        ROUTCODE(1-1Ø,12-128)
        PFKTAB(PFKTAB1)
        AUTH(MASTER)
        NAME(XDEVPRI)
        UNIT(327Ø-X)
        MONITOR(JOBNAMES-T)
        RBUF(1Ø)
        CON(N) SEG(39) DEL(R) RNUM(19) RTME(1) MFORM(J,T) AREA(NONE)
/*                                                                */
CONSOLE DEVNUM(BA3)                               /* MCONS - BA3    */
        ROUTCODE(1-1Ø,12-128)
        PFKTAB(PFKTAB1)
        AUTH(MASTER)
        NAME(XDEVTST)
        UNIT(327Ø-X)
        MONITOR(JOBNAMES-T)
        RBUF(1Ø)
        CON(N) SEG(39) DEL(R) RNUM(19) RTME(1) MFORM(J,T) AREA(NONE)
/*                                                                */
```

```
./ ADD    NAME=CONSOLPM
INIT     CMDDELIM(;)
         MLIM(9999)
         MONITOR(DSNAME)
         MMS(NO)
         PFK(ØØ)
         RLIM(1Ø)
         UEXIT(N)
         AMRF(N)
/*                                                                     */
DEFAULT ROUTCODE(ALL)
/*                                                                     */
CONSOLE DEVNUM(BAØ) ALTERNATE(BA1)                /* MCONS - BAØ    */
        ROUTCODE(1-1Ø,12-128)
        PFKTAB(PFKTAB1)
        AUTH(MASTER)
        NAME(XPRDEXT2)
        UNIT(327Ø-X)
        MONITOR(JOBNAMES-T)
        RBUF(1Ø)
        CON(N) SEG(39) DEL(R) RNUM(19) RTME(1) MFORM(J,T) AREA(NONE)
/*                                                                     */
CONSOLE DEVNUM(BA1) ALTERNATE(EAØ)                /* MCONS - BA1    */
        ROUTCODE(1-1Ø,12-128)
        PFKTAB(PFKTAB1)
        AUTH(MASTER)
        NAME(XPRDSEC)
        UNIT(327Ø-X)
        MONITOR(JOBNAMES-T)
        RBUF(1Ø)
        CON(N) SEG(39) DEL(R) RNUM(19) RTME(1) MFORM(J,T) AREA(NONE)
/*                                                                     */
CONSOLE DEVNUM(EAØ) ALTERNATE(EA1)                /* MCONS - EAØ    */
        ROUTCODE(1-1Ø,12-128)
        PFKTAB(PFKTAB1)
        AUTH(MASTER)
        NAME(XPRDEXT1)
        UNIT(327Ø-X)
        MONITOR(JOBNAMES-T)
        RBUF(1Ø)
        CON(N) SEG(39) DEL(R) RNUM(19) RTME(1) MFORM(J,T) AREA(NONE)
/*                                                                     */
CONSOLE DEVNUM(EA1) ALTERNATE(BA3)                /* MCONS - EA1    */
        ROUTCODE(1-1Ø,12-128)
        PFKTAB(PFKTAB1)
        AUTH(MASTER)
        NAME(XPRDPRI)
        UNIT(327Ø-X)
        MONITOR(JOBNAMES-T)
        RBUF(1Ø)
        CON(N) SEG(39) DEL(R) RNUM(19) RTME(1) MFORM(J,T) AREA(NONE)
/*                                                                     */
```

