In this issue

3  VSAM enhancements in DFSMS/MVS Version 1.4
10  KEYLIST – a utility to list VSAM keys
30  Testing to see whether a VSAM cluster is empty
34  Resetting a VSAM cluster
38  Updating VSAM definitions in the CSD
60  Organize your disks and claim Free Space
64  VSAM news

© Xephon plc 1998
Disclaimer
Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, EXECs, and other contents of this journal before making any use of it.

Subscriptions and back-issues
A year’s subscription to VSAM Update, comprising four quarterly issues, costs £120.00 in the UK; $180.00 in the USA and Canada; £126.00 in Europe; £133.00 in Australasia and Japan; and £130.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the April 1991 issue, are available separately to subscribers for £30.00 ($45.00) each including postage.

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

© Xephon plc 1998. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.
VSAM enhancements in DFSMS/MVS Version 1.4

With Version 1 Release 4 of DFSMS/MVS, IBM has made a number of VSAM enhancements and additions, namely:

- VSAM RLS KSDS extended addressability.
- VSAM system managed buffering.
- VSAM fast load.
- Updating the VSAM last reference date at close.
- Data class support for VSAM attributes.
- Catalog Search Interface (CSI).
- LSR resource pool change.
- Location of VSAM buffers and control blocks.

These are examined in turn below.

VSAM RLS KSDS EXTENDED ADDRESSABILITY

VSAM extended addressability allows VSAM datasets larger than 4GB to reside in Extended Format VSAM datasets. The external interface for VSAM extended addressability has been achieved by using two fullwords (8B) to hold a Relative Byte Address (RBA).

When IBM announced VSAM RLS support in DFSMS/MVS Version 1 Release 3, VSAM RLS still retained the 4GB architectural limit. DFSMS/MVS Version 1 Release 4 has removed this restriction by supporting RLS extended addressability for VSAM KSDSs. This allows VSAM RLS to support VSAM KSDSs up to the multi-volume limit of 59 DASD volumes. Note that a parallel sysplex environment is required in order to use VSAM RLS.

More information on VSAM KSDS extended addressability can be found in *DFSMS/MVS Version 1 Release 3 VSAM enhancements, VSAM Update* Issue 26 (July 1997).
VSAM SYSTEM MANAGED BUFFERING

One of the most important enhancements to VSAM in DFSMS/MVS Version 1 Release 4 is the new VSAM System Managed Buffers (SMB) facility. This enables VSAM to determine the optimum number of data and index buffers, as well as the type of buffer management to employ (i.e., sequential or direct).

SMB has one advantage over Batch LSR, which is that, where appropriate, a switch to Local Shared Resource (LSR) buffering occurs automatically and without any change to JCL.

For system managed buffering to occur, the following conditions must be met:

- The VSAM dataset must be in extended format.
- The ACB MACRF must be NSR. The ACB MACRF parameter must not contain LSR, GSR, RLS, ICI, AIX, or UBF.
- Either Record_Access_Bias must be set to SYSTEM in the SMS data class, or the JCL AMP parameter ACCBIAS must be set to SYSTEM, SW, DO, or DW. If the dataset is not in extended format (EF), Record_Access_Bias is ignored.

In the data class, REC_ACC_BIAS is a new sub-parameter for a DATA SET NAME TYPE of EXT. This can be specified as either USER or SYSTEM.

A new JCL AMP sub-parameter, ACCBIAS, can be used to specify access bias. This sub-parameter can have one of six specifications. These are:

- USER – bypass SMB. USER indicates that VSAM will continue to use buffers as it currently does without SMB.
- SYSTEM – this option will force SMB and allow the system to determine the buffering technique, according to the ACB MACRF (SEQ, DIR, SKP) parameter and storage class specifications. A value of SYSTEM specifies that VSAM is to determine the number of buffers to obtain for the dataset, when NSR processing is used. If VSAM chooses direct optimized (DO) as the most appropriate type of access, and NSR has been specified or defaulted, the buffering technique is changed from NSR to LSR.
When LSR buffer management is chosen, VSAM will also determine the number of virtual storage buffers to use.

- **SO** – SMB with sequential optimization.
- **SW** – SMB weighted for sequential processing. When SW is specified, most buffers will be used to support sequential processing, but some will be reserved for index buffers to help any direct processing.
- **DO** – SMB with direct optimization. This option will force a switch to LSR. When SMB converts NSR buffering to LSR buffering, three new optional AMP parameters can be specified to tell LSR buffer management how to handle the processing of the buffers. The three new sub-parameters are:
  - **SMBVSP** – specifies the amount of virtual storage to obtain for buffers when opening the dataset. The value specified is the total amount of virtual storage that can be addressed in a single address space. It does not take into account the storage required by the system or the access method. The sub-parameter is specified as follows:
    \[ \text{SMBVSP}=\text{xxK} \mid \text{SMSVSP}=\text{xxM} \]
  - **SMBHWT** – the amount of hiperspace to be used for LSR buffers. This can be specified with the SMBHWT sub-parameter of AMP. The value specified for SMBHWT is used as the hiperspace weighting factor for the number of hiperspace buffers to be established. The hiperspace buffer size will be a multiple of 4096(4K). The format of the SMBHWT parameter is as follows:
    \[ \text{SMBHWT}=\text{nn} \]
    where \( \text{nn} \) is a number between 1 and 99.
  - **SMBDFR** – can be deferred until the buffer is required for a different request or the dataset is closed. The sub-parameter is specified as follows:
    \[ \text{SMBDFR}=\text{Y}|\text{N} \]
The default for SHAREOPTIONS(1,3) and (2,3) is Y.
The default for SHAREOPTIONS(3,3), (4,3), and (x,4) is N.

- DW – SMB weighted for direct processing. When DW is specified, most buffers will be used to support fast direct access to the data, with relatively few buffers reserved for any sequential processing which might occur.

Note that:
- Specifying the type of Record Access Bias through the JCL AMP parameter will override anything specified in the SMS data class.
- If nothing has been specified for this parameter, the default is USER.

**VSAM FAST LOAD**

IBM has improved the performance of loading an extended format VSAM KSDS by reducing the number of I/O requests required to write the data. More information on extended format VSAM KSDSs can be found in *Enhanced VSAM support in DFSMS/MVS 1.2.0, VSAM Update Issue 20* (January 1996).

The following conditions must be met in order to use VSAM fast load:

- The VSAM KSDS must be in extended format.
- System Managed Buffers (SMB) must be requested in the data class or the JCL AMP parameter.
- The VSAM KSDS must be defined with the SPEED parameter. The SPEED parameter is specified in the IDCAMS DEFINE command or an SMS data class.

System Managed Buffering allows sufficient data buffers to be acquired in order to write each control area (CA) with a single I/O request. Previous releases required at least two I/O requests to write a control area, and more if the FREESPACE parameter had been specified for the VSAM dataset. With the new load implementation, the index component should be updated only once per data CA. In previous releases, the index was updated many times per CA.
DATA CLASS SUPPORT FOR VSAM ATTRIBUTES

The following VSAM dataset attributes can now be specified in an SMS data class:

- BWO (back up while open).
- LOG.
- LOGSTREAM ID.
- SPANNED/NONSPANNED attribute.

With DFSMS/MVS Version 1 Release 4, it is now possible to use JCL to define any VSAM dataset with all its related attributes, thus eliminating a separate IDCAMS DEFINE or ALTER step. VSAM partial space release, introduced in DFSMS/MVS Version 1 Release 2, is still supported only for extended format VSAM KSDSs to release over-allocated space, and is specified using JCL or an SMS management class parameter by:

- Coding the SPACE=(,,,(RLSE)) JCL parameter in a DD statement.
- Assigning a management class with partial release values of CI for Conditional Immediate and YI for Yes Immediate.

When a VSAM dataset is created, SMS will propagate the data class values for the attributes only if they apply to the VSAM dataset type. Figure 1 shows how to create a VSAM dataset using JCL.

```
DATA CLASS= VSAMDC1

KSDS
EXTENDED FORMAT
RECORDSIZE
SPEED
SPANNED
SPACE

//VSAM DD DSN=VSAM.JCL.DATA SET,

Figure 1: Creating a VSAM dataset using JCL
```
UPDATING THE VSAM LAST REFERENCE DATE (LRD) AT CLOSE

Before DFSMS/MVS Version 1 Release 4, the last reference date for VSAM datasets was updated at OPEN time. This presented a major problem for systems like CICS, because datasets which had been open for a number of days could all be eligible for migration by DFSMSShsm when the CICS system was stopped.

With Version 1 Release 4, the last reference date is now updated in the FORMAT-1 DSCB, on the first volume for the base component of a VSAM sphere, when the VSAM dataset is closed as well as at open time. This brings VSAM in line with what happens for non-VSAM datasets.

The following conditions must be met for the update to occur:

- The current date must be greater than the date on which the dataset was opened.
- The current date must be greater than the DS1REFD date in the FORMAT-1 DSCB.

Date stamp processing for close compares the date on which the VSAM dataset was opened with the date on which it is closed, to determine whether the date has changed. For a non-RLS VSAM dataset, the IDATMSTP (date stamp) routine is called during OPEN processing to retrieve a return code that specifies whether or not the date in the VTOC is to be changed. VSAM keeps this information until the VSAM dataset is closed. For VSAM RLS, date stamp processing is always performed.

CATALOG SEARCH INTERFACE

The Catalog Search Interface (CSI) was originally developed by IBM as an MVS read-only general purpose interface, to enable user application programs to extract data from ICF catalogs. It has been incorporated free of charge into the DFSMSdfp component of DFSMS/MVS Version 1 Release 4.

The CSI supports search keys containing ‘wild card’ specifications, so that information on multiple entries can be returned. The type or types of entries required can also be specified. Because field information
from entries contained in the ICF catalog is requested by specifying field names, the caller doesn’t need to know whether the information is in the Basic Catalog Structure (BCS) or in the VSAM Volume Dataset (VVDS).

The CSI can be used for a number of installation-provided facilities, such as:

- A tailored LISTCAT designed for the needs of an installation.
- Automatically determining when VSAM datasets should be reorganized.
- Obtaining performance data for VSAM datasets.
- Detecting down-level catalogs after volume recovery.
- Tailoring for DFSMSdss VSAM back-ups. This could include ICF catalog back-ups.

The CSI can be invoked as follows:

- 24-bit or 31-bit addressing mode.
- In any protection key.
- In either Supervisor or Problem State mode.

More information on the CSI can be found in *Methods of extracting VSAM information, VSAM Update Issue 21* (April 1996).

**LSR RESOURCE POOL CHANGE**

Before DFSMS/MVS Version 1 Release 4, the LSR resource pool specifications allowed each address space to allocate up to 16 index

![Figure 2: Values for RMODE31 with the JCL AMP parameter or the ACB macro](image-url)
resource pools and up to 16 data resource pools. The new release increases the number of LSR resource pools from 16 to 256.

LOCATION OF VSAM BUFFERS AND CONTROL BLOCKS
A new JCL AMP parameter, RMODE31, has been provided to allow the user to specify the location of buffers and control blocks. This will override any values specified in the corresponding parameter in the ACB macro. The values that can be specified for RMODE31 with the JCL AMP parameter or the ACB macro are shown in Figure 2.

KEYLIST – a utility to list VSAM keys

INTRODUCTION
The program presented here lists the keys from VSAM KSDS files. It has three major functions:

- **LIST**, which lists the keys from a single file (INPUT1). This is the default option.
- **MATCH**, which lists the keys that are contained in each of two files (INPUT1 and INPUT2).
- **UNIQUE**, which lists the keys that are contained in one file (INPUT1) but not in another file (INPUT2).

The listing may be in character format (default) or vertical hexadecimal format. In vertical hexadecimal, three lines are used to display a key: the first is the character, the second displays the zone nibble (bits 0-3) and is indicated by a Z in print position 1, and the third displays the numeric nibble (bits 4-7) and is indicated by an N in print position 1.

The above options are specified by PARM= parameters, as follows:

- **PARM=‘OPTION=LIST’** or no PARM results in a character listing of keys from a single file.
• PARM='HEX,OPTION=List' or PARM='HEX' results in a vertical hexadecimal listing of the keys from a single file.

• PARM='OPTION=Match' results in a character listing of the keys common to both files.

• PARM='HEX,OPTION=Match' results in a vertical hexadecimal listing of the keys common to both files.

• PARM='OPTION=Unique' results in a character listing of keys contained in the first file and not in the second.

• PARM='HEX,OPTION=Unique' results in a vertical hexadecimal listing of the keys found in the first file but not in the second.

When the ‘MATCH’ option is used, an asterisk (‘*’) is placed to the right of the displayed key if the records within the two files are identical.

```plaintext
//jobnamex JOB ...
/*/--------------------------------------------------------------------*/
/*/ COMPARE KEYS OF VSAM FILES                                      */
/*/--------------------------------------------------------------------*/
/S1   EXEC PGM=KEYLIST PARM='OPTION=MATT'
/STEPLIB DD DSN=MPAC2.MTST.LOADLIB,DISP=SHR
/SYSUDUMP DD SYSOUT=* 
/PRINTER DD SYSOUT=* 
/INPUT2 DD DSN=ADSPLUS.R6Ø.VSAM.AESSCR$,DISP=SHR
/INPUT1 DD DSN=ADSPLUS.R6Ø.VSAM.MTST.AESSCR$,DISP=SHR
/RANGES DD *,DCB=LRECL=8Ø
MAXL 8  RESTRICT TO FIRST 8 BYTES OF KEY
FROM AESATQJS  IGNORE ALL KEYS PRIOR TO AESATQJS*
THRU AEXHED4S  AND AFTER AEXHED4S*
FIND CEDFEXIT   UNTIL CEDFEXIT*
FROM YAIIDNZH  IGNORE ALL KEYS PRIOR TO YAIIDNZH*
THRU YA2Ø4NZ  AND AFTER YA2Ø4NZ* PLUS
EXCL YAINSZH   YAINSZH UNTIL YAINSZH*
FIND YA2Ø4NZH  ALLOW ABOVE 'THRU' TO COMPLETE
FROM%YU%      IGNORE ALL KEYS PRIOR TO YU*
THRU@9@       AND AFTER 9*
EXCL'YUH'     AND KEYS YUH*
NOTE:  THE ABOVE ASTERISKS ('*') INDICATE THE END OF A GENERIC KEY.
/*
//
```

Figure 1: KEYLIST run-time JCL
EXCLUSIONS

The dataset RANGES may contain records that limit the above listing to specific keys. The record size of RANGES (LRECL) must not exceed 261 (maximum key length plus five). The format of the records is ‘xxxxyzzz...zzzz’, where:

- xxxx is a code (‘FROM’, ‘THRU’, ‘EXCL’, ‘FIND’, or ‘MAXL’) to describe the desired exclusion or inclusion. This value should be in the first through the fourth position of the record.

- zzzz...zzzz is an EBCDIC character string that is used to compare against the VSAM keys. Use the ISPF EDIT function HEX ON (or a similar function) to enter non-keyable characters. This

KEYLIST - LIST VSAM KEYS.

