190

CICS update

September 2001

In this issue

CICS Update

Published by

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

North American office

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

Subscriptions and back-issues

A year's subscription to *CICS Update*, comprising twelve monthly issues, costs £175.00 in the UK; $270.00 in the USA and Canada; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1994 issue, are available separately to subscribers for £16.00 ($23.50) each including postage.

CICS Update on-line

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

Editor

Trevor Eddolls

Disclaimer

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

Contributions

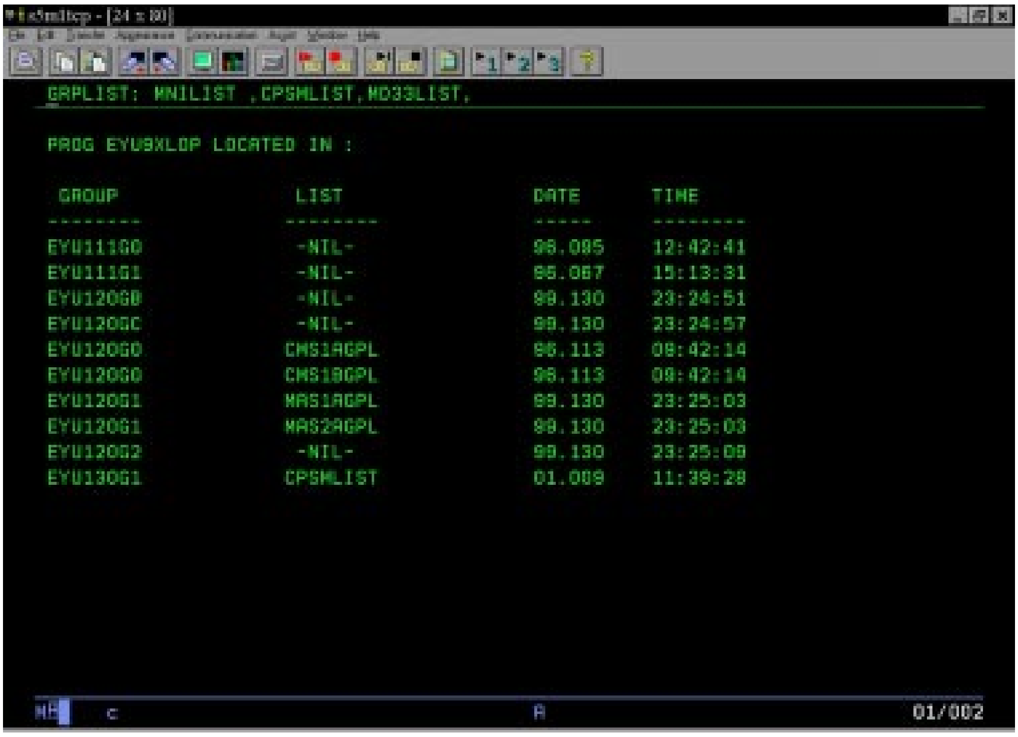
When Xephon is given copyright, articles published in *CICS Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a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

Printed in England.

Minimizing resources search time through CICS CSD

In a development environment, sometimes it is necessary to have duplicate resources in different CSD groups and lists for testing purpose. To maintain such a system, systems programmers have to issue CEDA commands to find where a resource is located in which groups, then more CEDA commands are used to find out whether the group is listed in the CSD list, then one may want to find out whether the list is on the default GRPLIST in the SIT tables.

The following program will minimize the search time for finding the location of a resource within the CSD group and list. It will display the GRPLIST order and then the resource with the group and list where it is located in the CICS CSD (see Figure 1). The search process is not a sequential search through all the entries in the CICS CSD, but only



```
   GRPLIST: MNILIST ,CPSMLIST,MD33LIST,

   PROG EYU9XLOP LOCATED IN :

    GROUP             LIST               DATE      TIME
   --------         --------             -----     --------
   EYU111G0          -NIL-               98.085    12:42:41
   EYU111G1          -NIL-               98.087    15:13:31
   EYU120GB          -NIL-               99.130    23:24:51
   EYU120GC          -NIL-               99.130    23:24:57
   EYU120G0         CMS1AGPL             98.113    08:42:14
   EYU120G0         CMS1BGPL             98.113    08:42:14
   EYU120G1         MAS1AGPL             99.130    23:25:03
   EYU120G1         MAS2AGPL             99.130    23:25:03
   EYU120G2          -NIL-               99.130    23:25:08
   EYU130G1         CPSMLIST             01.009    11:39:28









  MB    c                                A                             01/002
```

Figure 1: Example GRPLIST

a search of the group entries, and then it will do a direct read into the group by filling in the VSAM record key. The record key of a CSD VSAM file comprises four record fields – group name, sequence number, resource type, and resource name.