```
CONSOLE DEVNUM(BA3)                               /* MCONS - BA3    */
        ROUTCODE(1-1Ø,12-128)
        PFKTAB(PFKTAB1)
        AUTH(MASTER)
        NAME(XPRDTST)
        UNIT(327Ø-X)
        MONITOR(JOBNAMES-T)
        RBUF(1Ø)
        CON(N) SEG(39) DEL(R) RNUM(19) RTME(1) MFORM(J,T) AREA(NONE)
/*                                                                  */
./ ADD    NAME=CONSOLSM
INIT     CMDDELIM(;)
         MLIM(9999)
         MONITOR(DSNAME)
         MMS(NO)
         PFK(ØØ)
         RLIM(1Ø)
         UEXIT(N)
         AMRF(N)
/*                                                                  */
DEFAULT ROUTCODE(ALL)
/*                                                                  */
CONSOLE DEVNUM(9ØØ) ALTERNATE(91Ø)                /* MCONS - 9ØØ    */
        ROUTCODE(1-1Ø,12-128)
        PFKTAB(PFKTAB1)
        AUTH(MASTER)
        NAME(XSPGMSTR)
        UNIT(327Ø-X)
        MONITOR(JOBNAMES-T)
        RBUF(1Ø)
        CON(N) SEG(39) DEL(RD) RNUM(19) RTME(1) MFORM(J,T) AREA(NONE)
/*                                                                  */
CONSOLE DEVNUM(91Ø)                               /* MCONS - 91Ø    */
        ROUTCODE(1-1Ø,12-128)
        PFKTAB(PFKTAB1)
        AUTH(ALL)
        NAME(XSPGBKUP)
        UNIT(327Ø-X)
        MONITOR(JOBNAMES-T)
        RBUF(1Ø)
        CON(N) SEG(39) DEL(RD) RNUM(19) RTME(1) MFORM(J,T) AREA(NONE)
/*                                                                  */
./ ADD    NAME=COUPLEØØ
 COUPLE SYSPLEX(LOCAL)
./ ADD    NAME=CTIXCFØØ
TRACEOPTS ON
          BUFSIZE(72K)
./ ADD    NAME=DEVSUPØØ
COMPACT=YES,
VOLNSNS=YES
./ ADD    NAME=DIAGØØ
 VSM TRACK CSA(ON) SQA(ON)
 VSM TRACE GETFREE(OFF)
./ ADD    NAME=IEACMDØØ
COM='CD SET,SDUMP=(ALLPSA,GRSQ,RGN,LSQA,SWA,TRT,SUM,NUC,LPA,SQA),Q=YES'
COM='CD SET,SDUMP,TYPE=XMEME'
COM='SET DAE=ØØ'
COM='SET SLIP=ØØ'
COM='START LLA,SUB=MSTR'
COM='START BLSJPRMI,SUB=MSTR'
./ ADD    NAME=IEAFIXØØ
INCLUDE LIBRARY(SYS1.LPALIB)
        MODULES(IEAVARØØ,       /*  7K  RCT INIT/TERM     */
                IEAVARØ6,       /*  RCT INIT/TERM ALIAS   */
                IGCØØØ1G)       /*  456 RESTORE(SVC17)    */
./ ADD    NAME=IEALPAØØ
INCLUDE LIBRARY(CPAC.LINKLIB)
        MODULES(IEFACTRT)
INCLUDE LIBRARY(SDSF.SISFLPA)
        MODULES(ISFLPA,IGXØØØ11)
./ ADD    NAME=IEAPAKØØ
(IEFBR14)
./ ADD    NAME=IEASLPØØ
SLIP SET,C=Ø13,ID=XØ13,A=NOSVCD,J=JES2,END
SLIP SET,C=Ø28,ID=XØ28,A=NOSVCD,END
SLIP SET,C=Ø52,ID=XØ52,A=NODUMP,J=CATALOG,END
SLIP SET,C=Ø58,ID=XØ58,A=NODUMP,DATA=(15R,EQ,4,OR,15R,EQ,8,OR,
     15R,EQ,C,OR,15R,EQ,1Ø,OR,15R,EQ,2C,OR,15R,EQ,3Ø,OR,
     15R,EQ,3C),END
SLIP SET,C=Ø66,ID=XØ66,A=NODUMP,J=CATALOG,END
SLIP SET,C=Ø7Ø,ID=XØ7Ø,A=NODUMP,J=CATALOG,END
SLIP SET,C=Ø73,ID=XØ73,A=NODUMP,J=CATALOG,END
SLIP SET,C=ØDX,ID=XØDX,A=NODUMP,J=CATALOG,END
SLIP SET,C=ØE7,ID=XØE7,A=NOSVCD,END
SLIP SET,C=ØF3,ID=XØF3,A=NODUMP,END
SLIP SET,C=13E,ID=X13E,A=NODUMP,END
SLIP SET,C=1C5,RE=ØØØ9ØØØ4,ID=X1C5,A=NODUMP,END
SLIP SET,C=222,ID=X222,A=NODUMP,END
SLIP SET,C=322,ID=X322,A=NODUMP,END
SLIP SET,C=33E,ID=X33E,A=NODUMP,END
SLIP SET,C=3C4,REASON=1A,ID=S3C4,A=SVCD,END
SLIP SET,C=422,ID=X422,A=NODUMP,END
SLIP SET,C=47B,DATA=(15R,EQ,Ø,OR,15R,EQ,8),ID=X47B,A=NODUMP,END
SLIP SET,C=622,ID=X622,A=NODUMP,END
SLIP SET,C=71A,ID=X71A,A=NODUMP,END
SLIP SET,C=8Ø4,ID=X8Ø4,A=(NOSVCD,NOSYSU),END
SLIP SET,C=8Ø6,ID=X8Ø6,A=(NOSVCD,NOSYSU),END
SLIP SET,C=8ØA,ID=X8ØA,A=(NOSVCD,NOSYSU),END
SLIP SET,C=81A,ID=X81A,A=NODUMP,END
SLIP SET,C=91A,ID=X91A,A=NODUMP,END
SLIP SET,C=9FB,ID=X9FB,A=NOSVCD,J=JES3,END
SLIP SET,C=B37,ID=XB37,A=(NOSVCD,NOSYSU),END
SLIP SET,C=C1A,ID=XC1A,A=NODUMP,END
SLIP SET,C=D1A,ID=XD1A,A=NODUMP,END
SLIP SET,C=D37,ID=XD37,A=(NOSVCD,NOSYSU),END
SLIP SET,C=E37,ID=XE37,A=(NOSVCD,NOSYSU),END
SLIP SET,C=EC6,RE=0000FFXX,ID=XEC6,A=NODUMP,END
SLIP SET,C=EC6,RE=0000FDXX,ID=XXC6,A=NOSVCD,END
./ ADD    NAME=IEASYSDM
CON=DM                        SELECT CONSOLDM
./ ADD    NAME=IEASYSPM
CON=PM                        SELECT CONSOLPM
./ ADD    NAME=IEASYSSM
CON=SM                        SELECT CONSOLSM
./ ADD    NAME=IEASYS00
ALLOC=00,                     SELECT ALLOC00
APG=07,                       AUTOMATIC PRIORITY GROUP IS 7      DEFAULT
CLOCK=00,                     SELECT CLOCK00                     DEFAULT
CLPA,                         CREATE LINK PACK AREA
CMB=(UNITR,COMM,GRAPH,CHRDR), ADDITIONAL CMB ENTRIES
CMD=00,                       TOD PROMPT, SDUMP, TRACE ON AND RMF /*J3*/
CSA=(2000,100000),            MVS/ESA CSA RANGE
DEVSUP=00,                    IDRC COMPRESSION
DIAG=00,                      SELECT DIAG00, DIAGNOSTIC COMMANDS
DUMP=NO,                      NO DUMP DATASETS
FIX=00,                       FIX MODULES SPECIFIED               /*J3*/
GRS=NONE,                     NO COORDINATION OF GRS REQUESTS
ICS=00,                       SELECT IEAICS00 INSTALL CNTL SPECS FOR SRM
IOS=00,                       SELECT IECIOS00 MIH PARAMETERS
IPS=00,                       SELECT IEAIPS00 INSTALL PERF SPECS FOR SRM
LNKAUTH=LNKLST,               AUTHORIZE LNKLST00-DEFAULT, APFTAB IS ALT
LOGCLS=L,                     WILL NOT BE PRINTED BY DEFAULT
LOGLMT=999999,                MUST BE 6 DIGITS, MAX WTL MESSAGES QUEUED
LPA=00,                       SELECT LPALST00
MAXUSER=250,                  (SYS TASKS + INITS + TSOUSERS)
MLPA=00,                      MLPA PARAMETERS
MSTRJCL=00,                   MASTER JCL - MSTJCL00              DEFAULT
OPI=YES,                      ALLOW OPERATOR OVERRIDE TO IEASYS00
OPT=00,                       SPECIFY IEAOPT00 (SRM TUNING PARMETERS)
PAGE=(PAGE.VOS6MIN.PLPA,      PAGE DATASET (PLPA)
      PAGE.VOS6MIN.COMMON,    PAGE DATASET (COMMON)
      PAGE.VOS6MIN.LOCALM,L), PAGE DATASET (LOCAL)
PAGTOTL=(5,2),                ALLOW ADDITION 3 PAGE D/S AND 2 SWAP D/S
PAK=00,                       IEAPAK00
PLEXCFG=XCFLOCAL,             LOCAL SYSPLEX
PROD=00,                      SELECT IFAPRD00
PROG=00,                      DYNAMIC APF
REAL=128,                     ALLOWS 2 64K JOBS OR 1 128K JOB TO RUN V=R
RSU=0,                        NO RECONFIG STORAGE UNITS          DEFAULT
RSVSTRT=5,                    RESERVED ASVT ENTRIES              DEFAULT
RSVNONR=5,                    RESERVED ASVT ENTRIES              DEFAULT
SCH=00,                       SELECT SCHED00                     DEFAULT
SMF=00,                       SELECT SMFPRM00, SMF PARAMETERS    DEFAULT
SQA=(10,80),                  MVS/ESA SQA APPROX 1216K
SSN=00,                       SUBSYSTEM INITIALIZATION NAMES
SVC=00,                       SELECT IEASVC00                    DEFAULT
```

```
SYSNAME=MINI,                 SAME AS SID IN SMFPRMØØ
VAL=ØØ,                       SELECT VATLSTØØ                    DEFAULT
VIODSN=IGNORE,                NO VIO DATA SET
VRREGN=64                     DEFAULT REAL-STORAGE REGION SIZE   DEFAULT
./ ADD    NAME=IECIOSØØ
MIH DASD=Ø5:ØØ
MIH TAPE=1Ø:ØØ
MIH TIME=2Ø:ØØ,DEV=(33Ø-337,34Ø-343,B4Ø-B43)
MIH TIME=ØØ:ØØ,DEV=(444-447,C44-C47)
./ ADD    NAME=IEFSSNØØ
SUBSYS SUBNAME(JES2) PRIMARY(YES)                      /* JES2 */
./ ADD    NAME=IFAPRDØØ
   WHEN (HWNAME(*))
PRODUCT OWNER('IBM CORP')
        NAME(OS/39Ø)
        ID(5647-AØ1)
        VERSION(*) RELEASE(*) MOD(*)
        FEATURENAME(OS/39Ø)
        STATE(ENABLED)
PRODUCT OWNER('IBM CORP')
        NAME(OS/39Ø)
        ID(5647-AØ1)
        VERSION(*) RELEASE(*) MOD(*)
        FEATURENAME(DFSMSDSS)
        STATE(ENABLED)
PRODUCT OWNER('IBM CORP')
        NAME(OS/39Ø)
        ID(5647-AØ1)
        VERSION(*) RELEASE(*) MOD(*)
        FEATURENAME(SDSF)
        STATE(ENABLED)
./ ADD    NAME=IKJTSOØØ
/* LIB: CPAC.PARMLIB(IKJTSOØØ)                                         */
/* DOC: THIS MEMBER IS USED AT IPL TIME TO DEFINE THE AUTHORIZED       */
/*      COMMAND LIST, THE AUTHORIZED PROGRAM LIST, THE NOT             */
/*      BACKGROUND COMMAND LIST, THE AUTHORIZED BY THE TSO SERVICE     */
/*      FACILITY LIST, AND TO CREATE THE DEFAULTS THE SEND COMMAND     */
/*      WILL USE.                                                      */
/*                                                                     */
AUTHCMD NAMES(          /* AUTHORIZED COMMANDS       */ +
   RECEIVE              /* TSO COMMANDS              */ +
   CONSPROF             /*                           */ +
   TRANSMIT XMIT        /*                           */ +
   LISTB    LISTBC      /*                           */ +
   SE       SEND        /*                           */ +
   RACONVRT             /*                           */ +
   SYNC                 /*                           */ +
   LISTD    LISTDS      /*                           */ +
   IKJEHDS1             /*                           */ +
   TESTAUTH TESTA       /*                           */ +
   PARMLIB  IKJPRMLB    /*                           */ +
   BINDDATA BDATA       /* REQUIRED FOR ISMF         */ +
   LISTDATA LDATA       /* REQUIRED FOR ISMF         */ +
   SETCACHE SETC        /* REQUIRED FOR ISMF         */ +
   SHCDS                /*                           */ +
   DEFINE               /* DFP COMMANDS              */ +
   IMPORT               /* DFP COMMANDS              */ +
             )          /*                           */
                        /*                           */
AUTHPGM NAMES(          /* AUTHORIZED PROGRAMS       */ +
   IEBCOPY              /*                           */ +
   IDCAMS               /*                           */ +
             )          /*                           */
                        /*                           */
NOTBKGND NAMES(         /* COMMANDS WHICH MAY NOT BE */ +
                        /* ISSUED IN THE BACKGROUND  */ +
   OPER     OPERATOR    /*                           */ +
   TERM     TERMINAL    /*                           */ +
             )          /*                           */
                        /*                           */
AUTHTSF NAMES(          /* PROGRAMS TO BE AUTHORIZED */ +
                        /* WHEN CALLED THROUGH THE   */ +
                        /* TSO SERVICE FACILITY.     */ +
   IEBCOPY              /*                           */ +
   ICQASLIØ             /*                           */ +
   IKJEFF76             /*                           */ +
             )          /*                           */
                        /*                           */
SEND                    /* SEND COMMAND DEFAULTS     */ +
   OPERSEND(ON)         /*                           */ +
   USERSEND(ON)         /*                           */ +
   SAVE(ON)             /*                           */ +
   CHKBROD(ON)          /*                           */ +
   LOGNAME(SYS1.BRODCAST) /*                         */
                        /*                           */
ALLOCATE                /* ALLOCATE COMMAND DEFAULT  */ +
   DEFAULT(OLD)         /*                           */
                        /*                           */
TRANSREC                /* TRANSREC COMMAND DEFAULT  */ +
   NODESMF((NODEM,MINI))  /*                         */ +
   CIPHER(YES)          /*                           */ +
   SPOOLCL(B)           /*                           */ +
   OUTWARN(5ØØØØ,15ØØØ) /*                           */ +
   OUTLIM(5ØØØØØØ)      /*                           */ +
   VIO(VIO)             /*                           */ +
   LOGSEL(LOG)          /*                           */ +
   LOGNAME(MISC)        /*                           */ +
   DAPREFIX(TUPREFIX)   /*                           */ +
   USRCTL(NAMES.TEXT)   /*                           */ +
   SYSOUT(*)            /*                           */
./ ADD    NAME=ISFPRMØØ