<table>
<thead>
<tr>
<th>KEYLIST - LIST VSAM KEYS.</th>
<th>11/28/97 PAGE 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>AESATQJS YAP01NZH YAS01NZH YAS29NZH YA110NZ YUARSNZ YUFNCNZH YURELNZ YUUDCNZ</td>
<td></td>
</tr>
<tr>
<td>AESDBUGS YAP02NZH YAS02NZH YAS31NZ YA110NZH YUARSNZ YUGENNZ YURELNZH YUUDCNZH</td>
<td></td>
</tr>
<tr>
<td>AEXHED1S YAP02NZH YAS02NZH YAS31NZH YA111NZ YUARTNZ YUGENNZ YURFDNZ YUUDVNZ</td>
<td></td>
</tr>
<tr>
<td>AEXHED2S YAP03NZH YAS03NZ YAS32NZ YA120NZ YUARTNZ YUGRPNZ YURFDNZH YUUDVNZH</td>
<td></td>
</tr>
<tr>
<td>AEXHED3S YAP03NZH YAS03NZH YAS32NZH YA120NZH YUBEDNZ YUGRPNZH YURLGNZ YUVIPNZ</td>
<td></td>
</tr>
<tr>
<td>AEXHED4S YAP04NZH YAS04NZ YAS33NZ YA121NZ YUBEDNZH YUICCNZH YURLGNZH YUVIPNZH</td>
<td></td>
</tr>
<tr>
<td>YAI1DNZ YAP04NZH YAS04NZH YAS33NZH YA125NZ YUCGRNZ YUINFNZ YURLTNZ YUVISNZ</td>
<td></td>
</tr>
<tr>
<td>YAINFNZ YAP05NZH YAS05NZ YAS34NZ YA125NZH YUCGRNZH YUINFNZH YURLTNZH YUVISNZH</td>
<td></td>
</tr>
<tr>
<td>YAINSNZ YAP05NZH YAS05NZH YAS34NZH YA126NZ YUCHGNZ YUINTNZ YURMCNZ YUVIPNZ</td>
<td></td>
</tr>
<tr>
<td>YAIPNSNZ YAP06NZH YAS06NZ YAS35NZ YA130NZ YUCHGNZH YUINTNZH YURMCNZH YUZIPNZ</td>
<td></td>
</tr>
<tr>
<td>YAIFPNZH YAP06NZH YAS06NZH YAS35NZH YA130NZH YUCH1NZ YUISONZ YURGRNZ YU900NZ</td>
<td></td>
</tr>
<tr>
<td>YALICNZH YAP07NZH YAS07NZ YAS36NZ YA131NZ YUCH1NZH YUISONZH YURGRNZH YU900NZH</td>
<td></td>
</tr>
<tr>
<td>YAL0ANZH YAP07NZH YAS07NZH YAS36NZH YA140NZ YUCILNZ YULINIZ YURSKNZ YU901NZ</td>
<td></td>
</tr>
<tr>
<td>YAL0SNZ YAP08NZH YAS08NZ YAS37NZ YA140NZH YUCILNZH YULINIZH YURSKNZH YU901NZH</td>
<td></td>
</tr>
<tr>
<td>YMCDNZN YAP09NZH YAS08NZH YAS37NZH YA141NZ YUCLSNZ YULNGNZ YURS3NZ YU902NZ</td>
<td></td>
</tr>
<tr>
<td>YAMC1NZH YAP09NZH YAS09NZ YAS43NZ YA150NZ YUCLSNZH YULNGNZH YURS3NZH YU902NZH</td>
<td></td>
</tr>
<tr>
<td>YAMC2NZH YAP09NZH YAS09NZ YAS43NZH YA150NZH YUCLUDNZ YULANZ YUR54NZ YU903NZ</td>
<td></td>
</tr>
<tr>
<td>YAMC3NZH YAP10NZH YAS10NZ YAS44NZ YA151NZ YUCDNNZH YULANZH YUR54NZH YU903NZH</td>
<td></td>
</tr>
<tr>
<td>YAMC4NZH YAP10NZH YAS10NZH YAS44NZH YA160NZ YUCNSNZ YUCLUDNZH YUR55NZ YU904NZ</td>
<td></td>
</tr>
<tr>
<td>YAMC5NZH YAP10NZH YAS10NZH YAS44NZH YA160NZH YUCNSNZH YULDCNZ YUR55NZH YU904NZH</td>
<td></td>
</tr>
<tr>
<td>YAMC6NZH YAP11NZH YAS11NZ YAS45NZH YA161NZ YUCNTNZ YULDCNZH YURVWNZ</td>
<td></td>
</tr>
<tr>
<td>YAMC6NZH YAP11NZH YAS11NZH YAS45NZH YA161NZH YUCNTNZH YULDCNZH YURVWNZH</td>
<td></td>
</tr>
<tr>
<td>YAMC7NZH YAP12NZH YAS12NZ YAS46NZH YA170NZ YUCTRNZ YULWXNZ YURVWNZH</td>
<td></td>
</tr>
<tr>
<td>YAMC8NZH YAP12NZH YAS12NZH YAS46NZH YA170NZH YUCTRNZH YULWXNZH YUR1DNZ</td>
<td></td>
</tr>
<tr>
<td>YAMC9NZH YAP13NZ YAS13NZ YAS47NZ YA171NZ YUCSTNZH YUMARNZ YUR1DNZH</td>
<td></td>
</tr>
<tr>
<td>YAMC10NZ YAP13NZ YAS13NZ YAS47NZ YA171NZH YUCSTNZH YUMARNZH YU5DSNZ</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: KEYLIST sample output – list of keys
string should begin in the sixth position of the record.

- y is a character to indicate the ending of the character string (eg ‘THIS STRING CONTAINS BLANKS’ or *THIS STRING CONTAINS BLANKS AND ’ CHARACTERS*). A blank or space (X’40’) may be used if there are no embedded blanks in the string.

The ‘FROM’ statement excludes all keys whose values are less than the specified character string. As with the other statement types, a character string whose length exceeds the key size is truncated to the key size. Alternatively, the comparison is limited to the left-most characters contained in the character string.

The ‘THRU’ statement excludes all keys whose values are greater than the specified character string.
The ‘EXCL’ statement is used to exclude all keys that are equal to the character string. Once a key is found that is greater than the specified string, additional records are read from the RANGES dataset. At this time, additional ‘FROM’ and/or ‘THRU’ strings may be specified. Note that the character strings of ‘EXCL’ and ‘FIND’ statements must be specified in ascending sequence.

The ‘FIND’ statement is like the ‘EXCL’ statement, except that no inclusion is made. Its purpose is to provide a means of reading additional ‘FROM’ and ‘THRU’ key ranges. Note that if an ‘EXCL’ is used to exclude records from a previous ‘FROM-THRU’ definition, and is followed by another ‘FROM-THRU’ definition, the latter definition would be activated after the ‘EXCL’ values are passed. One way of avoiding this is to add a ‘FIND’ statement, with a string that matches or is greater than that of the preceding ‘THRU’, after the ‘EXCL’ and before the next ‘FROM-THRU’ definition (see ‘FIND YA204NZH’ in Figure 1 for an example of this usage).

The ‘MAXL’ statement is used to specify the maximum portion of the key to be used. This option excludes all records from the file(s) that are the same for the specified length. In this statement, the string (‘zzzz...zzzz’) is expected to be a decimal value from 1 to the shortest key of the file(s). If used, this statement should be the first statement in the RANGES dataset.

Figures 2 and 3 show sample output of the program.

PROGRAM SOURCE

LCLC &MYNAME
*
&MYNAME SETC 'KEYLIST' CSECT NAME
RBASEx EQUI 12 BASE REGISTER FOR CSECT
RBASE2 EQU 8 SECOND BASE REGISTER FOR CSECT
RBAL EQU 10 BAL REGISTER
*
TITLE '&MYNAME' LISTING TITLE
*********************************************************************
***
*** THIS PROGRAMS PERFORMS VARIOUS LISTINGS OF VSAM KEYS.
***
*** CONTROL IS BY PARM='HEX,OPTION=XXXXX', WHERE:
***

© 1998. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
1) 'HEX' IS SPECIFIED IF THE LISTING IS TO BE IN VERTICAL HEXADECIMAL. IF THIS PARAMETER IS NOT PRESENT THEN THE LISTING IS CHARACTER ONLY.

2) OPTION=MATCH PROVIDES LISTING OF THE KEYS THAT ARE CONTAINED IN BOTH INPUT1 AND INPUT2.

3) OPTION=UNIQUE PROVIDES LISTING OF THE KEYS THAT ARE CONTAINED IN INPUT1 BUT NOT INPUT2. IF THIS OPTION IS SPECIFIED AN '*' BEFORE THE KEY INDICATES THAT THE RECORDS ARE ALSO IDENTICAL.

4) OPTION=LIST PROVIDES LISTING OF THE KEYS OF FILE INPUT1. (INPUT2 IS NOT DEFINED).

IF 'HEX' IS PRESENT IT MUST BE IN POSITION 1-3 AND 'OPTION=' BEGINS IN POSITION 5. ELSE 'OPTION=' BEGINS IN POSITION 1.

IF 'OPTION' IS NOT SPECIFIED, 'LIST' IS ASSUMED.

*****************************************************************************

EJECT

*****************************************************************************

** LINKAGE CONVENTIONS ENTERING PROGRAM

*****************************************************************************

&MYNAME CSECT ,
    STM R14,R12,12(R13) SAVE REGS TO CALLER S.A.
    B (BEGIN-&MYNAME)(R15) BRANCH AROUND EYECATCHER
    DC A(L'NAME) LENGTH OF CSECT NAME
    NAME DC C'&MYNAME' CSECT NAME
    DC C' &SYSDATE &SYSTIME ' ASSEMBLY DATE/TIME STAMP
BEGIN LR RBASE,R15 LOAD BASE REGISTER
    LA RBASE2,2048(RBASE) RBASE + 2048
    LA RBASE2,2048(RBASE2) RBASE + 4096
    USING &MYNAME,RBASE,RBASE2 ADDRESSABILITY
    PRINT NOGEN
    GETMAIN R,LV=WORKDLEN GET SAVE/WORK AREA
    ST R1,8(Ø,R13) MY S.A. ADDR INTO CALLER S.A.
    ST R13,4(Ø,R1) CALLER S.A. ADDR INTO MY S.A.
    LR R13,R1 R13 POINTS TO MY S.A.
    USING WORKD,R13 ADDRESSABILITY OF SAVE AREA
    L R1,4(Ø,R13) R1 POINTS TO CALLER S.A.
    LM R15,R1,16(R1) R15 RØ AND R1 ARE RESTORED
* EJECT

*****************************************************************************

*** MAINLINE ROUTINE ***

MAIN EQU * BEGIN MAINLINE ROUTINE
ST R1,R1SAVE SAVE INITIAL R1
XC COMPCODE,COMPCODE CLEAR COMPLETION CODE
*
MVC JGMOTBL(13*L'JGMOTBL),JGMOTBLD COPY JULGREG DAYS/MONTH
*
BEGIN DCB INITIALIZATION
*
MVC PRINTER(PRINTERL),PRINTERD INITIALIZE DCB
MVC INPUT1(INPUT1L),INPUT1D INITIALIZE ACB
MVC INPUT2(INPUT2L),INPUT2D INITIALIZE ACB
*
MVC RANGES(RANGESL),RANGESD INITIALIZE RANGES DCB
*
END DCB INITIALIZATION
*
*
BEGIN DCB OPENS
*
MVC PROPENL(PROPENLN),OPEND INITIALIZE SET PRINTER OPEN LIST
OPEN (PRINTER,(OUTPUT)),MF=(E,PROPENL) OPEN PRINTER
*
MVC RGOPENL(RGOPENLN),OPEND SET RANGES OPEN LIST
OPEN (RANGES,(INPUT)),MF=(E,RGOPENL) OPEN RANGES
*
END DCB OPENS
*
*
MVI IDENT,C' ' CLEAR 'IDENTICAL' FLAG
ZAP IDENTS,=P'Ø' INITIALIZE 'IDENTICAL' RECORD COUNT
MVI EOFFLAGS,Ø CLEAR E-O-F FLAGS
MVC HEADER(L'HEAD),HEAD INITIALIZE HEADER
MVC HEADER+(L'HEADER-L'HEAD),HEADER+L'HEAD-1 CLEAR
MVC PAGENO-4(4),=-C'PAGE' SET PAGE NUMBER ID
ZAP COUNT1,=P'Ø' INITIALIZE INPUT1 VSAM RECORD COUNT
ZAP COUNT2,=P'Ø' INITIALIZE INPUT2 VSAM RECORD COUNT
ZAP COUNTDUP,=P'Ø' INITIALIZE DUPLICATE KEY COUNT
ZAP COUNTUNQ,=P'Ø' INITIALIZE UNIQUE KEY COUNT
ZAP COUNT1F,=P'Ø' INITIALIZE FROM KEY COUNT FOR INPUT1
ZAP COUNT1T,=P'Ø' " THRU KEY COUNT
ZAP COUNT1E,=P'Ø' " EXCLUDE KEY COUNT
ZAP COUNT1M,=P'Ø' " MAXL EXCLUSIONS
ZAP COUNT2F,=P'Ø' INITIALIZE FROM KEY COUNT FOR
ZAP COUNT2T,='Ø' " THRU KEY COUNT
ZAP COUNT2E,='Ø' " EXCLUDE KEY COUNT
ZAP COUNT2M,='Ø' " MAXL EXCLUSIONS
ZAP MAXKEYL,='Ø' MAXIMUM KEY LENGTH
ZAP PAGES,='1' INITIALIZE PAGE COUNT

TIME
ST R1,JGYYDDD SAVE JULIAN DATE
BAL RBAL,JULGREG CONVERT TO JULIAN DATE TO GREGDATE
MVC HEADDATE,JGMMDDYY MOVE MM/DD/YY TO HEADER