The program browses through the group entries using the resource type field for group entry, which is always X'0006', then performs a direct read into the group entry by filling in the resource name and the corresponding resource type hex value. If the direct read is successful, it will search for the CSD list entry, otherwise it will skip through the rest of the resources in the group entry by filling in high values in the sequence number field, which will effectively bring us to the next group entry. When searching for a list entry, the same concepts apply, except that we cannot predict the sequence number field (for group entry the sequence number field is always a zero value), so we have to search all the list entries.

For example, if you want to investigate program XYZ, just issue the transaction ID, followed by PROG(XYZ). Compare this with the conventional way of issuing CEDA EXPAND GROUP(*) PROG(XYZ), and then CEDA EXPAND LIST(*) GROUP(XYZGROUP). If XYZGROUP is the only group returned from the previous command, the next step probably would be looking up the GRPLIST parameter in the SIT table.

This program was written to maximize the productivity of systems programmers. It can also help you to clean up the CSD file by listing obsolete CSD groups and lists. The resource types supported are for CICS Version 4.1.0, but it can easily include other resource types found in CICS TS. This program needs to be compiled with translation option SP, and macro library CICS.SDFHMAC.

CSDPGM

```
         TITLE 'CSDPGM - FIND GROUPS/LISTS OF CSD RESOURCE'
* AUTHOR : Kah Soon HO
         PRINT  NOGEN
         EJECT
DFHEISTG DSECT ,
         DFHCSAD TYPE=DSECT
         DFHAFCD TYPE=DSECT
         DFHSIT  TYPE=DSECT
CSDPGM   CSECT
CSDPGM   AMODE 31
CSDPGM   RMODE ANY
         DFHREGS ,                 EQUATE REGISTERS
         MVC   RSCNAME,=CL8'        '
         EXEC  CICS RECEIVE INTO(INPUT) MAXLENGTH(8Ø) LENGTH(TEXTLEN)
         CLC   STYPE,=CL4'PROG'         MATCH TYPE OF RESOURCE
         BE    PROG
         CLC   STYPE,=CL4'TRAN'
         BE    TRAN
         CLC   STYPE,=CL4'TERM'
         BE    TERM
         CLC   STYPE,=CL4'TYPE'
         BE    TYPE
         CLC   STYPE,=CL4'SESS'
         BE    SESS
         CLC   STYPE,=CL4'PROF'
         BE    PROF
         CLC   STYPE,=CL4'MAPS'
         BE    MAPS
         CLC   STYPE,=CL4'FILE'
         BE    FILE
         CLC   STYPE,=CL4'LSRP'
         BE    LSRP
         CLC   STYPE,=CL4'CONN'
         BE    CONN
SYNTAX   MVC   DISPLAY,ERROR1      ELSE IS UNSUPPORTED TYPE
         BAL   R1,SENDMSG
         MVC   DISPLAY,ERROR2
         BAL   R1,SENDMSG
         MVC   DISPLAY,ERROR3
         BAL   R1,SENDMSG
         B     RETURNY             SEND MSG AND RETURN
FINDNAME DS ØH                     GET RESOURCE NAME
         LH    R8,TEXTLEN          LENGTH RECEIVE
         LA    R9,SDATA            DATA RECEIVE
         LA    R1Ø,RSCNAME         RESOURCE NAME
RNAMELP  DS ØH
         CLI   Ø(R9),X'4D'         IS IT ( ?
         BE    SPACELP             SKIP SPACE
         LA    R9,1(,R9)           POINT AT NEXT CHAR
         BCT   R8,RNAMELP          CHECK NEXT CHAR
SPACELP  LA    R9,1(,R9)           CHECK FOR SPACE