    /******************************************/
    /* OPTIONS Statement - Global SDSF Options */
    /******************************************/
```

```
OPTIONS ATHOPEN(YES),       /* Use authorized open for datasets    */
  DCHAR('?'),               /* Command query character             */
  DSI(NO),                  /* Bypass ENQ for dynamic allocation   */
  FINDLIM(5000),            /* Maximum lines to search for FIND    */
  IDBLKS(4096),             /* HASPINDX blocksize                  */
  INDEX(SDSF.HASPINDX),     /* HASPINDX dataset name               */
  JESDATA(VERSIONS),        /* Use checkpoint versioning           */
  LINECNT(55),              /* Print lines per page                */
  LOGLIM(0),                /* OPERLOG search limit in hours       */
  MENUS(SDSF.SISFPLIB),     /* Panels dataset name for TSO         */
  NIDBUF(5),                /* Number of haspindx buffers          */
  NSPBUF(5),                /* Number of spool buffers             */
  SCHARS('*%'),             /* Generic and placeholder characters  */
  SCRSIZE(1920),            /* Maximum screen size                 */
  SYSOUT(A),                /* Default print sysout class          */
  TRACE(C000),              /* Default trace mask                  */
  TRCLASS(A),               /* Default trace sysout class          */
  UNALLOC(NO)               /* Do not free dynalloc datasets       */

   /**************************************/
   /* GROUP ISFSPROG - System Programmers */
   /**************************************/

GROUP NAME(ISFSPROG),       /* Group name                          */
TSOAUTH(JCL,OPER,ACCT),     /* User must have JCL, OPER, ACCT      */
ACTION(ALL),                /* All route codes displayed           */
ACTIONBAR(YES),             /* Display the action bar on panels    */
APPC(OFF),                  /* Include APPC sysout                 */
AUPDT(2),                   /* Minimum auto update interval        */
AUTH(LOG,I,O,H,DA,DEST,PREF,     /* Authorized functions           */
     SYSID,ABEND,ACTION,INPUT,
     FINDLIM,ST,INIT,PR,TRACE,
     ULOG,MAS,SYSNAME,LI,SO,NO,PUN,RDR,JC,SE,RES),
CMDAUTH(ALL),               /* Commands allowed for all jobs       */
CMDLEV(7),                  /* Authorized command level            */
CONFIRM(ON),                /* Enable cancel confirmation          */
CURSOR(ON),                 /* Leave cursor on last row processed  */
DADFLT(IN,OUT,TRANS,STC,TSU,JOB),  /* Default rows shown on DA     */
DATE(DDMMYYYY),             /* Default date format                 */
DATESEP('/'),               /* Default datesep format              */
DFIELD2(DAFLD2),            /* Sample alternate field list for DA  */
DISPLAY(OFF),               /* Do not display current values       */
DSPAUTH(ALL),               /* Browse allowed for all jobs         */
GPLEN(2),                   /* Group prefix length                 */
ILOGCOL(1),                 /* Initial display column in log       */
ISYS(LOCAL),                /* Initial system default for DA       */
LANG(ENGLISH),              /* Default language                    */
LOGOPT(OPERACT),            /* Default log option                  */
OWNER(NONE),                /* Default owner                       */
UPCTAB(TRTAB2),             /* Upper case translate table name     */
VALTAB(TRTAB),              /* Valid character translate table     */
VIO(SYSALLDA)               /* Unit name for page mode output      */
```

```
   /*****************************/
   /* GROUP ISFOPER - Operators */
   /*****************************/

GROUP NAME(ISFOPER),        /* Group name                           */
TSOAUTH(JCL,OPER),          /* User must have JCL and OPER          */
ACTION(ALL),                /* All route codes displayed            */
ACTIONBAR(YES),             /* Display action bar on panels         */
APPC(OFF),                  /* Include APPC sysout                  */
AUPDT(2),                   /* Minimum auto update interval         */
AUTH(LOG,I,O,H,DA,PREF,DEST,  /* Authorized functions               */
     SYSID,ACTION,FINDLIM,ST,
     INIT,PR,ULOG,MAS,SYSNAME,LI,
     SO,NO,PUN,RDR,JC,SE,RES),
CMDAUTH(ALL),               /* Commands allowed for all jobs        */
CMDLEV(7),                  /* Authorized command level             */
CONFIRM(ON),                /* Enable cancel confirmation           */
CURSOR(ON),                 /* Leave cursor on last row processed   */
DADFLT(IN,OUT,TRANS,STC,TSU,JOB),  /* Default rows shown on DA      */
DATE(DDMMYYYY),             /* Default date format                  */
DATESEP('/'),               /* Default datesep format               */
DISPLAY(OFF),               /* Do not display current values        */
DSPAUTH(USERID,NOTIFY,AMSG),  /* Browse authority                   */
GPLEN(2),                   /* Group prefix length                  */
ILOGCOL(1),                 /* Initial display column in log        */
ISYS(LOCAL),                /* Initial system default for DA        */
LANG(ENGLISH),              /* Default language                     */
LOGOPT(OPERACT),            /* Default log option                   */
OWNER(NONE),                /* Default owner                        */
UPCTAB(TRTAB2),             /* Upper case translate table name      */
VALTAB(TRTAB),              /* Valid character translate table      */
VIO(SYSALLDA)               /* Unit name for page mode output       */

   /*********************************/
   /* GROUP ISFUSER - General Users */
   /*********************************/