* BAL RBAL,GETPARMS GO PROCESS PARM=
* LA R2,IN1RPL POINT TO ACB
* BAL RBAL,OPENVSAM GO BUILD RPL, ACB, OPEN INPUT1
* L R1,IN1KEYL GET KEY LENGTH FOR INPUT1
* TM OPTIONS,LISTBIT INPUT2 PROCESSING?
  BO SETMINKL NO
* LA R2,IN2RPL POINT TO ACB
  BAL RBAL,OPENVSAM GO BUILD RPL, ACB, OPEN INPUT2
* L R1,IN1KEYL GET KEY LENGTH FOR INPUT1
* C R1,IN2KEYL IS KEY OF INPUT1 > KEY OF INPUT2
  BNL SETMINKL NO
* L R1,IN2KEYL USE KEY LENGTH OF INPUT2
* SETMINKL STH R1,KEYLENMN SAVE KEY LENGTH
* LA R0,L'LINES-1 LENGTH-1 OF 'LINES' ENTRY
  CR R0,R1 LESS THAN KEY LENGTH?
  BL BIGKEY YES
  SR R0,R1 GET LAST POSSIBLE KEY POSITION
* BIGKEY STH R0,LASTCOL SAVE
* BCTR R1,Ø LENGTH-1
  STH R1,KEYLENM1 SAVE KEY LENGTH
* TM OPTIONS,LISTBIT LIST KEYS FROM INPUT1?
  BO DOHEAD YES
* TM OPTIONS,MATCHBIT MATCHING KEYS?
BZ DOUNQ NO
* MVC HEADER+L'HEAD-6(14),=C'MATCHING KEYS.' MODIFY HEADER
B DOHEAD GO PRINT PAGE HEADING
* DOUNQ MVC HEADER+L'HEAD-6(12),=C'UNIQUE KEYS.' MODIFY HEADER
* DOHEAD BAL RBAL,HEADPAGE PRINT PAGE HEADER
* BAL RBAL,GETRANGE GO READ RANGES FOR FROM,THRU,& EXCL
*
CP MAXKEYL.=P'Ω' WAS MAXIMUM KEY LENGTH SPECIFIED?
BE NOMAX NO
*
ZAP DOUBLE,MAXKEYL MOVE TO ALIGNED AREA
CVB R1,DOUBLE CONVERT TO BINARY
CH R1,KEYLENMN DOES IT EQUAL OR EXCEED ACTUAL?
BNL NOMAX YES
STH R1,KEYLENMN SAVE MAX
BCTR R1,Ø DECREMENT
STH R1,KEYLENM1 SAVE MAX-1
OI OPTIONS,MAXKBIT FLAG OPTION
*
NOMAX BAL RBAL,CLRPAGE CLEAR PAGE STORAGE AREA
*
BAL RBAL,DOMATCH IF OPTION NE LIST, PRINT (NO)MATCHES
*
BAL RBAL,DOLIST IF OPTION=LIST PRINT KEYS (INPUT1)
*
BAL RBAL,PRTPAGE PRINT LAST PAGE
*
BAL RBAL,DOTOTALS LINK TO DOTOTALS
*
* BEGIN DCB CLOSE
*
CLOSE MVC PRCLOS(PRCLOS LN),CLOSED INITIALIZE CLOSE LIST
CLOSE (PRINTER),MF=(E,PRCLOS) CLOSE IT
*
MVC RGCLOSE(RGCLOSE LN),CLOSED SET RANGES CLOSE LIST
CLOSE (RANGES),MF=(E,RGCLOSE) CLOSE RANGES
*
LA R2,INPUT1 POINT TO INPUT1 ACB
BAL RBAL,CLOSVSAM GO CLOSE INPUT1
*
TM OPTIONS,LISTBIT WAS INPUT2 OPENED?
BO ENDØØ NO
*
LA R2,INPUT2 POINT TO INPUT2 ACB
BAL RBAL,CLOSVSAM GO CLOSE INPUT2
*   END DCB CLOSE
*
ENDØØ LA R15,Ø                SET COMPLETION CODE ØØ
ST R15,COMPCode               INTO STORAGE
B ENDING GO TO ENDING
*
EJECT
***************************************************************************
***                                                      ***
***   LINKAGE CONVENTIONS EXITING PROGRAM                     ***
***                                                      ***
***************************************************************************
ENDING L R14,COMPCode             R14 SAVES COMP CODE
LR R1,R13                   R1 SAVES ADDR OF MY S.A.
L R13,4(Ø,R1)              R13 RESTORED, PTR CALLER S.A.
FREEMAIN R,LV=WORKDLEN,A=(R1)  FREE MY SAVE/WORK AREA
LR R15,R14                  R15 SET TO COMP CODE
LM RØ,R12,2Ø(R13)           RØ-R12 RESTORED
L R14,12(Ø,R13)            R14 RESTORED
MVI 12(R13),X'FF'            SET COMPLETION SIGNAL
BR R14                      RETURN TO CALLER
*
* BEGIN STUB DEFINE
*
*
EJECT
***************************************************************************
***                                                      ***
***   CONVERT JULIAN DATE TO GREGORIAN DATE                   ***
***                                                      ***
***************************************************************************
JULGREG ST RBAL,SAVJGBAL        SAVE LINKAGE REGISTER
*
CLI JGYYDDD,1                   IS ACTUAL CENTURY PRESENT?
BH JGACTUAL                YES
TR JGYYDDD(1),=X'192Ø'      CENTURY=Ø ==> 19XX, 1==>2ØXX
JGACTUAL ZAP JGDAYS,JGYYDDD+2(2)  SAVE DAYS FROM BEGINNING OF YEAR
ZAP JGMONTHS,=P'1'            INITIALIZE MONTH
*
LA R15,JANUARY                 LOAD ADDRESS OF DAYS/MONTH TABLE
LA RØ,L'JANUARY                ... WIDTH OF TABLE
LA R1,DECEMBER                ... END OF TABLE
*
ZAP FEBRUARY,=P'28'            SET NON LEAP YEAR DAYS
CLC =X'2ØØØ',JGYYDDD         YEAR 2ØØØ?
BE JGYR2ØØØ                YES
*
JG2ØTHCN TM JGYYDDD+1,1        LEAP YEAR?
BO JGLOOP                   NO
TM  JGYYDDD+1,X'12'
BM  JGLOOP   NO
JGYR2000 AP  FEBRUARY,-P'1' ADJUST
*
JGLOOP CP  JGDAYS,Ø(L'JANUARY,R15) CURRENT MONTH?
BNH JGFOUN  YES
AP  JGMONTHS,=P'1' INCREMENT MONTH
SP  JGDAYS,Ø(L'JANUARY,R15) DECREMENT DAYS PER CURRENT
    MONTH
BXLE R15,R0,JGLOOP CONTINUE
*
JGFOUN UNPK JGMMDDYY(2),JGMONTHS UNPACK MONTH
    UNPK JGMMDDYY+3(2),JGDAYS UNPACK DAY
    UNPK JGMMDDYY+6(3),JGYYDDD+1(2) UNPACK YEAR
    MVI JGMMDDYY+2,C'//' SEPARATE MONTH AND DAY
    MVI JGMMDDYY+5,C'//' SEPARATE DAY AND YEAR
    OI JGMMDDYY+1,C'Ø' FORCE MONTH NUMERIC
    OI JGMMDDYY+4,C'Ø' FORCE DAY NUMERIC
    OI JGMMDDYY+7,C'Ø' FORCE YEAR NUMERIC
*
JGRETURN L  RBAL,SAVJGBAL LOAD LINKAGE REGISTER
BR  RBAL RETURN
*
EJECT
*********************************************************************
***                                                               ***
***   MATCH KEYS                                                  ***
***                                                               ***
*********************************************************************
*
DOMATCH ST  RBAL,SAVDMBAL SAVE LINKAGE REGISTER
*
    TM  OPTIONS,LISTBIT OPTION=MATCH OR MATCH=UNIQUE?
    BO  DMRETURN NO
*
DMREST BAL  RBAL,READ1 READ RECORD FROM INPUT1
BAL  RBAL,READ2 READ RECORD FROM INPUT2
*
DMCOMP LH  R2,KEYLENM1 ARE KEYS SAME?
*
    L  R1,IN1LOC LOAD LOCATION OF INPUT1 RECORD
    A  R1,IN1RKP ADD OFFSET TO KEY
    L  R15,IN2LOC LOAD LOCATION OF INPUT2 RECORD
    A  R15,IN2RKP ADD OFFSET TO KEY
*
    EX  R2,DMCLC ARE KEYS SAME?
    BE  DMSAME YES
    BH  DM2LT KEY1<KEY2
*
    TM  OPTIONS,UNIQUETBIT UNIQUE OPTION?
    BZ  DMREAD1 NO
* BAL RBAL,PUTKEY GO PUT KEY IMAGE IN PRINT LINE ARRAY
  AP COUNTUNQ,=P'1' COUNT UNIQUE KEYS
* DMREAD1 BAL RBAL,READ1 READ INPUT1
  B DMCOMP GO CHECK FOR MATCH
* DM2LT BAL RBAL,READ2 READ INPUT2
  B DMCOMP GO CHECK FOR MATCH
* DMSAME AP COUNTDUP,=P'1' COUNT DUPLICATE KEYS
* TM OPTIONS,MATCHBIT MATCH OPTION?
  BZ DMREST NO
* CLC IN1RECL,IN2RECL ARE RECORDS SAME SIZE?
  BNE DMDIFF NO
* L RØ,IN1LOC LOAD LOCATION OF INPUT1 RECORD
  L R1,IN1RECL LOAD SIZE OF INPUT1 RECORD
  L R2,IN2LOC LOAD LOCATION OF INPUT2 RECORD
  L R3,IN2RECL LOAD SIZE OF INPUT2 RECORD
* CLCL RØ,R2 ARE RECORDS IDENTICAL?
  BNE DMDIFF NO
* AP IDENTS,=P'1' COUNT IDENTICAL RECORDS
  MVI IDENT,C'*' SET IDENTICAL FLAG
* DMDIFF BAL RBAL,PUTKEY GO PUT KEY IMAGE IN PRINT LINE ARRAY
* B DMREST GO GET ANOTHER PAIR OF RECORDS
* DMRETURN L RBAL,SAVDMBAL RESTORE LINKAGE REGISTER
  BR RBAL RETURN
* DMCLC CLC Ø(*-*,R1),Ø(R15)
* EJECT
*********************************************************************
***                                                               ***
***   READ INPUT1, SEARCH FOR 'KEY OF RECORD' IDENTIFIER           ***
***                                                               ***
*********************************************************************
* READ1 ST RBAL,SAVRIBAL SAVE LINKAGE REGISTER
* R1LOOP LA R2,IN1RPL POINT TO RPL

BAL RBAL,READVSAM GO READ RECORD FROM INPUT1
*
AP COUNT1,=P'1' COUNT RECORD
AP COUNT1F,COUNTFRM COUNT POSSIBLE 'FROM' EXCLUSION
AP COUNT1T,COUNTTHR COUNT POSSIBLE 'THRU' EXCLUSION
AP COUNT1E,COUNTXCL COUNT POSSIBLE 'EXCL' EXCLUSION
AP COUNT1M,COUNTMAX COUNT POSSIBLE 'MXCL' EXCLUSION
*
R1RETURN L RBAL,SAVR1BAL RESTORE LINKAGE REGISTER
BR RBAL RETURN
*
I1EOF AP COUNT1F,COUNTFRM COUNT POSSIBLE 'FROM' EXCLUSION
AP COUNT1T,COUNTTHR COUNT POSSIBLE 'THRU' EXCLUSION
AP COUNT1E,COUNTXCL COUNT POSSIBLE 'EXCL' EXCLUSION
AP COUNT1M,COUNTMAX COUNT POSSIBLE 'MVCL' EXCLUSION
*
TM OPTIONS,LISTBIT LIST OPTION?
BO DLRETURN YES, GO EXIT DOLIST
*
TM EOFFLAGS,2 E-O-F ON INPUT2?
BO DMRETURN YES, GO EXIT DOMATCH
*
OI EOFFLAGS,1 SET E-O-F ON INPUT1
*
I1EOF L BAL RBAL,READ2 FLUSH INPUT2 FOR COUNT
B I1EOF CONTINUE
*
EJECT
*********************************************************************
***                                                               ***
***   READ INPUT2, SEARCH FOR 'KEY OF RECORD' IDENTIFIER          ***
***                                                               ***
*********************************************************************
*
READ2 ST RBAL,SAVR2BAL SAVE LINKAGE REGISTER
*
R2LOOP LA R2,IN2RPL POINT TO RPL
*
BAL RBAL,READVSAM READ RECORD FROM INPUT2
*
AP COUNT2,=P'1' COUNT RECORD
AP COUNT2F,COUNTFRM COUNT POSSIBLE 'FROM' EXCLUSION
AP COUNT2T,COUNTTHR COUNT POSSIBLE 'THRU' EXCLUSION
AP COUNT2E,COUNTXCL COUNT POSSIBLE 'EXCL' EXCLUSION
AP COUNT2M,COUNTMAX COUNT POSSIBLE 'MXCL' EXCLUSION
*
R2RETURN L RBAL,SAVR2BAL RESTORE LINKAGE REGISTER
BR RBAL RETURN
*
I2EOF AP COUNT2F,COUNTFRM COUNT POSSIBLE 'FROM' EXCLUSION
AP COUNT2T,COUNTTHR COUNT POSSIBLE 'THRU' EXCLUSION
AP    COUNT2E,COUNTXCL    COUNT POSSIBLE 'EXCL' EXCLUSION
AP    COUNT2M,COUNTMAX    COUNT POSSIBLE 'MAXL' EXCLUSION

TM    EOFFLAGS,1          E-O-F ON INPUT1?
BO    DMRETURN             YES, GO EXIT DOMATCH

OI    EOFFLAGS,2          SET E-O-F ON INPUT1

I2EOFL BAL    RBAL,READ1    FLUSH INPUT1 FOR COUNT
B    I2EOFL    CONTINUE

******************************************************************************
***                                                               ***
***   SET 'LINES' ARRAY TO BLANKS                                     ***
***                                                               ***
******************************************************************************

CLRPAGE ST    RBAL,SAVCPBAL       SAVE LINKAGE REGISTER

LA    R15,LINES           POINT TO FIRST LINE
LA    RØ,L'LINES          LENGTH OF LINE
L    R1,=A(LPP*L'LINES-L'LINES) (LINE LENGTH) * (LINES - 1)
AR    R1,R15              POINT TO LAST LINE
MVI    LINES,C' '         SET SEED
MVC    CZN,=2C' ZN'       SET FOR CHARACTER, ZONE, NUMBER

LH    R2,KEYLENMN         GET KEY LENGTH
CH    R2,=AL2(L'LINES-2)  WILL KEY FIT ON ONE LINE?
BL    CPLLOOP             YES

MVC    CZN,=C' Z N'      SET TO DOUBLE SPACE INDICATORS

CPLLOOP MVC    1(L'LINES,R15),Ø(R15) CLEAR LINE TO BLANKS

TM    OPTIONS,HEXBIT      HEX OPTION?
BZ    CPLNOTHX             NO

MVC    Ø(1,R15),CZN       SET ' ', 'Z', OR 'N'
MVC    CZN(L'CZN-1),CZN+1  POSITION 2-N TO 1-(N-1)
MVC    CZN+L'CZN-1(1),Ø(R15) POSITION 1 TO N

CPLNOTHX BXLE  R15,RØ,CPLLOOP       CONTINUE

LA    R1,LINES           POINT TO FIRST LINE
ST    R1,LINENPTR        SAVE
XR    R1,R1               SET TO PRINT COLUMN 2

TM    OPTIONS,HEXBIT      IS LISTING IN VERTICAL HEX?
BZ    CPNOTHEX             NO
LA R1,1 SET TO PRINT COLUMN 3
*
CPNOTHEX STH R1,COLPTR CLEAR COLUMN DISPLACEMENT
*
L RBAL,SAVCPBAL RESTORE LINKAGE REGISTER
BR RBAL RETURN
*
EJECT
*************************************************************************
***
*** PRINT 'LINES'
***
*************************************************************************
*
PRTPAGE ST RBAL,SAVPPBAL SAVE LINKAGE REGISTER
*
LA R3,LINES POINT TO FIRST LINE
LA R4,L'LINES LENGTH OF LINE
L R5,=A(LPP*L'LINES-L'LINES) (LINE LENGTH) * (LINES - 1)
AR R5,R3 POINT TO LAST LINE
*
PPLOOP CLC LINE+1(L'LINES),Ø(R3) IS IMAGE BLANK?
BE PPFINISH YES
*
MVC LINE+1(L'LINES),Ø(R3) MOVE IMAGE TO PRINT LINE
BAL RBAL,PRINT PRINT LINE
BXLE R3,R4,PPLOOP CONTINUE
*
PPFINISH BAL RBAL,CLRPAGE CLEAR 'LINES' TO BLANKS
*
L RBAL,SAVPPBAL RESTORE LINKAGE REGISTER
BR RBAL RETURN
*
EJECT
*************************************************************************
***
*** PRINT TOTALS
***
*************************************************************************
*
DOTOTALS ST RBAL,SAVDTBAL SAVE LINKAGE REGISTER
*
BAL RBAL,Doublesp ALLOW FOR DOUBLE SPACE
MVC LINE(22),=C'NON-EXCLUDED RECORDS:' SET IDENTIFIER
BAL RBAL,PRINT GO PRINT IDENTIFIER
*
LA R2,INPUT1 POINT TO INPUT1 ACB
BAL RBAL,GETNAME GO GET DSN, FORMAT TOTALS, ETC.
*
TM OPTIONS,LISTBIT WAS INPUT2 READ?
BO DTNOT2 NO
* LA R2,INPUT2           POINT TO INPUT2 ACB
BAL RBAL,GETNAME        GO GET DSN, FORMAT TOTALS, ETC.
* MVC LINE+1(14),=C'DUPLICATE KEYS'
MVC LINE+16(6),=X'20206B202120' SET EDIT PATTERN
ED LINE+15(7),COUNTDUP FORMAT RECORD COUNT
BAL RBAL,PRINT         PRINT TOTAL DUPLICATE KEYS
* MVC LINE+1(14),=C'IDENTICAL RECS'
* TM OPTIONS,MATCHBIT   OPTION=MATCH?
BO DTMATCH              YES
* MVC LINE+1(14),=C'UNIQUE(INPUT1)'
ZAP IDENTS,COUNTUNQ    GET COUNT FROM INPUT
* DTMATCH MVC LINE+16(6),=X'20206B202120' SET EDIT PATTERN
ED LINE+15(7),IDENTS   FORMAT RECORD COUNT
BAL RBAL,PRINT         PRINT TOTAL DUPLICATE KEYS
* DTNOT2 BAL RBAL,DOUBLES     ALLOW FOR DOUBLE SPACE
MVC LINE(25),=C'ØINPUT1 EXCLUDED RECORDS:' SET IDENTIFIER
BAL RBAL,PRINT         GO PRINT IDENTIFIER
* MVC LINE+6(9),=C'BY ''FROM''
MVC LINE+16(6),=X'20206B202120' SET EDIT PATTERN
ED LINE+15(7),COUNT1F FORMAT RECORD COUNT
BAL RBAL,PRINT         PRINT TOTAL DUPLICATE KEYS
* MVC LINE+6(9),=C'BY ''THRU''
MVC LINE+16(6),=X'20206B202120' SET EDIT PATTERN
ED LINE+15(7),COUNT1T FORMAT RECORD COUNT
BAL RBAL,PRINT         PRINT TOTAL DUPLICATE KEYS
* MVC LINE+6(9),=C'BY ''EXCL''
MVC LINE+16(6),=X'20206B202120' SET EDIT PATTERN
ED LINE+15(7),COUNT1E FORMAT RECORD COUNT
BAL RBAL,PRINT         PRINT TOTAL DUPLICATE KEYS
* MVC LINE+6(9),=C'BY ''MAXL''
MVC LINE+16(6),=X'20206B202120' SET EDIT PATTERN
ED LINE+15(7),COUNT1M FORMAT RECORD COUNT
BAL RBAL,PRINT         PRINT TOTAL DUPLICATE KEYS
* TM OPTIONS,LISTBIT    WAS INPUT2 READ?
BO DTCLOSE              NO
* BAL RBAL,DOUBLES       ALLOW FOR DOUBLE SPACE
MVC LINE(25),=C'ØINPUT2 EXCLUDED RECORDS:' SET IDENTIFIER
BAL RBAL,PRINT         GO PRINT IDENTIFIER

MVC LINE+6(9),=C'BY ''FROM''
MVC LINE+16(6),=X'20206B202120' SET EDIT PATTERN
ED LINE+15(7),COUNT2F FORMAT RECORD COUNT
BAL RBAL,PRINT PRINT TOTAL DUPLICATE KEYS

MVC LINE+6(9),=C'BY ''THRU''
MVC LINE+16(6),=X'20206B202120' SET EDIT PATTERN
ED LINE+15(7),COUNT2T FORMAT RECORD COUNT
BAL RBAL,PRINT PRINT TOTAL DUPLICATE KEYS

MVC LINE+6(9),=C'BY ''EXCL''
MVC LINE+16(6),=X'20206B202120' SET EDIT PATTERN
ED LINE+15(7),COUNT2E FORMAT RECORD COUNT
BAL RBAL,PRINT PRINT TOTAL DUPLICATE KEYS

MVC LINE+6(9),=C'BY ''MAXL''
MVC LINE+16(6),=X'20206B202120' SET EDIT PATTERN
ED LINE+15(7),COUNT2M FORMAT RECORD COUNT
BAL RBAL,PRINT PRINT TOTAL DUPLICATE KEYS

DTCLOSE MVC RGCLOSE(RGCLOSEL),CLOSED SET RANGES CLOSE LIST
CLOSE (RANGES),MF=(E,RGCLOSEL) CLOSE RANGES TO REPROCESS

MVC RGOPENL(RGOPENLN),OPEND SET RANGES OPEN LIST
OPEN (RANGES,(INPUT)),MF=(E,RGOPENL) REOPEN RANGES

OI EOFFLAGS,X'40'
MVC LINE(14),=C'@COMMAND LIST:' IDENTIFY COMMAND LIST
BAL RBAL,DOUBLESP ALLOW FOR DOUBLE SPACE
BAL RBAL,PRINT PRINT IDENTIFIER