GROUP NAME(ISFUSER),        /* Group name                           */
TSOAUTH(JCL),               /* User must have JCL                   */
ACTION(11,12,USER),         /* Default route codes in log           */
ACTIONBAR(YES),             /* Display action bar on panels         */
APPC(OFF),                  /* Include APPC sysout                  */
AUPDT(1Ø),                  /* Default auto update interval         */
AUTH(I,O,H,DA,ST,SE),       /* Authorized functions                 */
CMDAUTH(USERID,NOTIFY),     /* Command authority                    */
CMDLEV(2),                  /* Command level                        */
CONFIRM(ON),                /* Enable cancel confirmation           */
CURSOR(ON),                 /* Leave cursor on last row processed   */
DADFLT(IN,OUT,TRANS,STC,TSU,JOB),  /* Default rows on DA            */
DATE(DDMMYYYY),             /* Default date format                  */
DATESEP('/'),               /* Default datesep format               */
DISPLAY(OFF),               /* Do not display current values        */
```

```
DSPAUTH(USERID,NOTIFY),     /* Browse authority                    */
ILOGCOL(1),                 /* Initial display column in log       */
LANG(ENGLISH),              /* Default language                    */
LOGOPT(OPERACT),            /* Default log option                  */
OWNER(USERID),              /* Default owner                       */
PREFIX(USERID),             /* Default prefix                      */
UPCTAB(TRTAB2),             /* Upper case translate table name     */
VALTAB(TRTAB),              /* Valid character translate table     */
VIO(SYSALLDA)               /* Unit name for page mode output      */

   /*******************/
   /* Sample NTBL list */
   /*******************/

NTBL NAME(SLIST)

  NTBLENT STRING(œS),OFFSET(1)
  NTBLENT STRING(P),OFFSET(7)
  NTBLENT STRING(PAY),OFFSET(3)

   /*******************************/
   /* Define default SDSF Codepage */
   /*******************************/

TRTAB CODPAG(SDSF) VALTAB(TRTAB) UPCTAB(TRTAB2)

   /*******************************************/
   /* Sample alternate field list for DA display */
   /*******************************************/

FLD NAME(DAFLD2) TYPE(DA)  /* Name is referenced by GROUP statement */

  FLDENT COLUMN(STEPN),TITLE(STEPNAME),WIDTH(D)
  FLDENT COLUMN(PROCS),TITLE(PROCSTEP),WIDTH(D)
  FLDENT COLUMN(JOBID),TITLE(JOBID),WIDTH(D)
  FLDENT COLUMN(OWNERID),TITLE(OWNER),WIDTH(D)
  FLDENT COLUMN(JCLASS),TITLE(C),WIDTH(D)
  FLDENT COLUMN(ASID),TITLE(ASID),WIDTH(D)
  FLDENT COLUMN(ASIDX),TITLE(ASIDX),WIDTH(D)
  FLDENT COLUMN(EXCP),TITLE(' EXCP-CNT'),WIDTH(D)
  FLDENT COLUMN(CPU),TITLE('  CPU-TIME'),WIDTH(D)
  FLDENT COLUMN(REAL),TITLE(REAL),WIDTH(D)
  FLDENT COLUMN(PAGING),TITLE(PAGING),WIDTH(D)
  FLDENT COLUMN(EXCPRT),TITLE('   SIO'),WIDTH(D)
  FLDENT COLUMN(CPUPR),TITLE('  CPU%'),WIDTH(D)
  FLDENT COLUMN(DP),TITLE(DP),WIDTH(D)
  FLDENT COLUMN(POS),TITLE(POS),WIDTH(D)
  FLDENT COLUMN(SWAPR),TITLE(SR),WIDTH(D)
  FLDENT COLUMN(PGN),TITLE(PGN),WIDTH(D)
  FLDENT COLUMN(DOMAIN),TITLE(DMN),WIDTH(D)
  FLDENT COLUMN(STATUS),TITLE(STATUS),WIDTH(D)
  FLDENT COLUMN(WORKLOAD),TITLE(WORKLOAD),WIDTH(D)
```

```
   FLDENT COLUMN(SRVCLASS),TITLE(SRVCLASS),WIDTH(D)
   FLDENT COLUMN(PERIOD),TITLE(SP),WIDTH(D)
   FLDENT COLUMN(RESGROUP),TITLE(RESGROUP),WIDTH(D)
   FLDENT COLUMN(SERVER),TITLE(SERVER),WIDTH(D)
   FLDENT COLUMN(QUIESCE),TITLE(QUIESCE),WIDTH(D)
   FLDENT COLUMN(SYSNAME),TITLE(SYSNAME),WIDTH(D)
   FLDENT COLUMN(SPAGING),TITLE(SPAG),WIDTH(D)
   FLDENT COLUMN(SCPU),TITLE('SCPU%'),WIDTH(D)
   FLDENT COLUMN(ECPU),TITLE(' ECPU-TIME'),WIDTH(D)
   FLDENT COLUMN(ECPUPR),TITLE(' ECPU%'),WIDTH(D)

./ ADD    NAME=JESPRMØØ
**********************************************************************
**********************************************************************
**                                                                  **
** BECAUSE OF ITS SIZE THE JES2 PARAMETER DECK IS NOT INCLUDED HERE. **
** IT IS ESSENTIALLY THE SAME AS THE PRODUCTION JES2 PARAMETER DECK  **
** WITH THE FOLLOWING EXCEPTIONS:-                                   **
**                                                                  **
** STATEMENT          CHANGES REQUIRED                              **
** _____         ________                              **
** CKPTDEF            CKPT1 VOLSER=OS6MIN                           **
**                    REMOVE CKPT2, NEWCKPT1 AND NEWCKPT2 PARAMETERS **
**                    DUPLEX=OFF                                    **
**                                                                  **
** EXIT(nnn)          REMOVE ALL EXIT(nnn) STATEMENTS               **
**                                                                  **
** LOADMOD(jxxxxxxx)  REMOVE ALL LOADMOD(jxxxxxxx) STATEMENTS        **
**                                                                  **
** MASDEF             OWNMEMB=MINI                                  **
**                                                                  **
** MEMBER(1)          NAME=MINI                                     **
**                                                                  **
** SPOOLDEF           VOLUME=OS6MI                                  **
**                                                                  **
**********************************************************************
**********************************************************************
./ ADD    NAME=LOADDM
IODF     ØØ SYSØ     XDEV     ØØ Y
NUCLEUS  1
PARMLIB  SYS1.PARMLIB                                OS6MIN
PARMLIB  SYS1.IBM.PARMLIB                            OS6MIN
SYSCAT   OS6MIN113CCATALOG.MVSICFM.VOS6MIN
SYSPARM  DM
./ ADD    NAME=LOADPM
IODF     ØØ SYSØ     XPRD     ØØ Y
NUCLEUS  1
PARMLIB  SYS1.PARMLIB                                OS6MIN
PARMLIB  SYS1.IBM.PARMLIB                            OS6MIN
SYSCAT   OS6MIN113CCATALOG.MVSICFM.VOS6MIN
SYSPARM  PM
./ ADD    NAME=LOADSM
IODF     ØØ SYSØ     XSPG     ØØ Y
```

```
NUCLEUS  1
PARMLIB  SYS1.PARMLIB                                 OS6MIN
PARMLIB  SYS1.IBM.PARMLIB                             OS6MIN
SYSCAT   OS6MIN113CCATALOG.MVSICFM.VOS6MIN
SYSPARM  SM
./ ADD    NAME=LPALSTØØ
SYS1.LPALIB,
ISPF.SISPLPA,
SDSF.SISFLOAD,
SDSF.SISFLPA
./ ADD    NAME=MSTJCLØØ
//MSTJCLØØ JOB MSGLEVEL=(1,1),TIME=144Ø
//         EXEC PGM=IEEMB86Ø,DPRTY=(15,15)
//STCINRDR DD SYSOUT=(A,INTRDR)
//TSOINRDR DD SYSOUT=(A,INTRDR)
//IEFPDSI  DD DSN=SYS1.PROCLIB,DISP=SHR
//         DD DSN=SYS1.IBM.PROCLIB,DISP=SHR
//SYSUADS  DD DSN=SYS1.UADS,DISP=SHR
//SYSLBC   DD DSN=SYS1.BRODCAST,DISP=SHR
./ ADD    NAME=PFKTABØØ
PFKTAB TABLE(PFKTAB1)
PFK(Ø1) CMD('K S,DEL=RD,SEG=19,CON=N,RNUM=39,RTME=Ø1,MFORM=J;K A,NONE')
PFK(Ø2) CMD('K E,1       ERASE ONE LINE')
PFK(Ø3) CMD('K E,D       ERASE STATUS DISPLAY IN AREA')
PFK(Ø4) CMD('K D,F       FRAME DISPLAY FORWARD IN AREA')
PFK(Ø5) CMD('K S,DEL=N   HOLD IN-LINE OUTPUT')
PFK(Ø6) CMD('K S,DEL=RD  RESUME IN-LINE OUTPUT')
PFK(Ø7) CMD('D A,L       LIST ACTIVE JOBS AND TSO USERS')
PFK(Ø8) CMD('D R,L       LIST OPERATOR REQUESTS')
PFK(Ø9) CMD('K D,U       UPDATE DYNAMIC DISPLAY')
PFK(1Ø) CMD('D C,M       MASTER CONSOLE')
PFK(11) CMD('D C,HC      HARDCOPY')
PFK(12) CMD('D A;D C     COMMANDS')
./ ADD    NAME=PROGØØ
APF FORMAT(DYNAMIC)
APF ADD DSNAME(SYS1.SISTCLIB)                               VOL(******)
APF ADD DSNAME(SYS1.VTAMLIB)                                VOL(******)
APF ADD DSNAME(SYS2.VTAMLIB)                                VOL(******)
/*                                                                  */
/* DYNAMIC LNKLST - PRODUCT DEFINITIONS                             */
/*                                                                  */
LNKLST DEFINE NAME(LNKLSTØØ)
LNKLST ADD NAME(LNKLSTØØ) DSN(CPAC.LINKLIB)
LNKLST ADD NAME(LNKLSTØØ) DSN(SYS1.CMDLIB)
LNKLST ADD NAME(LNKLSTØØ) DSN(SYS1.SHASLINK)
LNKLST ADD NAME(LNKLSTØØ) DSN(SYS1.SHASMIG)
LNKLST ADD NAME(LNKLSTØØ) DSN(ISPF.SISPLOAD)
LNKLST ADD NAME(LNKLSTØØ) DSN(ISPF.SISPSASC)
LNKLST ADD NAME(LNKLSTØØ) DSN(SDSF.SISFLINK)
LNKLST ADD NAME(LNKLSTØØ) DSN(SYS1.SCBDHENU)
LNKLST ACTIVATE NAME(LNKLSTØØ)
./ ADD    NAME=SCHEDØØ
```

```
 /*                   SDSF SERVER                                    */
MT SIZE(64K)                           /* MASTER TRACE TBL SIZE      */
PPT PGMNAME(ISFHCTL)                   /* PROGRAM NAME               */
    KEY(1)                             /* PROTECTION KEY             */
    NOSWAP                             /* NON-SWAPPABLE              */
./ ADD    NAME=SMFPRMØØ
     ACTIVE                      /* ACTIVE SMF RECORDING*/
     DSNAME(SYS1.MAN1)           /* DATA SETS*/
     NOPROMPT                    /* DO NOT PROMPT OPERATOR FOR OPTIONS*/
     REC(PERM)                   /* TYPE 17 PERM RECORDS ONLY*/
     MAXDORM(3ØØØ)               /* WRITE AN IDLE BUFFER AFTER 3Ø MIN*/
     STATUS(Ø1ØØØØ)              /* WRITE SMF STATS AFTER 1 HOUR*/
     JWT(ØØ28)                   /* 522 AFTER 28 MINUTES*/
     SID(MINI)                   /* SYSTEM ID IS MINI  .NU..RU.*/
     LISTDSN                     /* LIST DATA SET STATUS AT IPL*/
     SYS(NOTYPE(Ø:255),EXITS(IEFU83,IEFU84,IEFACTRT,IEFUJV,
                    IEFUSI,IEFUJI,IEFUTL,IEFU29),
                    INTERVAL(ØØ3ØØØ),DETAIL)

     /* WRITE ALL EXCEPT DATA MANAGEMENT RECORDS, TAKE ALL KNOWN
        EXITS, NOTE: JES EXITS CONTROLED BY JES , THERE IS NO
        DEFAULT INTERVAL RECORDS WRITTEN AND ONLY SUMMARY T32
        RECORDS AS A DEFAULT FOR TSO */

     SUBSYS(STC,EXITS(IEFU29,IEFU83,IEFU84,IEFUJP,IEFUSO))

     /* WRITE RECORDS ACCORDING TO SYS VALUE, TAKE ONLY FIVE
        EXITS, NOTE: IEFU29 EXECUTES IN THE MASTER ASID WHICH IS A
        STC ADDRESS SPACE SO IEFU29 MUST BE ON FOR STC. USE ALL OTHER
        SYS PARMETERS AS A DEFAULT  */

./ ADD    NAME=TSOKEYØØ
USERMAX=2ØØ,                                                          +
RECONLIM=3,                                                           +
BUFRSIZE=3Ø16,                                                        +
HIBFREXT=48ØØØ,                                                       +
LOBFREXT=24ØØØ,                                                       +
CHNLEN=4,                                                             +
SCRSIZE=192Ø,                                                         +
CONFTXT=NO
./ ADD    NAME=VATLSTØØ
VATDEF IPLUSE(PRIVATE),SYSUSE(PRIVATE)
OS6MIN,1,Ø,339Ø    ,N        MIMI SYSTEM
./       ENDUP
@@
//
```

PROCUPDT

```
//U13ØØMIN JOB ØØ6Ø1Ø3Ø1Ø,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//          CLASS=X,NOTIFY=&SYSUID,REGION=6M
```

```
//*
//*******************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                        *
//*   MEM: PROCUPDT                                                 *
//*   DOC: CREATES THE REQUIRED PROCLIB MEMBERS.                    *
//*******************************************************************
//*
//UPDATE  EXEC PGM=IEBUPDTE,PARM='NEW'
//*
//SYSPRINT DD  SYSOUT=*
//SYSUT2   DD  DSN=SYS1.PROCLIB,DISP=SHR,
//             UNIT=3390,VOL=SER=OS6MIN
//SYSIN    DD  DATA,DLM='@@'
./ ADD    NAME=IKJACCNT
//IKJACCNT PROC
//IKJACCNT EXEC PGM=IKJEFT01,DYNAMNBR=256,PARM=ISPPDF
//SYSPROC  DD DSN=CPAC.CMDPROC,DISP=SHR
//SYSHELP  DD DSN=SYS1.HELP,DISP=SHR
//SYSLBC   DD DSN=SYS1.BRODCAST,DISP=SHR
//SYSPRINT DD TERM=TS,SYSOUT=*
//SYSTERM  DD TERM=TS,SYSOUT=*
//SYSIN    DD TERM=TS
//*
./ ADD    NAME=JES2
//JES2    PROC M=00
//IEFPROC EXEC PGM=HASJES20,DPRTY=(15,15),TIME=1440,PERFORM=9
//PROC00   DD  DSN=SYS1.PROCLIB,DISP=SHR
//         DD  DSN=SYS1.IBM.PROCLIB,DISP=SHR
//HASPPARM DD  DSN=SYS1.PARMLIB(JESPRM&M),DISP=SHR
//IEFRDER  DD  SYSOUT=*
//HASPLIST DD  DDNAME=IEFRDER
./ ADD    NAME=NET
//NET    PROC
//NET     EXEC PGM=ISTINM01,DPRTY=(15,15),
//        TIME=1440,PERFORM=8,REGION=100M
//VTAMLST  DD DSN=SYS1.VTAMLST,DISP=SHR
//VTAMLIB  DD DSN=SYS2.VTAMLIB,DISP=SHR
//         DD DSN=SYS1.VTAMLIB,DISP=SHR
//SISTCLIB DD DSN=SYS1.SISTCLIB,DISP=SHR
//SYSABEND DD SYSOUT=*,HOLD=YES
//CSDATA   DD SYSOUT=Y
./ ADD    NAME=SDSF
//SDSF    PROC M=00,           /* SUFFIX FOR ISFPRMXX */
//             P='LC(X)'       /* USE SYSOUT CLASS X FOR SDSFLOG */
//*
//SDSF    EXEC PGM=ISFHCTL,REGION=32M,TIME=1440,PARM='M(&M),&P'
./ ADD    NAME=TSO
//TSO     PROC MBR=TSOKEY00
//STEP1   EXEC PGM=IKTCAS00,TIME=1440
//PARMLIB  DD  DSN=SYS1.PARMLIB(&MBR),DISP=SHR,FREE=CLOSE
//PRINTOUT DD  SYSOUT=*,FREE=CLOSE
./       ENDUP
@@
//
```

ZAPRML