DTLOOP GET RANGES,RINAREA READ CONTROL STATEMENT
MVC LINE+1(L'LINE-1),RINAREA MOVE TO PRINT LINE
BAL RBAL,PRINT PRINT CONTROL STATEMENT
B DTLOOP CONTINUE

DTRETURN L RBAL,SAVDTBAL RESTORE LINKAGE REGISTER
BR RBAL RETURN

*********************************************************************
*** READ VSAM RECORD ***
*********************************************************************

READVSAM ST RBAL,SAVRVBAL SAVE LINKAGE REGISTER

PRINT GEN
ZAP COUNTFRM,=P'Ø' INITIALIZE 'FROMKEY' EXCLUSION COUNT
ZAP COUNTTHR,=P'Ø' INITIALIZE 'FROMKEY' EXCLUSION COUNT
ZAP COUNTXCL,=P'Ø' INITIALIZE 'EXCLKEY' EXCLUSION COUNT
ZAP COUNTMAX,=P'Ø' INITIALIZE 'EXCLKEY' EXCLUSION COUNT

* RVNEXT GET RPL=(R2) READ RECORD
* LTR R15,R15 READ OKAY?
BNZ VSAMGERR NO
* LA RBAL,SHOWCB1-IN1RPL(R2) POINT TO SHOWCB TO GET RECLN
* SHOWCB RPL=(R2),MF=(E,(RBAL)) GET RECORD LENGTH-->IN_RECL
* RVEXCL L R1,IN1LOC-IN1RPL(R2) GET LOCATION OF RECORD
A R1,IN1RKP-IN1RPL(R2) GET LOCATION OF KEY
* TM OPTIONS,MAXKBIT WAS MAXIMUM KEY LENGTH SPECIFIED?
BZ RVNOMAX NO
* LH R14,KEYLENM1 GET SPECIFIED KEY LENGTH
LA R15,OLDKEY1-IN1RPL(R2) GET ADDRESS OF PREVIOUS KEY
EX R14,RANGECHK MATCH OF PREVIOUS KEY?
BNE RVNEWKEY NO
* AP COUNTMAX,=P'1' COUNT DUPLICATE KEYS
B RVNEXT GO READ ANOTHER RECORD
* RVNEWKEY EX R14,RVMVCKEY MOVE KEY TO OLDKEY
* RVNOMAX TM OPTIONS,EXCLBIT 'EXCLUDE KEY' SPECIFIED?
BZ RVNOTE NO
* LH R14,EXCLLEN GET LENGTH OF 'EXCL'/ 'FIND' STRING
LA R15,EXCLKKEY GET LOCATION OF 'EXCL'/'FIND' STRING
EX R14,RANGECHK HAS 'EXCL'/'FIND' KEY BEEN REACHED?
BH RVNOTE NO, KEY NOT YET FOUND
BL RVFIND YES, PAST EXCLUSION
* CLC =C'FIND',RINAREA WAS THIS A FIND COMMAND?
BE RVFIND YES
* AP COUNTXCL,=P'1' COUNT EXCLUSION
B RVNEXT GO GET NEXT RECORD
* RVFIND TM EOFFLAGS,X'8Ø' END OF RANGE FILE REACHED?
BO RVNOTE YES
* LR R7,R2        SAVE POINTER TO VSAM RPL
* BAL RBAL,GETRANGE      GO SEE IF OTHER CONTROL STATEMENTS
* LR R2,R7       RESTORE POINTER TO VSAM RPL
  B RVEXCL      GO RE-EXAMINE RECORD
  RVNOTE TM OPTIONS,FROMBIT 'FROMKEY' SPECIFIED?
  BZ RVNOTF      NO
    LH R14,FROMLEN GET LENGTH OF 'FROM' STRING
    LA R15,FROMKEY GET LOCATION OF 'FROM' STRING
    EX R14,RANGECHK HAS 'FROM' KEY BEEN REACHED?
    BNH RVNOTF      YES
    AP COUNTFRM,=P'1' COUNT 'FROM' EXCLUSION
    B RVNEXT       GO BYPASS EXCLUSION
* RVNOTF TM OPTIONS,THURBIT 'THRUKEY' SPECIFIED?
  BZ RVRETURN      NO
    LH R14,THRULEN GET LENGTH OF 'THRU' STRING
    LA R15,THRUKEY GET LOCATION OF 'THRU' STRING
    EX R14,RANGECHK HAS 'THRU' KEY BEEN PASSED?
    BNL RVRETURN      NO
    AP COUNTTHR,=P'1' COUNT 'THRU' EXCLUSION
    B RVNEXT       GO BYPASS EXCLUSION
* RVRETURN L RBAL,SAVRVBAL RESTORE LINKAGE REGISTER
  BR RBAL        RETURN
  RANGECHK CLC Ø(*-*.,R15),Ø(R1)
  RVMVCKEY MVC Ø(*-*.,R15),Ø(R1)
  * EJECT

*********************************************************************
*** OPEN VSAMFILE ***
*********************************************************************
* OPENVSAM ST RBAL,SAVOVBAL    SAVE LINKAGE REGISTER
  * LA RBAL,INPUT1-IN1RPL(R2) POINT TO ACB
  * OPEN ((RBAL)) OPEN VSAM FILE
  * LTR R15,R15    WAS OPEN SUCCESSFUL?
  BNZ VSAMOERR      NO
  * LA R3,IN1KEYL-IN1RPL(R2) POINT TO KEYLEN AREA
  * SHOWCB ACB=(RBAL),OBJECT=DATA,FIELDS=(KEYLEN,RKP,LRECL). -
AREA=(R3),LENGTH=12,MF=(G,SHOWCB3,LSHOWCB3)

* SHOWCB ACB=(RBAL),MF=(E,SHOWCB3) KEYLEN,RKP-->
  IN_KEYL,,IN_RKP
  LRECL--->IN_MAXRL
L R5,IN1MAXRL-IN1RPL(R2) LOAD MAX RECORD SIZE
GETMAIN R,lv=(R5) GET WORK AREA
ST R1,IN1LOC-IN1RPL(R2) SAVE ADDRESS OF RECORD WORK AREA

* LR R3,R1 POINT TO LOCATION ADDRESS
LA R4,IN1RPLX-IN1RPL(R2) POINT TO PARAMETER LIST

* GENCB BLK=RPL,ACB=(RBAL),AM=VSAM,AREA=(R3),AREALEN=(R5),
  OPTCD=(KEY,SEQ,FWD,NUP,MVE),MF=(G,(R4),LRPL),
  WAREA=(R2),LENGTH=LIN1RPL

* LA R3,INIRECL-IN1RPL(R2) POINT TO IN_RECL
LA RBAL,SHOWCB1-IN1RPL(R2) POINT TO SHOWCB_

* SHOWCB RPL=(R2),AREA=(R3),LENGTH=4,FIELDS=(RECL),
  MF=(G,(RBAL),LSHOWCB1) GEN SHOWCB FOR RECLEN-->
  IN_RECL

* L RBAL,SAVOVBAL RESTORE LINKAGE REGISTER
BR RBAL RETURN

EJECT

*********************************************************************
***  CLOSE VSAM FILE  ***
*********************************************************************

* CLOSVSAM ST RBAL,SAVCVBAL SAVE LINKAGE REGISTER
* AGO .NOCLOSE
  CLOSE ((R2)) OPEN VSAM FILE
  LTR R15,R15 WAS OPEN SUCCESSFUL?
  BNZ VSAMCERR NO
* .NOCLOSE ANOP
  L RBAL,SAVCVBAL RESTORE LINKAGE REGISTER
  BR RBAL RETURN

EJECT

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

Keith Nicaise (USA)  © Xephon 1998
Testing to see whether a VSAM cluster is empty

The program presented here was developed and tested under VSE/ESA Version 1.3. It is now running under VSE/ESA Version 2.2.

The program is called by the following job control statement:

    // EXEC TSTEMPTY,PARM='filename'

and checks whether a VSAM (ESDS or KSDS) cluster is empty.

The PARM string of the EXEC statement must be the filename (ddname) of the cluster that you want to check. Under VSE/ESA, this string can be up to seven bytes long.

One of the following return codes is passed to job control:

0   VSAM cluster is not empty (OPEN was successful).
4   VSAM cluster is empty.
9   PARM string missing or too long.
10  SHOWCB error (should not occur).
11  Other OPEN error (usually file not found); see console.

At our site, we use the program to skip steps of a batch job if processing those steps results in an empty output file. This is illustrated by the following example:

    // JOB EXAMPLE
    // ON $RC >= 8 GOTO ABEND
    ...
    // DLBL SORTIN1,'CLIENTS',,VSAM,CAT=...
    // DLBL SORTOUT,'SPECIAL.CLIENTS',,VSAM,CAT=...
    // EXEC SORT,SIZE=200K
    SORT FIELDS=...
    RECORD TYPE=F,LENGTH=...
    INCLUDE COND=...  // SELECTS SPECIAL CLIENTS
    INPFILE VSAM
    OUTFILE ESDS,REUSE
    /*
    // EXEC TSTEMPTY,PARM='SORTOUT'
    // IF $RC EQ 4 THEN
    // GOTO EMPTY
    */
TSTEMPTY

TITLE 'TSTEMPTY - TEST IF VSAM CLUSTER IS EMPTY'
TSTEMPTY CSECT
*********************************************************************
*        REGISTER EQUATES
*********************************************************************
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
    EJECT
*********************************************************************
* REGISTER USAGE:
*    R15 PROGRAM ENTRY POINT, RETURN CODE
*    R14 RETURN ADDRESS
*    R13 SAVE AREA ADDRESS
*    R12
*    R11
*    R1Ø
*    R9 BASE REGISTER
*    R8
*    R7
*    R6
*    R5
*    R4
*    R3 WORK REGISTER
*    R2 LENGTH OF FILENAME (INPUT PARAMETER)
*    R1 ADDRESS OF FILENAME (INPUT PARAMETER), USED BY IBM MACROS
*    RØ WORK REGISTER, USED BY IBM MACROS
*********************************************************************
EJECT
*********************************************************************
* TEST INPUT PARAMETER AND MOVE IT TO WORKING STORAGE
*********************************************************************
BALR R9,Ø                LOAD BASE REGISTER
USING *,R9                ESTABLISH ADDRESSABILITY
LA    R13,SAVEAREA        ADDRESS OF SAVE AREA
CR    R1,R15              PARM STRING EXISTS
BE    PARMERR              NO, INFORM JOB CONTROL
TM    Ø(R1),X'80'          HIGH ORDER BIT OK
BNO   PARMERR              NO, INFORM JOB CONTROL
L     R1,Ø(R1)             ADDRESS OF PARAMETER
LH    R2,Ø(R1)             LENGTH OF PARAMETER
LTR   R2,R2                LENGTH OF PARAMETER POSITIVE
BNP   PARMERR              NO, INFORM JOB CONTROL
LA    R3,L'FILENM-1       LOAD GREATEST ALLOWED LENGTH
CR    R2,R3                PARAMETER TOO LONG
BH    PARMERR              YES, INFORM JOB CONTROL
MVI   FILENM,C' '          INITIALIZE STORAGE
MVC   FILENM+1(L'FILENM-1),FILENM
BCTR  R2,Ø                 LENGTH FOR EXECUTE
EX    R2,MVCPARM           MOVE PARAMETER TO STORAGE
EJECT
*********************************************************************
* STORE FILENAME IN VSAM ACCESS CONTROL BLOCK
*********************************************************************
MODCB AM=VSAM,              *
           ACB=VSAMFIL,          *
           DDNAME=(*,FILENM)
EJECT
*********************************************************************
* OPEN FILE
*********************************************************************
OPEN  VSAMFIL             OPEN FILE
LTR   R15,R15             TEST RETURN CODE
BNZ   OPENERR             ERROR
EJECT
*********************************************************************
* CLOSE SUCCESSFULLY OPENED FILE
*********************************************************************
CLOSE VSAMFIL             CLOSE FILE
SR    R15,R15             FILE NOT EMPTY, RETURN CODE Ø
EJECT
*********************************************************************
* TERMINATE PROGRAM WITH RETURN CODE IN REGISTER 15
*********************************************************************
RETURN   EOJ   RC=(R15)
EJECT
* TEST RETURN CODE FROM OPEN
*********************************************************************
OPENERR DS ØH
   SHOWCB ACB=VSAMFIL, AM=VSAM, AREA=OPENRC, FIELDS=ERROR, LENGTH=4
   LTR R15,R15 TEST RETURN CODE FROM SHOWCB
   BNE SHOWERR MACRO SHOWCB WITH ERROR
   CLI OPENRC+L'OPENRC-1,X'6E' TEST, IF FILE WAS EMPTY
   BNE OPENOTH NO, OTHER ERROR
   LA R15,4 SET RETURN CODE TO 4
   B RETURN INFORM JOB CONTROL
   MVC Parm MVC FILENM(Ø),2(R1) MOVE INPUT PARAMETER TO STORAGE
   EJECT
* SEVERE ERRORS, SET RETURN CODE
*********************************************************************
PARMERR DS ØH MORE THAN ONE PARAMETER
   LA R15,9 SET RETURN CODE TO 9
   B RETURN INFORM JOB CONTROL
SHOWERR DS ØH SHOWCB IN ERROR
   LA R15,10 SET RETURN CODE TO 10
   B RETURN INFORM JOB CONTROL
OPENOTH DS ØH OPEN ERROR
   LA R15,11 SET RETURN CODE TO 11
   B RETURN INFORM JOB CONTROL
   EJECT
*********************************************************************
* VSAM ACCESS CONTROL BLOCK
*********************************************************************
VSAMFIL ACB AM=VSAM, MACRF=(ADR,SEQ,NRS,IN)
   EJECT
*********************************************************************
* WORKING STORAGE
*********************************************************************
SAVEAREA DS 9D OWN SAVE AREA
OPENRC DS F ERROR CODE FROM SHOWCB
FILENM DS CL8 FILENAME (INPUT PARAMETER)
END TSTEMPTY

Walter Richters
(Germany) © Xephon 1998
Resetting a VSAM cluster

The program presented in this article was developed and tested under VSE/ESA Version 1.3. It is now running under VSE/ESA Version 2.2.

The program is called by the following job control statement:

```
// EXEC SETEMPTY,PARM='filename'
```

and resets (empties) a VSAM cluster. The cluster must be an ESDS or a KSDS defined with the REUSE attribute.

The PARM string of the EXEC statement must be the filename (ddname) of the cluster that you want to reset. Under VSE/ESA, this string can be up to seven bytes long.

One of the following return codes is passed to job control:

0  Reset was successful.
9  PARM string missing or too long.
11 OPEN error; see console.

You can of course use other methods, such as job control, to reset a reusable cluster when it is opened or closed. However, the first or last processing step using the file is not always the right moment to reset the cluster. In these cases, as the following example shows, it helps to use the program presented here.

EXAMPLE

```
// JOB EXAMPLE
// ON $RC >= 8 GOTO ABEND
...
// DLBL TESTFIL,'SPECIAL.CLIENTS',,VSAM,CAT=...
// EXEC SETEMPTY,PARM='TESTFIL'
...
```
SETEMPTY

TITLE 'SETEMPTY - RESET VSAM CLUSTER'

SETEMPTY CSECT

**************************************************************
* REGISTER EQUATES
**************************************************************
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15

EJECT

**************************************************************
* REGISTER USAGE:
*    R15 PROGRAM ENTRY POINT, RETURN CODE
*    R14 RETURN ADDRESS
*    R13 SAVE AREA ADDRESS
*    R12
*    R11
*    R1Ø
*    R9 BASE REGISTER
*    R8
*    R7
*    R6
*    R5
*    R4
*    R3 WORK REGISTER
*    R2 LENGTH OF FILENAME (INPUT PARAMETER)
*    R1 ADDRESS OF FILENAME (INPUT PARAMETER), USED BY IBM MACROS
*    RØ WORK REGISTER, USED BY IBM MACROS

EJECT

**************************************************************
* TEST INPUT PARAMETER AND MOVE IT TO WORKING STORAGE
**************************************************************
BALR R9,Ø
USING *,R9
LA R13,SAVEAREA
CR R1,R15
BE PARMERR
TM Ø(R1),X'8Ø'
BNO PARMERR
L R1,Ø(R1)
LH R2,Ø(R1)
LTR R2,R2
BNO PARMERR
LA R3,L'FILENM-1
CR R2,R3
BH PARMERR
MVI FILENM,C' ' 
MVC FILENM+1(L'FILENM-1),FILENM
BCTR R2,Ø
EX R2,MVCPARM
EJECT
*********************************************************************
*                  STORE FILENAME IN VSAM ACCESS CONTROL BLOCK
*********************************************************************
MODCB AM=VSAM,          *
               ACB=VSAMFIL,    *
               DDNAME=(*,FILENM)
EJECT
*********************************************************************
*                  OPEN AND RESET FILE
*********************************************************************
OPEN VSAMFIL
LTR R15,R15
BNZ OPENERR
EJECT
*********************************************************************
*                  CLOSE SUCCESSFULLY OPENED FILE
*********************************************************************
CLOSE VSAMFIL
SR R15,R15
EJECT
*********************************************************************
*                  TERMINATE PROGRAM WITH RETURN CODE IN REGISTER 15
*********************************************************************
RETURN EOJ RC=(R15)
EJECT
*********************************************************************
*                  SEVERE ERRORS, SET RETURN CODE
*********************************************************************
PARMERR DS ØH MORE THAN ONE PARAMETER
LA R15,9 SET RETURN CODE TO 9
B RETURN INFORM JOB CONTROL
OPENERR DS ØH OPEN ERROR
LA R15,11 SET RETURN CODE TO 11
B RETURN INFORM JOB CONTROL
MVCPARM MVC FILENM(Ø),2(R1) MOVE INPUT PARAMETER TO STORAGE

*********************************************************************
*        VSAM ACCESS CONTROL BLOCK
*********************************************************************
VSAMFIL ACB AM=VSAM,
         MACRF=(ADR,SEQ,RST,OUT)
EJECT

*********************************************************************
*        WORKING STORAGE
*********************************************************************
SAVEAREA DS 9D OWN SAVE AREA
FILENM DS CL8 FILENAME (INPUT PARAMETER)
END SETEMPTY

Walter Richters
(Germany)
© Xephon 1998

Approximately 3,500 files containing code from Xephon’s technical journals can be viewed and downloaded from our Web site, free of charge. All code published before the end of 1996 is included. (Articles from January 1997 onwards are still controlled by password.)