```
//U13ØØMIN JOB ØØ6Ø1Ø3Ø1Ø,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//         CLASS=X,NOTIFY=&SYSUID,REGION=6M
//*
//*******************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                        *
//*   MEM: ZAPRML                                                   *
//*   DOC: REMOVES OLD-STYLE RESOURCE MANAGER ROUTINES              *
//*******************************************************************
//*
//ZAPRML  EXEC PGM=AMASPZAP,PARM='IGNIDRFULL'
//*
//SYSLIB   DD  DSN=SYS1.LPALIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
 NAME IGCØØØ1C IEAVTRML
 VER  ØØØØØØ   D4D6C4D5,C1D4C5F1      MODNAME1 <— CHANGE TO MOD NAME
 REP  ØØØØØØ   ØØØØØØØØ,ØØØØØØØØ      BINARY ZEROS =================
/*
//
```

Note 1 – this job should only be updated and run if CSECT IEAVTRML, the old-style resource manager routines, has been zapped with module name(s). Failure to remove the module name(s) will cause an abend S806 and subsequent IPL failure if the module(s) could not be located on the one-pack system.

TSOADD

```
//U13ØØMIN JOB ØØ6Ø1Ø3Ø1Ø,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//         CLASS=X,NOTIFY=&SYSUID,REGION=6M
//*
//*******************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                        *
//*   MEM: TSOADD                                                   *
//*   DOC: CREATES TSO USERID 'IBMUSER' WITH PASSWORD 'IBMUSER'     *
//*******************************************************************
//*
//TSOADD  EXEC PGM=IKJEFTØ1
//*
//SYSTSPRT DD  SYSOUT=*
//SYSUADS  DD  DSN=SYS1.UADS,DISP=SHR,
//             VOL=SER=OS6MIN,UNIT=339Ø
//SYSLBC   DD  DSN=SYS1.BRODCAST,DISP=SHR,
//             VOL=SER=OS6MIN,UNIT=339Ø
//SYSTSIN  DD  *
ACCOUNT
    SYNC UADS
    D (IBMUSER)
    A (IBMUSER IBMUSER ACCT# IKJACCNT) JCL OPER ACCT NOMOUNT NOLIM     +
     SIZE(4096) UNIT(SYSALLDA)
    LIST (*)
END
/*
//
```

INITLOGR

```
//U1300MIN JOB 0060103010,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//         CLASS=X,NOTIFY=&SYSUID,REGION=6M
//*
//*******************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                        *
//*   MEM: INITLOGR                                                 *
//*   DOC: INITIALISES OS6MIN SYS1.LOGREC DATASET                   *
//*******************************************************************
//*
//INITLOG EXEC PGM=IFCDIP00
//*
//SERERDS  DD  DSN=SYS1.LOGREC,DISP=SHR,
//             UNIT=3390,VOL=SER=OS6MIN
//
```

LISTMCAT

```
//U1300MIN JOB 0060103010,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//         CLASS=X,NOTIFY=&SYSUID,REGION=6M
//*
//*******************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                        *
//*   MEM: LISTMCAT                                                 *
//*   DOC: LISTS THE OS6MIN MASTER CATALOG ENTRIES                  *
//*******************************************************************
//*
//LISTCAT EXEC PGM=IDCAMS
//*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  LISTCAT CATALOG(CATALOG.MVSICFM.VOS6MIN) ALL
/*
//
```

JOB:DSSBSAT

```
//U1300MIN JOB 0060103010,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//         CLASS=X,NOTIFY=&SYSUID,REGION=6M
//*
```

```
//*******************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                        *
//*   MEM: DSSBSAT                                                  *
//*   DOC: COPIES DFSMS STAND-ALONE SERVICES PGM TO FILE 1 OF NL CART *
//*******************************************************************
//*
//BUILDSA EXEC PGM=ADRDSSU,PARM='UTILMSG=YES'
//*
//SAMODS   DD  DSN=SYS1.SADRYLIB,DISP=SHR
//TAPEDD   DD  DSN=ADRSA.IPLT,UNIT=CART,LABEL=(,NL),DISP=(NEW,KEEP),
//             DCB=(DSORG=PS,RECFM=U,BLKSIZE=3276Ø,LRECL=3276Ø),
//             VOL=SER=MINSPG
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  BUILDSA -
  INDD(SAMODS) -
  OUTDD(TAPEDD) -
  OPERCNSL(9ØØ) -
  IPL(TAPE)
/*
//
```

FULLDUMP

```
//U13ØØMIN JOB ØØ6Ø1Ø3Ø1Ø,'OS6MIN SYSTEM',MSGLEVEL=(1,1),MSGCLASS=X,
//         CLASS=X,NOTIFY=&SYSUID,REGION=6M
//*******************************************************************
//*   LIB: SYSP.MINISYS.CNTL                                        *
//*   MEM: FULLDUMP                                                 *
//*   DOC: COPIES OS6MIN SYSTEM TO FILE 2 OF NL CARTRIDGE           *
//*******************************************************************
//*
//DUMP    EXEC PGM=ADRDSSU
//SYSPRINT DD  SYSOUT=*
//INVOL1   DD  VOL=SER=OS6MIN,UNIT=339Ø,DISP=SHR
//OUTDD1   DD  DSN=FULLDUMP,DISP=(NEW,KEEP),UNIT=CART,VOL=SER=MINSPG,
//             DCB=BLKSIZE=3276Ø,LABEL=(2,NL)
//SYSIN    DD  *
  DUMP FULL -
  INDDNAME(INVOL1) -
  OUTDDNAME(OUTDD1) -
  ALLDATA(*) -
  ALLEXCP -
  CANCELERROR -
  OPTIMIZE(4) -
  WAIT(2,2)
/*
//
```

IPL PROCEDURES

Our one-pack system has been created on a shared DASD volume which is accessible from our development, production and systems programming LPARs. As such, the above jobs have been created to allow the one-pack system to be IPLed in any of the LPARs. Essentially the only differences are the console definitions because the systems may not always use the same device numbers. The alternate CONSOLxx members are accessed via the IEASYSxx members which are pointed to by the relevant LOADxx members (where xx is DM, PM or SM for development mini, production mini or sysprogs mini respectively). This technique means we do not need to create and maintain three seperate one-pack systems.

In a disaster recovery situation you could IPL from the cartridge which was created in the DSSBSAT and FULLDUMP jobs above. This would invoke the DFSMS Stand-Alone Services program and it is then a case of using it to restore the OS6MIN volume from the second file on the cartridge. From this point, simply follow the procedures below.

Specify the device number of OS6MIN in the IPL Unit Address as well as in the Load Parm field. For example, in our case we have an IPL Unit Address of 0FB3 and a Load Parm of 0FB3PM (for production). As mentioned above we could use a Load Parm of 0FB3DM for development, or 0FB3SM for sysprogs. The one-pack system should then be IPLed according to the usual procedures for your machine make/model.

After the IPL has started the following messages will appear on the operator console.

```
IEA376I VIODSN PARAMETER IS 'IGNORE'. NO VIO JOURNALING

IEA3Ø1I IEASVCØØ NOT FOUND IN PARMLIB

ILRØØ5E PLPA PAGE DATA SET FULL, OVERFLOWING TO COMMON DATA SET

IEA3Ø1I IEAICSØØ NOT FOUND IN PARMLIB
IEA341A RESPECIFY ICS      PARM OR PRESS ENTER
IRA851E SKELETON ICS IN EFFECT.  PLEASE ISSUE THE SET ICS COMMAND.

===> Press ENTER to the IEA341A message <====
IEFC452I OMVS - JOB NOT RUN - JCL ERROR

IEE95ØI SMF SYS1.MAN1 DATASET CANNOT BE ALLOCATED
    RETURN CODE=ØØØ4. ERROR CODE=17Ø8. INFO. CODE=ØØØ2
IEE366I NO SMF DATA SETS AVAILABLE—DATA BEING BUFFERED TIME=HH:MM:SS

ADYØ1ØE THE DAE START TRANSACTION FAILED
   ALLOCATION FOR SYS1.DAE FAILED
   THE DATA ET COULD NOT BE FOUND
```

From here the IPL should look fairly normal. The next main message is:

```
     œHASP426 SPECIFY OPTIONS - JES2 OS 2.5.Ø

     ===> Reply R Ø1,COLD,NOREQ to the œHASP426 message <===
```

A COLD start is only required when the system is IPLed for the first time after being (re)built. Thereafter a WARM start will suffice.

```
     ===> S TSO after NET has initialized <===
```

Log-on to TSO with user-id IBMUSER, account IKJACCNT, and password IBMUSER.

POST-IPL PROCEDURES

Once logged on to the one-pack system you can correct any errors in the ‘real’ system datasets, restore other volumes etc. Once this is done, it is a simple case of shutting down the recovery system and IPLing the ‘real’ system using the usual IPL Unit Address and Load Parms.

FINAL THOUGHT

Although we would all prefer not to be in the situation where we cannot IPL our production system, it is comforting to know that we have this system lying around. To this end I would recommend that you keep it fairly up to date and test it on a regular basis. I would certainly consider rebuilding it every time you upgrade your OS/390 maintenance levels.

Ian McArthur
Systems Programmer (UK)

Initializing COBOL variables

THE PROBLEM

Initializing variables is always a good idea. Sometimes it is an absolute requirement. Mostly it's a pain. Moving zeros and spaces to all those working-storage variables takes a lot of code. If we had a structure like this:

```
  Ø1 GROUP1.
Ø5 LEVEL-1-STUFF.
  1Ø NAME            PIC X(3Ø).
  1Ø AGE             PIC 9(3).
Ø5 OTHER-STUFF.
  1Ø SPOUSE-NAME     PIC X(3Ø).
  1Ø SPOUSE-AGE      PIC 9(3).
     88 SHE-WONT-TELL           VALUE ZERO.
Ø5 PHONE-NUMBER      PIC X(1Ø).
Ø5 US-SHORT-ZIP-CODE  PIC 9(5).
```

Our procedure division code might look like this:

```
MOVE SPACES TO NAME
               SPOUSE-NAME
               PHONE-NUMBER.
MOVE ZEROS  TO AGE
               SPOUSE
               US-SHORT-ZIP-CODE.
```

AN EASIER WAY

INITIALIZE was introduced to make our job easier. All of our procedure division code can be replaced with one line:

```
INITIALIZE GROUP1.
```

INITIALIZE will move either spaces or zeros as appropriate to each elementary-level variable. These are the default values. We can get fancier.

Tables can be initialized by specifying the table as a group item that contains the complete table. A subscripted item may be specified, but no item or anything within it can contain the 'DEPENDING' option of the 'OCCURS' clause nor can it contain a 'RENAMES' clause. Also, it cannot be an index data item. The general format is (from *SAA CPI COBOL Level 2 Reference (IBM)*):

```
|                     <———————+
| >>—INITIALIZE——identifier-1——————————————————————————|
|
|
| >——————————————————————————————————————————————————>< |
|    |                <—————————————————————————————+ |                  |
|   +-REPLACING———ALPHABETIC———————————BY——identifier-2——+
|
|                     +-ALPHANUMERIC————|  +-DATA-+      +-literal-1——+
|
|                     +-NUMERIC——————|
|
|                     +-ALPHANUMERIC-EDITED-|
|
|                     +-NUMERIC-EDITED———|
|
|                     +-DBCS——————————+
```

When the REPLACING phrase is used identifier-2 or literal-1 (the element after the ‘BY’) must be compatible with the category in the REPLACING phrase. In other words, it must obey the rules for a MOVE. Obviously, you cannot repeat the same category again within the same statement.

Give it a try. If you want to try something unusual, code it up with a dump of the area immediately following your INITIALIZE statement to see what happened.

Allan Kalar
Systems Programmer (USA)

Cursor-sensitive ISPF (part 4)

This month we complete our look at the cursor-sensitive ‘DS’ command.

```
/*——————————————————*/
/* check user-level libraries first */
/*——————————————————*/
If libtype = 'DATASET' & Left(file,3) = 'ISP' Then Do
   If file = 'ISPFILE'
      Then userlib = 'ISPFILU'
      Else userlib = Left(file,4)'USR'
   Call SEARCH_FILE(userlib)            /* search libraries    */
```

```
        End

      /*----------------*/
      /* check normal LIBDEF libraries */
      /*----------------*/
      If libtype = 'DATASET' ! libtype = 'EXCLDATA' Then Do
         dsnlist = Translate(libid,"  ","'",") /* no quotes or commas */
         Call FIND_MEMBER                      /* look for the member */
         If result = Ø Then EXIT dsn           /* RETURN the DSNAME   */
         End

      If libtype = 'LIBRARY' ! libtype = 'EXCLLIBR' Then
         Call SEARCH_FILE(libid)               /* search libraries    */
      End

/*--------------------------------*/
/* Not REXX/CLIST or LIBDEF, so check the specified ddname */
/*--------------------------------*/
  Call SEARCH_FILE(file)                       /* search libraries    */

  Select                                       /* RETURN: ERROR MSG   */
    When result = 4 Then
      EXIT member 'not found in' Strip(file)
    When result = 8 ! (result = 69 & SYSREASON = 2) Then
      EXIT 'DDname:' file 'not found'
    When result = 12 Then
      EXIT member 'has zero-length in' sysdsname
    When result = 69 Then
      EXIT 'LISTDSI Error, reason code='sysreason ,
           '   Searching file:' Strip(file)
    Otherwise
      EXIT 'Error RC='result ' searching file:' Strip(file)
    End

/*==============================================================*/
/* Search for the member in the passed ddname                    */
/*   - When member is found EXIT is done, returning the dsn (& vol) */
/*-------------------------------------*/
SEARCH_FILE: Arg dd

    Call CHECK_1ST_DSN(dd)                     /* check 1st library   */
    If result = Ø Then
       EXIT sysdsname sysvolume                /* RETURN DSNAME & VOL */
    If result = 12 ! result > 16
       Then Return result                      /* listdsi/browse error*/

    Call FIND_DSNLIST(dd)                      /* get library list    */
    If result = 8 Then Return 8                /* dd not found        */

    dsnlist = Subword(dsnlist,2)               /* start with 2nd dsn  */
    Call FIND_MEMBER                            /* look for the member */
    If result = Ø Then EXIT dsn                 /* RETURN the DSNAME   */

    Return 4                                    /* member not found    */

/*===================================================================*/
/* Check for the member in the first library in the ddname          */
/*   - if BROWSE rc>Ø then no panel would be displayed, and a dummy */
/*     DISPLAY is done to use the pending 'END' command             */
/*-----------------------------------*/
CHECK_1ST_DSN: Arg ddfind

    x = LISTDSI(ddfind 'FILE')                  /* get 1st dsn and vol */
    If x > Ø Then Return 69

    "CONTROL DISPLAY SAVE"                      /* save what we have   */
    "CONTROL NONDISPL END"                      /* pending 'END' cmd   */
    "BROWSE DATASET('"sysdsname"("member")') VOLUME("sysvolume")"
    browse_rc = rc                              /* rc=Ø when found     */
    If browse_rc > Ø Then
       "DISPLAY PANEL(ISRBROM)"                 /* dummy display       */
    "CONTROL DISPLAY RESTORE"                   /* restore environment */

    Return browse_rc                            /* rc=14,16: not found */

/*===================================================================*/
/* Create a list of the datasets in the specified file allocation   */
/*   - this does not get the volser for each dataset.               */
/*-----------------------------------*/
FIND_DSNLIST: Arg ddfind          /* the DDNAME to find             */
trace O
    If var.Ø = 'VAR.Ø' Then Do    /* do LISTA only once!            */
       x = Outtrap('var.')        /* trap TSO messages in stem var. */
       Address TSO "LISTA ST"     /* list dataset allocations in TSO */
       x = Outtrap('OFF')         /* end message trapping           */
       End

    ddname = ''
    dsnlist = ''
    Do i = 2 To var.Ø By 2        /* process the trapped messages   */
       parse var var.i dsn .
       If dsn = 'TERMFILE' ! dsn = 'NULLFILE' Then i = i - 1
       Else Do
          j = i + 1
          Parse Var var.j newdd disp
          If disp <> ' ' Then
             ddname = Strip(newdd)   /* get a new DDNAME            */
          End
       If ddname = ddfind Then       /* If it's the DDNAME we want  */
          dsnlist = dsnlist dsn      /* .... add the dsn to dsnlist */
       Else   /* ddname <> ddfind */
          If dsnlist <> '' Then      /* if dsnlist has been created */
             Return Ø
       End