There are three means of access:

- A chronological listing by issue date.
- An alphabetical listing by article title.
- A keyword free-text search facility (only article titles are indexed).

Our Web site is at http://www.xephon.com
INTRODUCTION

This article describes a simple and automatic method for setting the RECORDSIZE and KEYLENGTH parameters in the definitions of VSAM files held in the DFHCSD, without using an IDCAMS LISTCAT or other tool (eg FILE-AID). The RECORDSIZE and KEYLENGTH parameters specified in the DFHCSD file are checked to ensure that they correspond to the actual values of the files.

THE PROBLEM

Many installations have a TOR-AOR-DOR structure. The management of file definitions in the DFHCSD requires a single definition for every file used by both the DOR and the AOR, and the CICS region-id has to be specified in the DOR’s REMOTESYSTEM parameter. The file definitions also need the RECORDSIZE and KEYLENGTH parameters to be specified. This means that an IDCAMS LISTCAT (or third-party) utility must be executed for each file (about a thousand in our case), which is both time-consuming and error-prone.

THE SOLUTION

To automate and simplify the process, we perform the following steps:

1. Execute the DFHCSDUP batch utility, specifying in the LIST command the group or list containing the files to be processed (more than one LIST command can be specified). The output file (SYSPRINT) of the DFHCSDUP utility is assigned to a sequential file with the characteristics of RECFM: VBA, an LRECL of 125, and an appropriate BLKSIZE, eg 13200.

2. Execute the user batch program CSDVER, which analyses the output listing written by the DFHCSDUP utility, and also:
– Gets the DSNAME (which must be present) and the GROUPNAME for each file, together with the RECORDSIZE and KEYLENGTH parameters (if specified).

– Dynamically allocates the file by using the DSNAME obtained from the listing (this uses the user batch program DYNALLOC).

– Opens the file (for input).

– Gets the RECORDSIZE and KEYLENGTH parameters from the ACB.

– Compares these parameters with those obtained from the output listing written by the DFHCSDUP utility, and, if they differ, writes the following commands:

```
ALTER FILE(xxxxxxxx) GROUP(xxxxxxxx) RECORDSIZE(xxxxx)
```

and/or

```
ALTER FILE(xxxxxxxx) GROUP(xxxxxxxx) KEYLENGTH(xxxxx)
```

into a sequential fixed block output file (use LRECL=80 and an appropriate BLKSIZE, eg 8000).

– Closes and deallocates the file.

3 After verifying (if desired) the sequential file written by the CSDVER program, execute the DFHCSDUP batch utility, assigning SYSIN to the sequential file containing the ALTER commands. We chose to analyse the output listing produced by the DFHCSDUP utility rather than have direct access to the DFHCSD, so that we would be unaffected by any modifications to the DFHCSD in future releases of CICS. The DYNALLOC program, written to dynamically allocate files, can be used by any other batch program.

The source for the user batch program CSDVER is given below. The source code for DYNALLOC and sample JCL will be published in the next issue.
SOURCE CODE PROGRAM CSDVER

TITLE 'CSDVER - CHECKS THE CSD KEYLENGTH && RECORDSIZE'
* PROGRAM: CSDVER
* FOR CICS ESA 3.3.0:
* VERIFY IF IN YOUR ENVIRONMENT HAS BEEN APPLIED THE FOLLOWING PTF
* FOR DFHCSDUP UTILITY:
*                      PTF UN64969
* MACRO DEFINITION
MACRO                   MACRO HEADER
CSSET                   PROTOTYPE STATEMENT
PGMNAME                DC    CL8'&SYSECT'       PROGRAM NAME
MEND                    MACRO END
CSDVER      CSECT
* BATCH PROGRAM
* THE PURPOSE OF THIS PROGRAM IS TO:
* 1) CHECK LRECL AND KEYLENGTH SPECIFIED IN DFHCSD
*    CORRESPOND TO THE ACTUAL CHARACTERISTICS OF THE FILE.
* 2) IF A MISMATCH IS FOUND, WRITE A CONTROL CARD FOR THE
*    'DFHCSDUP' UTILITY PROGRAM (ALTER..) TO CORRECT LRECL AND/OR
*    KEYLENGTH.
* 3) TO DO SO, THE PROGRAM DYNAMICALLY ALLOCATES THE FILE,
*    THEN OPENS AND CLOSES IT (TO DEALLOCATE IT).
* 4) THE PROGRAM USES, AS INPUT, THE LISTING OF THE FILES OBTAINED WITH
*    THE 'LIST' COMMAND OF THE UTILITY 'DFHCSDUP' PROGRAM
*    (DDNAME=CSDPRT).
* 5) IT CREATES A 'SYSIN' FILE (DDNAME=CSDVARY) FOR THE 'DFHCSDUP'
*    UTILITY IN ORDER TO CHANGE THE CSD.
*    -THE PRINT FILE (DDNAME=TRACE) CONTAINS A TRACE OF THE ACTIVITY OF
*      THE PROGRAM ITSELF.
*    -THE STANDARD ASSIGNMENT OF THIS PRINT FILE IS 'DUMMY'.
*    -THE PRINT FILE (DDNAME=PRINT) SHOWS THE CHARACTERISTICS OF THE
*      FILES THAT HAVE BEEN EXAMINED AND THE MISMATCHES.
*    -THE FILE (DDNAME=CSDVARY) CONTAINS THE SYSIN FOR THE UTILITY
*      PROGRAM 'DFHCSDUP'.
* EXAMPLE OF A 'LIST' COMMAND FOR THE 'DFHCSD' UTILITY (ANY SYNTAX
* ACCEPTED BY DFHCSDUP UTILITY WORKS):
* LIST GROUP(*FP+) OBJECTS
* OR
* LIST GROUP(GROUPØ1) OBJECTS
* OR
* LIST LIST(LISTØ1) OBJECTS
* ATTENTION: IF THE DSNAME HAS NOT BEEN SPECIFIED IN THE DFHCSD
* (AS IN REMOTE FILES), NO CONTROL CAN BE CARRIED OUT AND THE
* ENTRY IS BYPASSED
* REGISTER USAGE
RBAL1    EQU   R1 LEVEL 1 BAL
RBAL2    EQU   R2 LEVEL 2 BAL
**Program Identifier**

```
B 28(Ø,R15) BRANCH AROUND CONSTANTS
DC CL8'CSDVER' PROGRAM NAME
DC CL8'&SYSDATE' TODAY'S DATE
DC CL8'&SYSTIME' TIME OF COMPIL
```

**Addressability & Save Area Chaining**

```
STM R14,R12,12(R13)
LR RBASE1,R15
USING CSDVER,RBASE1,RBASE2,RBASE3 TELL ASM
LA RBASE2,2048(RBASE1)
LA RBASE2,2048(RBASE2)
LA RBASE3,2048(RBASE2)
LA RBASE3,2048(RBASE3)
LA RØ,SAVEAREA ADDRESS OF SAVEAREA
ST R13,SAVEAREA+4 INVOKER'S SAVE AREA ADDR IN
* MY SAVE AREA
* ST RØ,8(R13) MY SAVE AREA IN INVOKER'S SAVE AREA
* LR R13,RØ LOAD R13 WITH MY SAVE AREA
* DATA CONTROL BLOCK ADDRESSABILITY USING IHADCB,RDCB
* OPEN PRINT FILES
OPEN (PRINT,(OUTPUT),TRACE,(OUTPUT))
LA RDCB,PRINT LOAD DCB ADDRESS
TM DCBOFLGS,DCBOFOPEN OPEN SUCCESSFULLY COMPLETED ?
BO OKOPRINT ...YES
* OPEN ERROR. NOTIFY CONSOLE
MVC WTOMSG(45),=CL45'ERROR ON OPEN FILE PRINT-EXECUTION TERMINATED'
BAL RBAL2,WTO SEND MESSAGE TO CONSOLE
MVC RC,=F'16' SET I/O ERROR ON RETURN CODE
B ENDERR EXECUTION TERMINATED
```

**Data Control Block Addressability**

```
OKOPRINT DS ØH
LA RDCB,TRACE LOAD DCB ADDRESS
TM DCBOFLGS,DCBOFOPEN OPEN SUCCESSFULLY COMPLETED ?
BO OKOTRACE ...YES
OI SWTRACE,X'80' NO TRACE BECAUSE OPEN ERROR
```
OKOTRACE DS ØH

* PRINT TOP PAGE
  MVI IOAREAP,X'8B'     SKIP TO CHANNEL 1 IMMED.
  BAL RBAL2,PRINTR      PRINT DDNAME=PRINT
  MVC MSGPRT,HEADR1
  MVI IOAREAP,X'09'     SPACE 1 LINE AFTER WRITE
  BAL RBAL2,PRINTR      PRINT DDNAME=PRINT
  MVC MSGPRT,HEADR
  MVI IOAREAP,X'09'     SPACE 1 LINE AFTER WRITE
  BAL RBAL2,PRINTR      PRINT DDNAME=PRINT
  MVI IOAREAP,X'19'     SPACE 3 LINES AFTER WRITE
  BAL RBAL2,PRINTR      PRINT DDNAME=PRINT
  MVI IOAREAP,X'8B'     SKIP TO CHANNEL 1 IMMED.
  BAL RBAL2,PRINTT      PRINT DDNAME=TRACE
  MVC MSGPRT,HEADR1
  MVI IOAREAP,X'09'     SPACE 1 LINE AFTER WRITE
  BAL RBAL2,PRINTT      PRINT DDNAME=TRACE
  MVC MSGPRT,HEADR
  MVI IOAREAP,X'09'     SPACE 1 LINE AFTER WRITE
  BAL RBAL2,PRINTT      PRINT DDNAME=TRACE
  MVI IOAREAP,X'19'     SPACE 3 LINES AFTER WRITE
  BAL RBAL2,PRINTT      PRINT DDNAME=TRACE

* LOAD PROGRAM DYNALLOC
  LOADDYN DS ØH
  LOAD EP=DYNALLOC       LOAD PROGRAM DYNALLOC
  LTR R15,R15           PROGRAM LOADED CORRECTLY ?
  BZ OKLOAD            ..YES
  B ERDYNAM            ..NO

OKLOAD DS ØH
  ST R0,VDYN           SAVE PGM DYNALLOC ADDRESS
  MVC MSGPRT(20),=CL20'OK LOAD DYNALLOC PGM'
  MVI IOAREAP,X'09'     SPACE 1 LINE AFTER WRITE
  BAL RBAL2,PRINTT      PRINT DDNAME=TRACE

* OPEN LISTING DATASET (OUTPUT OF DFHCSDUP UTILITY)
* OPER OUTPUT DATASET FOR DFHCSDUP SYSIN
  OPEN (CSDPRT,(INPUT),CSDVARY,(OUTPUT))
  LA RDCB,CSDPRT       LOAD DCB ADDRESS
  TM DCBOFLGS,DCBOFOPN OPEN SUCCESSFULLY COMPLETED?
  BO OKOPRT            ...YES

* OPEN ERROR. MESSAGE ON PRINTER
  MVC MSGPRT(46),=CL46'ERROR ON OPEN FILE CSDPRT-EXECUTION TERMINATED'
  MVI IOAREAP,X'09'     SPACE 1 LINE AFTER WRITE
  BAL RBAL2,PRINTR      PRINT DDNAME=PRINT
  MVC RC,=F'16'         I/O ERROR ON RETURN CODE
  B END                  EXECUTION TERMINATED
OKOPRT   DS    ØH
LA    RDCB,CSDVARY LOAD DCB ADDRESS
TM    DCOBFGLG3,DCBOFOPN OPEN SUCCESSFULLY COMPLETED?
B0    OKOVARY ...YES

* OPEN ERROR. MESSAGE ON PRINTER
MVC    MSGPRT(47),-CL47'ERROR ON OPEN FILE CSDVARY-EXECUTION TERMINATED'
MVI    IOAREAP.X'Ø9' SPACE 1 LINE AFTER WRITE
BAL    RBAL2,PRINT PRINT DDNAME=PRINT
B     END EXECUTION TERMINATED

OKOVARY DS    ØH
* SEARCH KEYWORD (SEE TABWORD)
LA    RWKR1,TABWORD LOAD KEYWORD TABLE
ST    RWKR1,ATABW AND SAVE

GETCSD   DS    ØH
BAL    RBAL1,SEARCHW SEARCH KEYWORD IN DFHCSDUP LIST

* THE WORK FIELDS, INITIALIZED WITH A "*" IN THE 1ST BYTE,
* MUST ALL BE PRESENT BEFORE ALLOCATION
* NEXT AN OPEN AND A CLOSE (WITH DEALLOCATION) IS EXECUTED IN ORDER TO
* GET THE INFORMATION TO BE VERIFIED
CLI    FILENAME,C'*' FILE NAME NOT SET ?
BE    GETCSD ...YES
CLI    DSN,C'*' DSNAME NOT SET ?
BE    GETCSD ...YES
CLI    GROUPNAM,C'*' GROUP NAME NOT SET ?
BE    GETCSD ...YES
CLI    BACKT,C'*' LAST ENTRY IN THE LISTING (BACKUPTYPE) NOT SET ?
BE    GETCSD ...YES
NI    SWALTER,255-X'8Ø' ALL FIELDS SET

* GET INFORMATION FROM ACB
BAL    RBAL1,GETDATA GET INFORMATION FROM ACB

* COMPARES WITH THOSE OF THE LISTING
BAL    RBAL1,COMPARE COMPARE WITH THOSE OF THE DFHCSDUP LISTING

* SET FIRST BYTE TO "*"
BAL    RBAL2,RESET SET FIRST BYTE TO "*"

* SKIP TO NEXT LINE
B     GETCSD NEXT LINE

ENDCSD   DS    ØH

* OFF-LINE CYCLE
* 1) THE WORK FIELDS, INITIALIZED WITH A "*" IN THE 1ST BYTE,
* MUST ALL BE PRESENT BEFORE ALLOCATION
* 2) NEXT AN OPEN AND A CLOSE (WITH DEALLOCATION) IS EXECUTED IN ORDER TO
* GET THE INFORMATION TO BE VERIFIED
CLI    FILENAME,C'*' FILE NAME NOT SET ?
BE    ENDCSD1 ...YES
CLI    DSN,C'*' DSNAME NOT SET ?
BE    ENDCSD1 ...YES
CLI GROUPNAM,C'**' GROUP NAME NOT SET ?
BE ENDCSD1 ...YES
CLI BACKT,C'**' LAST ENTRY IN THE LISTING (BACKUPTYPE) NOT SET ?
BE ENDCSD1 ...YES
* GETS THE PARAMETERS FROM THE ACB
BAL RBAL1,GETDATA GET PARAMETERS FROM VSAM ACB
* COMPARES WITH THOSE FROM THE LISTING
BAL RBAL1,COMPARE COMPARE WITH DFHCSDUP LISTING
ENDCSD1 DS ØH
* CLOSE INPUT LIST & OUTPUT SYSIN DATASETS
CLOSE (CSDPRT,,CSDVARY)
END DS ØH
* CLOSE PRINT & TRACE DATA SETS
CLOSE PRINT
TM SWALTER,X'8Ø' NO TRACE ?
BO ENDERR ...YES NO CLOSE
CLOSE TRACE
ENDERR DS ØH
L R15,RC SET RETURN CODE
L R13,SAVEAREA+4
RETURN (14,12),RC=(15)
* SEARCH KEYWORDS IN THE DFHCSDUP LISTING
SEARCHW DS ØH
ST RBAL1,VOXBAL1 SAVE RETURN ADDRESS
SEARCHWN DS ØH
MVC CSDAREA,BLANK CLEAR I/O AREA
GET CSDPRT,CSDAREA-4 GET PRINT LINE
CLC CSDAREA+1(L'CSDAREA-1),BLANK BLANK LINE ?
BE SEARCHWN GET NEXT LINE
LA RWKR1,ATABW GET TABKEYWORD ADDRESS
ST RWKR1,ATABW SAVE ADDRESS
LOOPS DS ØH
L RWKR1,ATABW LOAD SAVED ADDRESS
CLI Ø(RWKR1),X'FF' END OF TABLE ?
BE FSW BRANCH IF YES
SR RWKR2,RWKR2 ZERO WORK REGISTER
ICM RWKR2,B'ØØ11',Ø(RWKR1) LOAD KEYWORD LENGTH
LA RWKR3,CSDAREA LOAD LINE ADDRESS
SH RWKR2,=H'1' -1 KEYWORD LENGTH FOR EXECUTE INSTRUCTION
BM NFW ??
LOOPS1 DS ØH
EX RWKR2,CLCW SEARCH KEYWORD IN PRINT LINE
BE FFW BRANCH IF FOUND
L RWKR1,ATABW LOAD TAB KEYWORD ADDR
SR RWKR2,RWKR2 ZERO WORK REGISTER
ICM RWKR2,B'ØØ11',Ø(RWKR1) LOAD KEYWORD LENGTH
SH    RWKR2,=H'1'    -1 KEYWORD LENGTH FOR EXECUTE INSTRUCTION
*
BM    NFW ??
LA    RWKR3,1(RWKR3) NEXT BYTE IN PRINT LINE ?
C    RWKR3,=A(CSDAREA+L'CSDAREA) END OF PRINT LINE ?
BH    NFW BRANCH IF YES
B    LOOPSA1 ..NO CONTINUE SEARCH IN THE PRINT LINE
*
CLCW    CLC 2(1,RWKR1),Ø(RWKR3) COMPARE KEYWORD
NFW    DS ØH WORD NOT FOUND. GO TO NEXT WORD IN TABLE
*
L    RWKR1,ATABW LOAD TAB KEYWORD ADDR
SR    RWKR2,RWKR2 ZERO WORK REGISTER
ICM    RWKR2,B'ØØ11',Ø(RWKR1) LOAD KEYWORD LENGTH
LA    RWKR1,6(RWKR1) GO TO NEXT WORD IN TABLE
AR    RWKR1,RWKR2 R1 POINT TO NEXT ELEMENT
ST    RWKR1,ATABW SAVE NEW TABLE ADDRESS
B    LOOPSA1 LOOP IN THE LINE
FFW    DS ØH WORD FOUND
ST    RWKR3,ABEGINW SAVE BEGIN OF WORD FOUND
L    RWKR1,ATABW LOAD TAB WORD ADDRESS
LH    RWKR2,Ø(RWKR1) LOAD WORD LENGTH
LA    RWKR2,2(RWKR2) ADD HALF WORD LENGTH
AR    RWKR2,RWKR1 POINT TO ROUTINE ADDRESS
ICM    RWKR2,B'1111',Ø(RWKR2) LOAD ROUTINE ADDRESS
LA    RWKR1,NFW SET RETURN ADDRESS
BR    RWKR2 EXEC KEYWORD ROUTINE
FSW    DS ØH
L    RBA1,VOXBA1 LOAD RETURN ADDRESS
BR    RBA1 RETURN TO CALLER
*
GETDATA    DS ØH
ST    RBA1,VOXBA1 SAVE RETURN ADDRESS
*
PREPARE DYNALLOC COMMON DATA AREA
* AND INITIALIZE IT TO DEFAULT DATA
MVC    TDD(TRISP-TDD),BLANK BLANK TO COMMON DATA AREA
MVC    TDD,=CL8'VSAMF' SET DDNAME
MVC    TDS,DSN SET DSNAM
MVC    TSTATUS,=CL7'KEEP' DISPOSITION
MVC    TSTATUSC,=CL7'KEEP' CONDITIONAL DISPOSITION
MVC    TDISP,=CL3'SHR' STATUS
MVC    TDSORG,=CL3'VS' DATASET ORGANIZATION VSAM
MVC    TBLKSIZ,=CL5'00000' BLOCKSIZE
MVC    TLRECL,=CL5'00000' LRECL
MVC    TBUNNO,=CL3'005' BUFNO
MVI    TRISP,C'N' IF SVC 99 ERROR NO WTO MESSAGE
XC    AREAS,AREAS
L    R15,VDYN LOAD DYNALLOC ADDRESS
CALL (15),(TDD) CALL DYNALLOC PROGRAM

CLI TRISP,X'Ø' DYNAMIC ALLOCATION OK ?
BNE NODYN .NO
MVC MSGPRT(18),=CL18'OK ALLOCATION DSN:'
MVC MSGPRT+18(L'TDS),TDS DATASET NAME
MVI IOAREAP,X'Ø9' SPACE 1 LINE AFTER WRITE
BAL RBAL2,PRINTT PRINT DDNAME=TRACE
OPEN VSAMACB OPEN ACB VSAM
B *+4(R15)
B OKOPEN RC=Ø
B OKOPEN RC=4
B EROPEN RC=8
OKOPEN DS ØH OPEN SUCCESSFULLY
MVC MSGPRT(18),=CL18'OK OPEN DSN:'
MVC MSGPRT+18(L'TDS),TDS DATASET NAME
MVI IOAREAP,X'Ø9' SPACE 1 LINE AFTER WRITE
BAL RBAL2,PRINTT PRINT DDNAME=TRACE
* GET INFORMATION FROM ACB (KEYLEN & LRECL)
SHOWCB ACB=VSAMACB, / AREA=AREAS,
OBJE=DATA,
FIELDS=(DDNAME, 
KEYLEN, 
LRECL),
LENGTH=L'AREAS
LTR RWKR15,RWKR15 SHOWCB OK ?
BNZ ERSHOW ...NO
MVC MSGPRT(18),=CL18'OK SHOWCB DSN:'
MVC MSGPRT+18(L'TDS),TDS
MVI IOAREAP,X'Ø9' SPACE 1 LINE AFTER WRITE
BAL RBAL2,PRINTT PRINT DDNAME=TRACE
MVI MSGPRT,C'**'
MVI IOAREAP,X'Ø9' SPACE 1 LINE AFTER WRITE
BAL RBAL2,PRINTR PRINT DDNAME=PRINT
MVC MSGPRT(L'TDS),TDS
MVI IOAREAP,X'Ø9' SPACE 1 LINE AFTER WRITE
BAL RBAL2,PRINTR PRINT DDNAME=PRINT
MVC MSGPRT(Ø7),=CLØ7'DDNAME:'
MVC MSGPRT+Ø7(8),DDNAME
MVC MSGPRT+16(Ø7),=CLØ7'KEYLEN:'
L RWKR1,KEYLEN LOAD KEYLENGTH
CVD RWKR1,DOUBLE PREPARE TO EDIT
UNPK MSGPRT+23(5),DOUBLE+5(3)
OI MSGPRT+23+4,X'FØ'
MVC MSGPRT+39(Ø6),=CLØ6'LRECL:'
L RWKR1,LRECL LOAD ACB LRECL
CVD RWKR1,DOUBLE PREPARE TO EDIT
UNPK MSGPRT+45(3),DOUBLE+6(2)
OI MSGPRT+45+2,X'FØ'
MVI IOAREAP,X'Ø9' SPACE 1 LINE AFTER WRITE
BAL RBAL2,PRINTR PRINT DDNAME=PRINT
CLOSE VSAMACB           CLOSE & DEALLOCATE VSAM FILE
B     O+4(R15)
B     OKCLOS   RC=Ø
B     OKCLOS   RC=4
B     ERCLOS   RC=8

OKCLOS  DS  ØH
MVC  MSGPRT(18),=CL18'OK  CLOSE DSN:'
MVC  MSGPRT+18(L'TDS),TDS
MVI  IOAREAP,X'Ø9'   SPACE 1 LINE AFTER WRITE
BAL  RBAL2,PRINTT    PRINT DDNAME=TRACE
MVI  MSGPRT,C''
MVI  IOAREAP,X'Ø9'   SPACE 1 LINE AFTER WRITE
BAL  RBAL2,PRINTTR   PRINT DDNAME=PRINT
L     RBAL1,VOXBAL1  LOAD RETURN ADDRESS
BR    RBAL1          RETURN TO CALLER

EROPEN  DS  ØH
MVC  MSGPRT(18),=CL18'KO  OPEN DSN:'
MVC  MSGPRT+18(L'TDS),TDS
MVI  IOAREAP,X'Ø9'   SPACE 1 LINE AFTER WRITE
BAL  RBAL2,PRINTT    PRINT DDNAME=TRACE
BAL  RBAL2,RESET SET FIRST BYTE TO "**"
CLOSE VSAMACB
MVI  TRISP,C'Y'   DEALLOCATION WITH WTO MSGS
XC  AREAS,AREAS
L     R15,VDYN    LOAD DYNALLOC ADDRESS
CALL (15),(TDD)  CALL DYNALLOC PROGRAM
CLI  TRISP,X'Ø'   DEALLOCATION SUCCESSFULLY ?
BE    OKUNLC       ..YES
MVC  MSGPRT(18),=CL18'KO  DEALLOC DSN:'
MVC  MSGPRT+18(L'TDS),TDS
MVI  IOAREAP,X'Ø9'   SPACE 1 LINE AFTER WRITE
BAL  RBAL2,PRINTT    PRINT DDNAME=TRACE

OKUNLC  DS  ØH
MVC  MSGPRT(18),=CL18'OK  DEALLOC DSN:'
MVC  MSGPRT+18(L'TDS),TDS
MVI  IOAREAP,X'Ø9'   SPACE 1 LINE AFTER WRITE
BAL  RBAL2,PRINTT    PRINT DDNAME=TRACE
MVI  IOAREAP,X'19'   SPACE 3 LINES AFTER WRITE
MVC  MSGPRT,HEADR1
MVC  MSGPRT+18(L'TDS),TDS
MVI  IOAREAP,X'Ø9'   SPACE 1 LINE AFTER WRITE
BAL  RBAL2,PRINTT    PRINT DDNAME=TRACE
BAL  RBAL2,RESET SET FIRST BYTE TO "**"
B GETCSD

ERSHOW DS ØH
MVC MSGPRT(18),=CL18'KO SHOWCB DSN:'
MVC MSGPRT+18(L'TDS),TDS
MVI IOAREAP,X'09' SPACE 1 LINE AFTER WRITE
BAL RBAL2,PRINTT PRINT DDNAME=TRACE
MVI IOAREAP,X'19' SPACE 3 LINES AFTER WRITE
MVC MSGPRT,HEADR1
BAL RBAL2,PRINTT PRINT DDNAME=TRACE
BAL RBAL2,RESET SET FIRST BYTE TO "*"
B GETCSD

NODYN DS ØH
MVC MSGPRT(18),=CL18'KO ALLOCATION DSN:'
MVC MSGPRT+18(L'TDS),TDS
XC DOUBLE,DOUBLE
MVC DOUBLE+L'DOUBLE-1(1),TRISP
MVC MSGPRT+18+L'TDS+1(3),=CL03'RC:'
UNPK MSGPRT+18+L'TDS+4(9),DOUBLE(L'DOUBLE+1)
TR MSGPRT+18+L'TDS+4(9),TABEX-240
MVC MSGPRT+18+L'TDS+4+8(L'MSGPRT-18-L'TDS-4-8),BLANK
MVI IOAREAP,X'09' SPACE 1 LINE AFTER WRITE
BAL RBAL2,PRINTT PRINT DDNAME=TRACE
MVI IOAREAP,X'19' SPACE 3 LINES AFTER WRITE
MVC MSGPRT,HEADR1
BAL RBAL2,PRINTT PRINT DDNAME=TRACE
BAL RBAL2,RESET SET FIRST BYTE TO "*"
B GETCSD

ERDYNAM DS ØH
MVI IOAREAP,X'09' SPACE 1 LINE AFTER WRITE
MVC MSGPRT(27),=CL27'LOAD ERROR PROGRAM DYNALLOC'
BAL RBAL2,PRINTR PRINT DDNAME=PRINT
CLOSE (PRINT,,TRACE) CLOSE PRINT & TRACE DATASETS
L R13,SAVEAREA+4
RETURN (14,12),RC=16

* RESET WORK FIELDS
RESET DS ØH
ST RBAL2,VOXBAL2
MVC FILENAME,BLANK CLEAR
MVI FILENAME,C'**' INITIALIZE FIRST BYTE WITH *
MVC GROUPNAM,BLANK CLEAR
MVI GROUPNAM,C'**' INITIALIZE FIRST BYTE WITH *
MVC DSN,BLANK CLEAR
MVI DSN,C'**' INITIALIZE FIRST BYTE WITH *
MVC BACKT,BLANK CLEAR
MVI BACKT,C'**' INITIALIZE FIRST BYTE WITH *
L RBAL2,VOXBAL2 RESTORE BRANCH REGISTER
BR RBAL2 RETURN TO CALLER

* COMPARES THE ACB DATA WITH THAT OF THE LISTING
COMPARE DS ØH
ST RBAL1,VOXBAL1 SAVE RETURN ADDRESS
CLC  KEYLEN,=F'Ø'   IF Ø VSAM RRDS OR ESDS
BNE  COMPARK      NO RRDS/ESDS
MVC  KEYLEN,=F'4'  IF RRDS OR ESDS FORCE KEYLENGTH TO 4

COMPARK  DS  ØH
L  RWKR1.KEYLEN   LOAD KEYLENGTH
CVD RWKR1.DOUBLE  PREPARE TO EDIT
MVC MSGPRT(2Ø),=CL2Ø'KEYLENGTH(CSD/FILE):'
MVC MSGPRT+2Ø(L'KEYF),LKEYF
UNPK MSGPRT+2Ø+L'KEYF+1(3),DOUBLE+6(2)
OI MSGPRT+2Ø+L'KEYF+1+2,X'FØ'
MVI IOAREAP,X'Ø9'  SPACE 1 LINE AFTER WRITE
BAL RBAL2.PRINTT  PRINT DDNAME=TRACE
PACK DOUBLE,LKEYF
CVB RWKR1.DOUBLE
C RWKR1.KEYLEN   KEYLENGTH OK IN CSD
BE OKKEY          ...YES
BAL RBAL2.TESALTER COMPLETE SYSIN WITH DSNAME
MVI CSDAREAV,C' '
MVC CSDAREAV+1(L'CSDAREAV-1),CSDAREAV
MVC CSDAREAV(L'VARFIX),VARFIX
MVC CSDAREAV+11(L'FILENAME),FILENAME
MVC CSDAREAV+27(L'GROUPNAM),GROUPNAM
MVC CSDAREAV+L'VARFIX+1(2Ø),=CL2Ø'KEYLENGTH(XXX)'
L  RWKR1.KEYLEN   PREPARE KEYLENGTH
CVD RWKR1.DOUBLE  TO EDIT
UNPK KEYWORK(5),DOUBLE+5(3)
OI KEYWORK+4,X'FØ'
MVC CSDAREAV+L'VARFIX+11(L'KEYWORK-2),KEYWORK+2
PUT CSDVARY,CSDAREAV WRITE DHCSDUP SYSIN
MVI IOAREAP,X'Ø9'  SPACE 1 LINE AFTER WRITE
MVC MSGPRT(L'CSDAREAV),CSDAREAV
BAL RBAL2.PRINTT  PRINT DDNAME=PRINT

OKKEY  DS  ØH
L  RWKR1.LRECL    LOAD ACB RECORD LENGTH
CVD RWKR1.DOUBLE
MVC MSGPRT(2Ø),=CL2Ø'LRECL(CSD/FILE) :
MVC MSGPRT+2Ø(5),RECSZF
UNPK MSGPRT+26(5),DOUBLE+5(3)
OI MSGPRT+26+4,X'FØ'
MVI IOAREAP,X'Ø9'  SPACE 1 LINE AFTER WRITE
BAL RBAL2.PRINTT  PRINT DDNAME=TRACE
PACK DOUBLE,RECSZF CONVERT TO DECIMAL CSD RECORD LENGTH
CVB RWKR1.DOUBLE
C RWKR1.LRECL    CSD = ACB ?
BE OKLRECL       ...YES
BAL RBAL2.TESALTER WRITE DSNAME TO SYSIN
MVI CSDAREAV,C' '
MVC CSDAREAV+1(L'CSDAREAV-1),CSDAREAV
MVC CSDAREAV(L'VARFIX),VARFIX
MVC CSDAREAV+11(L'FILENAME),FILENAME
MVC CSDAREAV+27(L'GROUPNAM),GROUPNAM
MVC CSDAREAV+L'VARFIX+1(20),=CL20'RECORDSIZE(XXXXX)'
L RWKR1,LRECL LOAD ACB RECORD LENGTH
CVD RWKR1.DOUBLE PREPARE TO EDIT
UNPK CSDAREAV+L'VARFIX+12(5),DOUBLE+5(3)
OI CSDAREAV+L'VARFIX+12+4,X'F0'
PUT CSDVARY,CSDAREAV
MVC MSGPRT(L'CSDAREAV),CSDAREAV
MVI IOAREAP,X'99' SPACE 1 LINE AFTER WRITE
BAL RBAL2,PRINTR PRINT DDNAME=PRINT