    If dsnlist <> ''
       Then Return Ø                 /* dsnlist created             */
       Else Return 8                 /* dsnlist not created         */

/*===================================================================*/
/* Look for the member in the list of (cataloged) datasets          */
/*-----------------------------------*/
FIND_MEMBER:

    If dsnlist = '' Then Return 1Ø
    "CONTROL DISPLAY SAVE"                   /* save what we have   */

    Do i = 1 To Words(dsnlist)
       dsn = Word(dsnlist,i)
       "CONTROL NONDISPL END"                /* pending 'END' cmd   */
       "BROWSE DATASET('"dsn"("member")')"
       browse_rc = rc                        /* rc=Ø when found     */
       If browse_rc = Ø Then Leave
       End

    If browse_rc > Ø Then
       "DISPLAY PANEL(ISRBROM)"              /* dummy display       */
    "CONTROL DISPLAY RESTORE"                /* restore environment */

    Return browse_rc                         /* rc=14,16: not found */
```

BOOKSRCH EXEC

```
/*=========================>> REXX <<================================*/
/*                                                                   */
/*  BOOKSRCH   : Invoke BookManager to search for a string          */
/*                                                                   */
/*             This is invoked by the DS EXEC                       */
/*                                                                   */
/*  Version    : 2.1                                                 */
/*===================================================================*/
  Address ISPEXEC                   /* commands go to ISPEXEC        */

  Parse Arg bkshelf textstring      /* get shelf and search argument  */

  If bkshelf = 'ALL' Then Do        /* ALL bookshelves               */
     bkshelf = ''
     If textstring <> '' Then Do
        textstring = ''
        msg = '   (search not possible until you select a bookshelf)'
```

```
         ZERRLM = msg
         ZERRTP = 'WARNING'
         "SETMSG MSG(ISRZØØ3) COND"
         End
      End
   If textstring = ''
      Then srch = ''
      Else srch = "CMD(SEARCH "textstring")"

   "SELECT CMD(%MSGLIBS ALLOC)"      /* allocate BookManager libraries */

   "CONTROL NONDISPL ENTER"          /* simulate ENTER key             */

   "SELECT CMD(%EOXVSTRT",
               bkshelf srch ")",     /* bookshelf and search argument  */
              "MODE(FSCR) SUSPEND",
              "NEWAPPL(EOXR) PASSLIB"

   "SELECT CMD(%MSGLIBS FREE)"       /* deallocate libraries           */

   Return
```

IMPLEMENTATION

The following information provides a useful guide to the implementation process.

COPY EXECS

As usual, copy the EXECS to a dataset allocated to SYSEXEC ddname. Note that DS needs the MSGLIBS and CSRWORD EXECs from the first part of this topic in *MVS Update* 151.

The DS EXEC has actions ‘Q’ and ‘ST’ commented out. That is because they require program DA$ENQS and the product StarTools, which are not available at most sites. Even if you do not use these actions, that code gives good examples of invoking other products.

Many of these EXECs can be used independently from DS. For example: DSLIST can be used via its own ISPF command, and FINDMEM could be used by any EXEC searching for a member in a dataset concatenation.

COPY PANELS

Copy all the panels shown here to a panel library. (You could also include all the panels from the article *Customising edit/browse panels* from *MVS Update* 150 in the same library.) Either allocate it to ddname ISPPLIB or let the DS EXEC use it via a LIBDEF. Customize the start of the DS EXEC appropriately to specify your library name for the LIBDEF statement.

DEFINE COMMAND

A DS command should be defined in an ISPF command table as follows:

```
Command    Description / Action
───    ───────────────────────

  DS      cursor-sensitive dataset actions
               SELECT CMD(%DS &ZPARM) NEWPOOL
```

If the ‘DS’ command name (verb) conflicts with any ISPF menu selection option you may want to change it to something else. That would cause no problem, (but remember to update the DSHELP panel appropriately). In any case, use the action as shown.

If you are using ISPF before Version 4.5 you should use action:

```
SELECT CMD(%DS &ZPARM) NEWPOOL SUSPEND
```

To remove any existing pop-up window when you invoke DS. Otherwise, you can always use the ‘RESIZE’ command to remove pop-up windows.

If you do not define a command, you must type ‘TSO %DS’ on the command line each time, or perhaps define a PF key with ‘TSO %DS’.

Note that the ‘%’ before the ‘DS’ is required. Otherwise the code gets the wrong address for the screen buffer.

DEFINE PF KEY(S)

To save typing, it can be convenient to define a PF key with the DS command. In addition, PF keys will even work from a panel without a command line, thus you can use the DS command from EVERY panel, using the default action.

Therefore, enter ‘KEYS’ to go into the ‘Keylist Utility’, then enter ‘DS’ in one of the definitions. I normally use PF4 for this since it is usually not defined. Remember that ISPF Version 4 uses many keylists, hence you may have to define it in a few of them to cover the variety of places that DS is used.

If you have not defined a PF key, you must type ‘DS’ on the command line (then position the cursor on the dataset-name) and press ENTER. Alternately, type ‘DS (action)’ on the command line, then double-click on the dataset name to invoke it.

CONCLUSION

Cursor-sensitive commands can provide some extremely useful functionality.

The DS command provides the user with new ways to invoke various ISPF functions for datasets. It works recursively from (almost) any ISPF panel. For specifying the dataset name, you can use either the cursor position or do it by parameters on the DS command.

Do not be put off by the size of this code. It is very simple to use, and it would be quite easy to add some of your own ‘actions’ to it. But to fully appreciate the usefulness of the code you really need to try it.

Ron Brown
Systems Progammer (Germany)

MVS news

Demand Technology has relaunched TapeSaver Release 2.2.0 on behalf of Technologic Software Inc. TapeSaver is an intelligent MVS tape stacking and consolidating utility which reduces the size of conventional tape or cartridge libraries by up to 40%. It is a migration aid for moving from reel to cartridge, or from a conventional library to ATL and/or a Virtual Tape Server. The DiskSaver sub-product stacks sequential disk files to tape.

TapeSaver requires a Library Management System such as CA-1, CA-Dynam/TLMS, Zara, or IBM DFSMSrmm, and will work with all the major MVS tape/cartridge and silo systems. TapeSaver Release 2.2.0 fully supports the new 3590 Image feature for the StorageTek 9840 Escon device for all TapeSaver tape stacking, volume moving and DiskSaver operations. TapeSaver is ISPF panel-driven and designed for ease of use, with no additional JCL needed. Once installed TapeSaver is run regularly as part of the installation's overnight batch work - typically in a OPC-ESA or CA-7 jobstream.

For further information contact:

Demand Technology UK Ltd,PO Box 156,Winchester, SO22 6ZE, UK.
Tel: (01962) 878 165
Fax: (01962) 878 278
http://www.demandtech.co.uk

Technologic Software Concepts Inc, University Tower, Suite 400,4199 Campus Drive,Irvine, California 92612-2698, USA.
Tel: (949) 509 5000
Fax: (949) 509 5015
http://www.technologic.com

* * *

NEON Systems has begun shipment of its NEON 24X7 management console, designed to provide a single point of control for managing the availability, integrity, and performance of IMS databases and replace the need for multiple software tools.

The proactive software runs continuously, notifying the DBA that a problem is developing, then runs the maintenance jobs to correct the problem. It is said to eliminate the need to run batch jobs and constantly analyse reports.

Specifically, it automates routine database administration tasks and enables administrators to perform reorganizations only when needed and before crises develop. It allows DBAs to refine database definitions based on each database's available space and other production variables. Through ISPF screens, DBAs can change NEON 24X7 default thresholds to fit their own IMS database requirements.

For further information contact:

NEON Systems Inc, 14141 Southwest Freeway, Suite 6200, Sugar Land, TX 77478, USA.
Tel: (281) 491 4200
Fax: (281) 242 3880

NEON Systems UK Ltd, Third Floor, Sovereign House, 26-30 London Road, Twickenham, Middlesex, TW1 3RW, UK.
Tel: (0181) 607 9911
Fax: (0181) 607 9933.
http://www.neonsys.com

* * *

xephon