OKLRECL DS ØH
MVI CSDAREAV,C' '
MVC CSDAREAV+1(L'CSDAREAV-1),CSDAREAV
MVC CSDAREAV(L'VARFIX),VARFIX
MVC CSDAREAV+11(L'FILENAME),FILENAME
MVC CSDAREAV+27(L'GROUPNAM),GROUPNAM

FCOMPAR DS ØH
MVC MSGPRT,HEADR1
MVI IOAREAP,X'19' SPACE 3 LINES AFTER WRITE
BAL RBAL2,PRINTT PRINT DDNAME=TRACE
TM SWALTER,X'80'
BZ FCOMPAR
MVI IOAREAP,X'99' SPACE 1 LINE AFTER WRITE
MVC MSGPRT(L'CSDAREAV),HEADR1
BAL RBAL2,PRINTR PRINT DDNAME=PRINT

FCOMPAR1 DS ØH
L RBAL1,VOXBAL1 LOAD RETURN ADDRESS
BR RBAL1 RETURN TO CALLER

TESALTER DS ØH
* IF "ALTER COMMAND" ALREADY WRITTEN ON SYSIN
* ALSO WRITE THE DATASET NAME (COMMENT)
TM SWALTER,X'80' ALTER COMMAND WRITTEN ?
BOR RBAL2 ...NO
ST RBAL2,VOXBAL2 SAVE RETURN ADDRESS
OI SWALTER,X'80' SET SWITCH
MVC CSDAREAV,HEADR1
MVI CSDAREAV,C'*'
PUT CSDVARY,CSDAREAV
MVC CSDAREAV,BLANK BLANK I/O AREA
MVI CSDAREAV,C'*'
MVC CSDAREAV+2(L'TDS),TDS MOVE DSNAME
PUT CSDVARY,CSDAREAV
L RBAL2,VOXBAL2 RESTORE BRANCH REGISTER
BR RBAL2 RETURN TO CALLER

* ROUTINES FOR HANDLING KEYWORDS
* FILE WORD
FILE DS ØH
ST RWKR1,ARET SAVE RETURN ADDRESS
L RWKR1,ABEGINW FILE( POINT TO BEGINNING OF WORD
CLC ¬'FILE()',Ø(RWK1) NULL WORD ?
BNE FILESET NO FILE SET
L RWKR1,ARET LOAD RETURN ADDRESS
BR RWKR1 RETURN TO CALLER

FILESET DS ØH
CLI FILENAME,C'**' FILE ALREADY SET ?
BE FILEE ... NO
BAL RBAL2,RESET RESET WORK FIELDS

FILEE DS ØH
L RWKR1,ATABW LOAD TAB KEYWORD ADDR
SR RWKR2,RWK1 CLEAR WORK REGISTER
ICM RWKR2,B'ØØ11',Ø(RWK1) LOAD WORD LENGTH
L RWKR1,ABEGINW FILE( POINT TO BEGINNING OF WORD
AR RWKR1,RWK1 POINT TO FILENAME
LA RWKR2,FILENAME CLEAR FILENAME WORK
MVC FILENAME,BLANK FIELD

LFILE DS ØH
CLI Ø(RWK1),C')' END OF WORD ?
BE SWEND ... YES
MVC Ø(1,RWK2),Ø(RWK1) MOVE CHAR IN WORK FLD
LA RWKR1,1(RWK1) NEXT CHAR IN WORD
LA RWKR2,1(RWK2) NEXT BYTE IN FILENAME

SWEND DS ØH FILENAME COMPLETED
MVC MSGPRT(3),=CL3'===' MVI IOAREAP,X'Ø9' SPACE 1 LINE AFTER WRITE
BAL RBAL2,PRINTT PRINT DDNAME=TRACE
MVC MSGPRT(L'FILENAME),FILENAME MOVE FILENAME
MVI IOAREAP,X'Ø9' SPACE 1 LINE AFTER WRITE
BAL RBAL2,PRINTT PRINT DDNAME=TRACE
MVC MSGPRT(L'GROUPNAM),GROUPNAM MOVE GROUPNAME
MVI IOAREAP,X'Ø9' SPACE 1 LINE AFTER WRITE
BAL RBAL2,PRINTT PRINT DDNAME=TRACE
MVC MSGPRT(3),=CL3'===' MVI IOAREAP,X'Ø9' SPACE 1 LINE AFTER WRITE
BAL RBAL2,PRINTT PRINT DDNAME=TRACE
L RWKR1,ARET LOAD RETURN ADDRESS
BR RWKR1 RETURN

GROUP DS ØH
CLI GROUPNAM,C'**' GROUP NAME ALREADY SET ?
BNER RWKR1 ... YES RETURN
ST RWKR1,ARET SAVE RETURN ADDRESS
L RWKR1,ABEGINW GROUP( LOAD BEGINNING OF WORD ADDR
CLC =C'GROUP()',Ø(RWKR1) NULL WORD ?
BNE GROUPSET ...NO
L RWKR1,ARET LOAD RETURN ADDRESS
BR RWKR1 RETURN

GROUPSET DS ØH
L RWKR1,ATABW LOAD TABLE ADDRESS
SR RWKR2,RWKR2 CLEAR WORK REGISTER
ICM RWKR2,B'ØØ11',Ø(RWKR1) LOAD WORD LENGTH
L RWKR1,ABEGINW GROUP( LOAD BEGINNING OF WORD
AR RWKR1,RWKR2 POINT TO NAME
CLI Ø(RWKR1),C'*' GENERIC NAME ?
BE SWEND ...YES, GO TO COMMON END ROUTINE
* AND SEARCH ANOTHER GROUP NAME IN THE LIST
CLI Ø(RWKR1),C'+' GENERIC NAME ?
BE SWEND ...YES, GO TO COMMON END ROUTINE
* AND SEARCH ANOTHER GROUP NAME IN THE LIST
LA RWKR2,GROUPNAM CLEAR WORK FIELD
MVC GROUPNAM,BLANK WITH BLANK

LGROUPN DS ØH
CLI Ø(RWKR1),C')' END OF FIELD ?
BE SWEND ...YES
CLI Ø(RWKR1),C'*' GENERIC NAME ?
BE LGROUPR ...YES
CLI Ø(RWKR1),C'+' GENERIC NAME ?
BE LGROUPR ...YES
MVC Ø(1,RWKR2),Ø(RWKR1) MOVE CHARACTER IN WORK FIELD
LA RWKR1,1(RWKR1) NEXT CHARACTER IN LISTING
LA RWKR2,1(RWKR2) NEXT BYTE IN WORK FIELD
C RWKR2,=A(GROUPNAM+L'GROUPNAM) END OF WORK FIELD ?
BH SWEND ...YES
B LGROUPN CONTINUE LOOP

LGROUPR DS ØH
MVC GROUPNAM,BLANK RESET WORK FIELD
MVI GROUPNAM,C'*' SET * IN FIRST BYTE
B SWEND GO TO COMMON END ROUTINE

* DSNAME WORD
DSNAME DS ØH
CLI DSN,C'*' FIELD SET ?
BNER RWKR1 ...YES
ST RWKR1,ARET SAVE RETURN ADDRESS
L RWKR1,ABEGINW DSNAME(, BEGINNING OF WORD
CLC =C'DSNAME()',Ø(RWKR1) NULL WORD ?
BNE DSNSET ...NO
L RWKR1,ARET LOAD RETURN ADDRESS
BR RWKR1 RETURN

DSNSET DS ØH
L RWKR1,ATABW LOAD TABLE ADDRESS
SR RWKR2,RWKR2 CLEAR REGISTER
ICM RWKR2,B'ØØ11',Ø(RWKR1) LOAD WORD LENGTH
L   RWKR1,ABEGINW   DSNAME(... BEGINNING OF WORD
AR   RWKR1,RWKR2   POINT TO NAME
LA   RWKR2,DSN   SAVE DATASET NAME
MVC   DSN,BLANK
LDSN   DS   ØH
CLI   Ø(RWKR1),C')'   END OF KEYWORD ?
BE   LDSNE   ...YES
MVC   Ø(1,RWKR2),Ø(RWKR1)   MOVE CHARACTER
LA   RWKR1,1(RWKR1)   NEXT CHARACTER IN LINE
LA   RWKR2,1(RWKR2)   NEXT BYTE IN WORK FIELD
C   RWKR2,=A(DSN+L'DSN)   END OF FIELD ?
BH   LDSNE   ...YES
B   LDSN   LOOP
LDSNE   DS   ØH
CLC   DSN,BLANK   DATASET NAME IS BLANK ?
BNE   SWEND   ...NO
MVI   DSN,C'**
B   SWEND   GO TO COMMON END ROUTINE

* RECORDSIZE WORD
RECSZ   DS   ØH
ST   RWKR1,ARET   SAVE RETURN ADDRESS
L   RWKR1,ABEGINW   RECORDSIZE(... BEGINNING OF WORD
CLC   =C'RECORDSIZE()',Ø(RWKR1)   RECORDSIZE MISSING ?
BNE   RSZSET   ...NO
L   RWKR1,ARET   LOAD RETURN ADDRESS
BR   RWKR1   RETURN
RSZSET   DS   ØH
L   RWKR1,ATABW   LOAD TABLE ELEMENT ADDRESS
SR   RWKR2,RWKR2   CLEAR
ICM   RWKR2,B'ØØ11',Ø(RWKR1)   LOAD WORD LENGTH
L   RWKR1,ABEGINW   RECORDSIZE(... BEGINNING OF WORD
AR   RWKR1,RWKR2   POINT TO NAME
LRSZØ   DS   ØH   END OF WORD SEARCH
CLI   Ø(RWKR1),C')'   END OF WORD ?
BE   FRSZØ   ...YES
LA   RWKR1,1(RWKR1)   NEXT BYTE
B   LRSZØ   LOOP
FRSZØ   DS   ØH
SH   RWKR1,=H'1'   POINT TO LAST NUMBER
LA   RWKR2,RECSZF+L'RECSZF-1   POINT TO LAST BYTE IN WORK FIELD
MVC   RECSZF,=5C'Ø'   INITIALIZE WORK FIELD TO Ø
LRSZ   DS   ØH
CLI   Ø(RWKR1),C'('   BEGINNING OF WORD ?
BE   SWEND   ...YES GO TO COMMON END ROUTINE
MVC   Ø(1,RWKR2),Ø(RWKR1)   MOVE FROM LIST TO WORK FIELD
SH   RWKR1,=H'1'   PREVIOUS CHARACTER
SH   RWKR2,=H'1'   PREVIOUS BYTE
C   RWKR2,=A(RECSZF)   BEGINNING OF WORD FIELD ?
BL   SWEND   ...YES GO TO COMMON END ROUTINE
B   LRSZ   LOOP

* KEYLENGTH WORD

LKEY DS ØH
ST RWKR1,ARET SAVE RETURN ADDRESS
L RWKR1,ABEGINW KEYLENGTH(.. BEGIN OF WORD
CLC =C'KEYLENGTH()',Ø(RWKR1) KEYLENGTH NOT SET ?
BNE KLNSET ...NO
L RWKR1,ARET LOAD RETURN ADDRESS
BR RWKR1 RETURN

KLNSET DS ØH
L RWKR1,ATABW TABLE ELEMENT ADDRESS
SR RWKR2,RWKR2 CLEAR
ICM RWKR2,B'ØØ11',Ø(RWKR1) KEYWORD LENGTH
L RWKR1,ABEGINW KEYLENGTH(.. BEGINNING OF WORD
AR RWKR1,RWKR2 POINT TO NAME

LKLNØ DS ØH END OF KEYWORD SEARCH
CLI Ø(RWKR1),C')' END OF KEYWORD
BE FKLNØ ...YES
LA RWKR1,1(RWKR1) NEXT NUMBER
B LKLNØ LOOP

FKLNØ DS ØH POINT TO LAST BYTE IN WORK FIELD
LA RWKR2,LKEYF+L'LKEYF-1 END OF WORK FIELD
MVC LKEYF,=5C'Ø' INITIALIZE TO Ø

LKLN DS ØH BEGINNING OF WORD ?
BE SWEND GO TO COMMON END ROUTINE
MVC Ø(1,RWKR2),Ø(RWKR1) MOVE FROM LIST TO WORK FIELD
SH RWKR1,=H'1' PREVIOUS CHARACTER
SH RWKR2,=H'1' PREVIOUS BYTE
C RWKR2,=A(LKEYF) BEGINNING OF WORK FIELD
BL SWEND ...YES GO TO COMMON END ROUTINE
B LKLN LOOP

* BACKUPTYPE WORD

BACKTYPE DS ØH
CLI BACKT,C'*' FIELD SET ?
BNER RWKR1 ...YES
ST RWKR1,ARET SAVE RETURN ADDRESS
L RWKR1,ABEGINW BACKUPTYPE(.. BEGINNING OF WORD
CLC =C'BACKUPTYPE()',Ø(RWKR1) BACKUPTYPE NOT SET ?
BNE BACKSET ...NO
L RWKR1,ARET LOAD RETURN ADDRESS
BR RWKR1 RETURN

BACKSET DS ØH
L RWKR1,ATABW TABLE ELEMENT ADDRESS
SR RWKR2,RWKR2 CLEAR
ICM RWKR2,B'ØØ11',Ø(RWKR1) KEYWORD LENGTH
L RWKR1,ABEGINW BACKUPTYPE(..BEGINNING OF NAME
AR RWKR1,RWKR2 CLEAR
LA RWKR2,BACKT WORK FIELD ADDRESS
MVC BACKT,BLANK BLANK
LBACK  DS  ØH
CLI  Ø(RWKR1),C')'    END OF KEYWORD ?
BE  SWEND  ...YES
MVC  Ø(1,RWKR2),Ø(RWKR1) MOVE FROM LINE TO WORK FIELD
LA  RWKR1,1(RWKR1) NEXT CHARACTER
LA  RWKR2,1(RWKR2) NEXT BYTE
C  RWKR2,=A(BACKT+L'BACKT) END OF WORK FIELD ?
BH  SWEND  ...YES GO TO COMMON END ROUTINE
B  LBACK  LOOP

* PRINT REPORT
PRINTR  DS  ØH
ST  RBAL2,VOXBAL2 SAVE RETURN ADDRESS
PUT  PRINT,IOAREAP
MVC  MSGPRT,BLANK CLEAR PRINT LINE
L  RBAL2,VOXBAL2 LOAD RETURN ADDRESS
BR  RBAL2 RETURN TO CALLER

* TRACE REPORT
PRINTT  DS  ØH
ST  RBAL2,VOXBAL2 SAVE RETURN ADDRESS
TM  SWTRACE,X'8Ø' NO TRACE DDNAME ?
BO  NOPRINTT  ... YES
PUT  TRACE,IOAREAP

NOPRINTT  DS  ØH
MVC  MSGPRT,BLANK CLEAR PRINT LINE
L  RBAL2,VOXBAL2 LOAD RETURN ADDRESS
BR  RBAL2 RETURN TO CALLER

* WRITE TO OPERATOR ROUTINE
WTO  DS  ØH
ST  RBAL2,VOXBAL2 SAVE RETURN ADDRESS
MVC  WTOHD1(L'PGMNAME),PGMNAME INITIALIZE WITH PROGRAM NAME
LA  R1,WTOBLK
SVC  35
MVC  WTOMSG,BLANK CLEAR PRINT LINE
L  RBAL2,VOXBAL2 LOAD RETURN ADDRESS
BR  RBAL2 RETURN TO CALLER

* I/O ERROR HANDLER FOR SEQUENTIAL DATASETS
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* SYNAD EXIT *
* USER ERROR ANALYSIS ROUTINE *
* QUEUED SEQUENTIAL ACCESS METHOD -QSAM- *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
IOERRQS  DS  ØH
CP  SYQSMCTR,SYQSMCNT ERROR COUNTER
BH  SYQSMABE
SYNADAF ACSMETH=QSAM QSAM METHOD
MVC  WTOMSG(Ø6),=CLØ6'IOE **'
MVC  WTOMSG+6(78),5Ø(1) MOVE OUTPUT MESSAGE
AP  SYQSMCTR,SYQSMINC ADD CTR CHECK
UNPK  SYQSMCHK+6(3),SYQSMCTR EDIT CTR CHECK
DI  SYQSMCHK+8,X'FØ'

MVC WTONSG+85(L'SYQSMCHK),SYQSMCHK MOVE OUTPUT MESSAGE
BAL RBAL2,WTO SEND MESSAGE TO CONSOLE
SYNADRLS RELEASE SAVE AREA
LTR R0,R0 IF X'08' ERROR SINADAF EXIT
BNZ SYQSMER
BR R14 RETURN IOCS
SYQSMER DS 0H ERROR EXIT SYQSMRLS
ST R14,SYQSMR14
MVC WTONSG(27),=CL27'SYNADRLS EXIT ERROR ?????'
BAL RBAL2,WTO SEND MESSAGE TO CONSOLE
L R14,SYQSMR14
BR R14 RETURN
SYQSMABE DS 0H ABEND EXIT
SR R1,R1 CLEAR REG.1
IC R1,SYQSMCTR+1 +1 CTR ERROR
SRL R1,4 SHIFT 4 BIT
ABEND (R1),DUMP,STEP
SYQSMR14 DC A(0) SAVE REG 14
SYQSMCTR DC PL2'0' CTR ERROR COUNTER
SYQSMINC DC PL1'1' CTR INCREMENT
SYQSMCNT DC PL1'6' RETRY COUNTER
SYQSMCHK DC CL13'CTRCHK ****'
CNOP 2,4
* DATA DEFINITIONS & FILES
LTORG
* WTO CONTROL BLOCK
CNOP 0,4
WTOBLK DS 0H
DC Y(WTOBLKE-WTOBLK)
DC B'0000000000000000' MCSFLAGS
WTOHD1 DS 0CL16
DC CL8',',CL1'-'
WTONSG DC CL107' '
WTOBLKE EQU *
* END OF WTO CONTROL BLOCK
SAVEAREA DS 0D
WORD1 DC F'0' PL/I ONLY
WORD2 DC F'0' ADDRESS OF THE CALLER'S SAVE AREA
WORD3 DC F'0' ADDRESS OF THE SAVEAREA OF THE CALLED PGM
WORD4 DC F'0' REGISTER 14 RETURN ADDRESS WITHIN THE
* CALLING PHASE
WORD5 DC F'0' REGISTER 15 ENTRY POINT ADDRESS OF THE
* CALLED PHASE
WORD6 DC F'0' REGISTER 0
WORD7 DC F'0' REGISTER 1
WORD8 DC F'0' REGISTER 2
WORD9 DC F'0' REGISTER 3
WORD10 DC F'0' REGISTER 4
WORD11 DC F'0' REGISTER 5
WORD12 DC F'0' REGISTER 6
WORD13 DC F'Ø' REGISTER 7
WORD14 DC F'Ø' REGISTER 8
WORD15 DC F'Ø' REGISTER 9
WORD16 DC F'Ø' REGISTER 10
WORD17 DC F'Ø' REGISTER 11
WORD18 DC F'Ø' REGISTER 12
ATABW DC A(Ø) TABWORD ADDRESS
ABEGINW DC A(Ø) KEYWORD BEGIN ADDRESS
ARET DC A(Ø) RETURN ADDRESS FROM ROUTINES FOR
* HANDLING OF KEYWORDS
* WORK FIELDS SET WITH THE DATA FORM THE DFHCSUDP LIST
FILENAME DC CL8'**' FILE NAME
GROUPNAM DC CL8'**' CSD GROUP NAME
DSN DC CL44'**' DATASET NAME
BACKT DC CL7'**' STATIC/DYNAMIC
RECSZF DC CL5'00000' RECORD SIZE
LKEYF DC CL3'000' KEY LENGTH
* END OF WORK FIELDS
VDYN DC A(Ø) DYNALLOC PROGRAM ADDRESS
* DYNALLOC PROGRAM COMMON DATA AREA
TDD DC CL8'VSAMF'
TDS DC CL44'
TSTATUS DC CL7'KEEP'
TSTATUSC DC CL7'KEEP'
TDISP DC CL3'SHR'
TLABEL DC CL3' EG SL NL BLP ...
TUNIT DC CL5' EG 3480 3380 SYSDA ..... 
TVOLSER DC CL6' EG SM1820
TSPACET DC CL1' C = CYLINDERS T = TRACKS
TSPACEP DC CL3' PRIMARY SPACE EG Ø20
TSPACES DC CL3' SECONDARY SPACE EG Ø10
TTAPES DC CL4' TAPE DATASET SEQUENCE
TDCBR DC CL8' REFERENCE TO DDNAME FOR DCB PARAMETERS
TDSORG DC CL3'VS' DATASET ORGANIZATION EG PS PO ..
TMEMBER DC CL3' MEMBER FOR DS PARTITIONED
TRECFM DC CL3' RECORD FORMAT EG F FB FBS
TBKLKSIZ DC CL5'00000' BLOCKSIZE
TLRECL DC CL5'00000' LRECL
TBUFNO DC CL3'005' BUFNO
TOPTCD DC CL3' OPTCD
TRISP DC X'Ø' AT CALL TIME: N = ALLOCATION AND
* NO WTO IF SVC 99 ERROR
* X = DEALLOCATION AND
* NO WTO IF SVC 99 ERROR
* Y = DEALLOCATION AND
* WTO IF SVC 99 ERROR
* OTHERWISE
* ALLOCATION AND
* WTO IF SVC 99 ERROR
AFTER CALL : RESPONSE BYTE
VALUES:
OK X'ØØ'
KO NE X'ØØ'

R15 AFTER SVC 99
X'FF' (ERROR CODE NE Ø
AFTER SVC 99)

END OF COMMON DATA AREA

VSAM ACB
VSAMACB ACB AM=VSAM,
DDNAME=VSA MF,
MACRF=(SEQ,IN)

VSAM RPL
RPL RPL ACB=VSAMACB,
AM=VSAM,
AREA=IOADDR,
AREALEN=L'IOADDR,  
OPTCD=(LOC,SEQ,NUP)

DCB LIST FILE WRITTEN BY DFHCSDUP UTILITY PROGRAM
CSDPRT DCB DSORG=PS,LRECL=125,MACRF=GM,SYNAD=IOERRQQS,EODAD=ENDCSD,
DDNAME=CSDPRT,RECFM=VBA

DCB SYSIN FILE FOR DFHCSDUP
CSDVARY DCB DSORG=PS,LRECL=8Ø,MACRF=PM,SYNAD=IOERRQQS,  
DDNAME=CSDVARY

DCB REPORT FILE
PRINT DCB DSORG=PS,LRECL=133,BLKSIZE=133,MACRF=PM,SYNAD=IOERRQQS,  
RECFM=FM,DDNAME=PRINT

DCB TRACE FILE
TRACE DCB DSORG=PS,LRECL=133,BLKSIZE=133,MACRF=PM,SYNAD=IOERRQQS,  
RECFM=FM,DDNAME=TRACE

I/O AREA CSDPRT, CSDVARY
DC XL4'Ø' RRD W
CSDAREA DC CL121'
CSDAREAV DC CL8Ø'

I/O AREA - REPORT & TRACE FILE
IOAREAP DS ØCL133
DC X'Ø' I/O COMMAND CODE

MSGPRT DC CL132'
HEADR1 DS ØCL132
DC 132C'-'

HEADR DS ØCL132
DC CL132'-

S CHECKING FROM DFHCSD'
ORG +-1
DC C'-'

HEADRB DS ØCL132
DC CL132'-

APPLICATION TRACE'
ORG  *-1
DC  C'-'

* WORK FIELDS
SWALTER  DC  X'Ø' X'8Ø' = ALTER COMMAND BUILD
SWTRACE  DC  X'Ø' X'8Ø' = NO PRINT TRACE
DOUBLE  DC  D'Ø'
RC  DC  F'Ø' PROGRAM RETURN CODE : 16=I/O ERROR
VOXBAL1  DC  A(Ø) SAVE ADDRESS BAL LEVEL 1
VOXBAL2  DC  A(Ø) SAVE ADDRESS BAL LEVEL 2
IOADDR  DC  A(Ø) I/O AREA ADDRESS FOR VSAM FILE

* SHOWCB FIELDS
  DS  ØD
AREAS  DS  ØXL16
DDNAME  DC  D'Ø'
KEYLEN  DC  F'Ø'
LRECL  DC  F'Ø'

* END OF SHOWCB FIELDS
KEYWORK  DC  CL5'ØØØØØ' WORK AREA TO KEYLEN EDIT
BLANK  DC  CL132' '
TABEX  DC  256X'Ø'
  ORG  TABEX+X'FØ'
  DC  C'0123456789ABCDEF'
  ORG
VARFIX  DC  CL36'ALTER FILE(XXXXXXXX) GROUP(XXXXXXXX)'

* KEYWORDS FOR DFHCSDUP LIST SEARCH
TABWORD  DS  ØH
  * DC HL2'..' WORD LENGTH
  * DC C'.......' WORD
  * DC AL4(...) WORD ROUTINE ADDRESS
  * BACKUPTYPE MUST BE THE LAST ELEMENT IN THE TABLE
    DC  HL2'5',C'FILE(',AL4(FILE)
    DC  HL2'6',C'GROUP(',AL4(GROUP)
    DC  HL2'7',C'DSNAME(',AL4(DSNAME)
    DC  HL2'11',C'RECORDSIZE(',AL4(RECSZ)
    DC  HL2'10',C'KEYLENGTH(',AL4(LKEY)
    DC  HL2'11',C'BACKUPTYPE(',AL4(BACKTYPE)
    DC  X'FF'
CSSET
DCBD  DSORG=PS
END  CSDVER

Editor's note: the source code for DYNALLOC and sample JCL will be published in the next issue.

Giuseppe Rallo
Senior Technical Analyst
Sicilcassa spa (Italy) © Xephon 1998
Organize your disks and claim Free Space

Do you ever need to move files from one volume to another quickly and cleanly? Do you ever wonder why user X likes to allocate one cylinder instead of just one track to create a ten-line file? If you do, you may find something of interest below.

IBM supplies a utility program with MVS known as ADRDSSU. In its standard form, it is not very user-friendly. However, thanks to Mike Cowlishaw, we can easily overcome that handicap and make it work for our benefit by designing REXX programs around it. That is what I have done with the following program.

MOVEFILE is designed around the COPY option of ADRDSSU, and allows you to move a file or a group of files between volumes. Simply invoke the MOVEFILE EXEC, passing as argument the name of the file you want to move. The EXEC will ask you the original volume of the file and the destination volume. With those three arguments, the EXEC creates and submits a job that will perform the operation. ADRDSSU allows you to specify how you want the file to be allocated – in blocks, tracks, or cylinders. If you choose tracks, you can take advantage of the move operation to reduce those cylinder mammoths to more decent proportions.

USAGE NOTES

MOVEFILE is especially useful for dealing with groups of files. They can be VSAM, SEQs, or PDS. To specify a group of files, use the ADRDSSU filtering rules (see DFSMSdss Storage Administration Reference), for example:

- **IBM.*** Means any file with only two qualifiers, the first being IBM.
- **IBM.** Means any file with any number of qualifiers, the first being IBM.
- **IBM*.** Means any file with any number of qualifiers, the first beginning with IBM.
If a file that is to be processed is allocated by another task, it will not be processed. The same is true for an empty PDS. If such is the case, a return code of 8 or 4 will appear. You can ignore it, as all the other files will be processed correctly.

MOVEFILE

/* REXX MVS  *************************************************************/
/*                                                                 */
/*      MoveFile - Moves a file or group of files                      */
/*                 from one volume to another                           */
/*                                                                 */
/***************************************************************************/

jobfile = userid()||".movefile"          /* job file          */
xx = msg(off)                             /* check if jobfile */
"free da(''jobfile'')"                    /* already exists   */
okay = sysdsn(jobfile)                    /* if not, create it*/
if okay¬="OK" then do
  "free da(''jobfile'')"
  "alloc da(''jobfile'') dd(ddtemp),
    new reuse blksizer(3200) lrec(80),
    recfm(f,b) dsorg(ps) space(1 1) tracks"
  if rc ¬= Ø then do
    say "Error" rc " allocating "jobfile
    signal saida
  end
end
else do                                       /* If jobfile exists,*/
  "alloc da(''jobfile'') dd(ddtemp) shr"    /* retrieve previous */
  if rc ¬= Ø then do                       /* volume to use     */
    say "Error" rc " allocating "jobfile    /* as default      */
    signal saida
  end
  execio 5 diskr ddtemp
  do 5
    pull linha
  end
  parse var linha . "DS(INCLUDE(" dsn11 "))"
  execio 1 diskr ddtemp
  parse pull linha . "(" vol11 ")".
  execio 1 diskr ddtemp "(finis"
  parse pull linha . "(" vol22 ")".
end
arg dsn1 .                                        /* get arg (filename)*/
if dsn1 ¬= "" then do                             /* get its volume    */
dsn11 = dsn1
xx = listdsi(dsn1)
vol11 = sysvolume
end
say"MoveFile: Input File? ( ENTER for" dsn1
pull dsn1 .
if dsn1 = "" then dsn1 = dsn11
say" Input Volume? ( ENTER for" vol11
pull vol1 .
if vol1 = "" then vol1 = vol11
say" Output Volume? ( ENTER for" vol22
pull vol2 .
if vol2 = "" then vol2 = vol22
dropbuf
dsn1 = strip(dsn1,",""")
queue "/"userid()"Ø JOB MSGCLASS=X,MSGLEVEL=(1,1)"
queue "/"STEP1 EXEC PGM=ADDRSSU,REGION=2M"
queue "/"SYSPRINT DD SYSOUT=*
queue "/"SYSIN DD *
queue " COPY DS(INCLUDE("dsn1")) -"
queue " INDYNAM ("vol1") -"
queue " OUTDYNAM ("vol2") -"
queue " CATALOG -"
queue " DELETE -"
queue " FORCE -"
queue " TGTALLOC (TRK) -"
queue " PROCESS (SYS1)"
queue "/*"
queue ""
"execio * diskw ddtemp (finis"
"submit '"jobfile'"
saida:
"free da('"jobfile")"
"free dd(ddtemp)"
exit

Luis Paulo Figueiredo Sousa Ribeiro
Systems Programmer
(Portugal)
Contributing to *VSAM Update*

In addition to *VSAM Update*, the Xephon family of Update publications now includes *CICS Update*, *VM Update*, *MVS Update*, *TCP/SNA Update*, *VSE Update*, *DB2 Update*, *RACF Update*, *AIX Update*, *Domino Update*, *NT Update*, *Oracle Update*, and *Web Update*. Although the articles published are of a very high standard, the vast majority are not written by professional writers, and we rely heavily on our readers themselves taking the time and trouble to share their experiences with others. Many have discovered that writing an article is not the daunting task that it might appear to be at first glance.

They have found that the effort needed to pass on valuable information to others is more than offset by our generous terms and conditions and the recognition they gain from their fellow professionals. Often, just a few hundred words are sufficient to describe a problem and the steps taken to solve it.

If you have ever experienced any difficulties with VSAM or made an interesting discovery, you could receive a cash payment, a free subscription to any of our *Updates*, or a credit against any of Xephon’s wide range of products and services, simply by telling us all about it. For a copy of our *Notes for Contributors*, which explains the terms and conditions under which we publish articles, please write to the editor, Fiona Hewitt, at any of the addresses shown on page 2, or e-mail her on 100336.1412@compuserve.com
Platinum Technology has begun shipping TransCentury File Age, its rules-based, data-ageing software designed to take advantage of the impact analysis efforts of Y2K teams and speed up the data testing process.

For more information, contact:
Platinum Technology, 1815 S Meyers Road, Oakbrook Terrace, IL 60181-5241, USA.
Telephone: (714) 453 4000.
Platinum Technology, Turnberry House, 30 Caldecote Lake Drive, Milton Keynes, Bucks, MK7 8LE, UK.
Telephone: (01908) 274777.

***

XDB Systems, recently acquired by Micro Focus, has announced Version 2.0 of its ExpressLane data access middleware, providing connectivity between PC-based graphical environments and mainframe databases including DB2 for MVS/ESA, IMS, and VSAM.

For more information, contact:
Micro Focus, 2465 E Bayshore Rd, Palo Alto, CA 94303, USA.
Tel: (415) 856 4161.
Micro Focus, Speen Court, 7 Oxford Road, Newbury, Berks, RG14 1PB.
Tel: (01635) 32646.

***

VMark Software has announced Release 3.0 of its DataStage data extraction and transformation tool. Features include change data capture, mainframe data access, and a new set of developer productivity tools.

For more information, contact:
VMark Software, 50 Washington Street, Westboro, MA 01581-1021, USA.
Tel: (508) 366 3888.
VMark Software, Edenfield, London Road, Bracknell, Berks, RG12 2XH, UK.
Tel: (01344) 355500.

***

Data mart specialist Informatica has launched PowerCenter 1.0, which allows data marts to be networked together into a virtual warehouse, and then managed from a single point.

For more information, contact:
Informatica Corp, 1200 Chrysler Drive, Menlo Park, CA 94025, USA.
Tel: (415) 462 8900.

***

Haht Software and Neon Systems plan to integrate Hahtsite e-business tools and Neon’s Shadow Direct, which accesses legacy mainframe data and business logic, in an alliance to sell more software that Web-enables legacy systems.

For more information, contact:
Neon Systems Inc, 14141 Southwest Freeway, Suite 6200, SugarLand, TX 77478, USA.
Tel: (713) 491 4200/(800) 505 NEON.
Neon Systems UK Ltd, Third Floor, Sovereign House, 26-30 London Road, Twickenham, Middx, TW1 3RW, UK.
Tel: (0181) 607 9911.