SPACELP2 CLI   Ø(R9),X'4Ø'         IS IT SPACE ?
         BNE   RNAMEL              GET RESOURCE NAME
         LA    R9,1(,R9)           POINT AT NEXT CHAR
         BCT   R8,SPACELP2         CHECK NEXT CHAR
         LA    R1Ø,RSCNAME         RESOURCE NAME
```

```
RNAMEL   CLI   Ø(R9),X'5D'         CHECK FOR END LOOP
         BE    START
         CLI   Ø(R9),X'4Ø'
         BE    RCOUNT
         MVC   Ø(1,R1Ø),Ø(R9)      GET RESOURCE NAME
         LA    R1Ø,1(R1Ø)
RCOUNT   LA    R9,1(R9)
         BCT   R8,RNAMEL
         B     SYNTAX
START    DS    ØH
         DFHAFCD TYPE=LOCATE       POINT TO AFCB
         USING DFHAFCB,R15         USE SIT DSECT
         L     R15,AFCSA           POINT TO CSA
         USING DFHCSADS,R15        USE CSA DSECT
         L     R14,CSASITBA        LOAD SIT ADDRESS FROM CSA
         DROP  R15
         USING DFHSITDS,R14        USE SIT DSECT
         MVC   DISPLAY,=CL79' '     BUILD MASSAGE
         MVC   DISPLAY(9),=CL9'GRPLIST:'
         MVC   DISPLAY+9(8),SITGRPLI
         MVC   DISPLAY+17(1),=CL1','
         MVC   DISPLAY+18(8),SITGRPL2
         MVC   DISPLAY+26(1),=CL1','
         MVC   DISPLAY+27(8),SITGRPL3
         MVC   DISPLAY+35(1),=CL1','
         MVC   DISPLAY+36(8),SITGRPL4
         MVC   DISPLAY+77(2),=XL2'1515'
         BAL   R1,SENDMSG
         MVC   DISPLAY,=CL79' '
         MVC   DISPLAY(4),STYPE
         MVC   DISPLAY+5(8),RSCNAME
         MVC   DISPLAY+14(12),=CL12'LOCATED IN :'
         MVC   DISPLAY+77(2),=XL2'1515'
         BAL   R1,SENDMSG
         MVC   DISPLAY,=CL79' '
         MVC   DISPLAY(6),=CL6' GROUP'
         MVC   DISPLAY+2Ø(5),=CL5' LIST'
         MVC   DISPLAY+4Ø(5),=CL5' DATE'
         MVC   DISPLAY+5Ø(5),=CL5' TIME'
         BAL   R1,SENDMSG
         MVC   DISPLAY(8),=CL8'--------'
         MVC   DISPLAY+2Ø(8),=CL8'--------'
         MVC   DISPLAY+41(5),=CL5'-----'
         MVC   DISPLAY+51(8),=CL8'--------'
         BAL   R1,SENDMSG
         MVC   RIDF,=XL22'Ø'       START AT FIRST RECORD
         EXEC CICS SET FILE('DFHCSD') ENABLED OPEN
STRTBR   EXEC CICS STARTBR FILE('DFHCSD') RIDFLD(RIDF) RESP(RESPONSE)  X
           REQID(1) KEYLENGTH(18) GENERIC GTEQ
```

```
FINDGRP  DS    ØH
         EXEC CICS READNEXT FILE('DFHCSD') INTO(CSDREC) RIDFLD(RIDF)   X
           RESP(RESPONSE) REQID(1) KEYLENGTH(18)
         CLC   RESPONSE,DFHRESP(ENDFILE)    END OF FILE?
         BE    ENDFILE
         CLC   RESPONSE,DFHRESP(NOTFND)     DOES THE RECORD EXIST?
         BE    NOTFOUND
         CLC   RESPONSE,DFHRESP(NORMAL)     UNEXPECTED ERROR?
         BNE   ERRORS
         CLC   RIDTYPE,=XL2'ØØØ6'           IS IT GROUP
         BNE   FINDGRP                      FIND NEXT GROUP
         MVC   RIDTYPE,HTYPE                FILL IN RECORD ID FOR -
         MVC   RIDNAME,RSCNAME               DIRECT READ
         EXEC CICS READ FILE('DFHCSD') INTO(CSDREC)  RIDFLD(RIDF)      X
           KEYLENGTH(22) EQUAL RESP(RESPONSE)
         CLC   RESPONSE,DFHRESP(NOTFND)     DOES THE RECORD EXIST?
         MVC   RIDSEQNO,=XL4'FFFFFFFF'      SKIP THE REST OF ENTRIES -
         BE    FINDGRP                       WITH THE GROUP
         CLC   RESPONSE,DFHRESP(NORMAL)     UNEXPECTED ERROR?
         BNE   ERRORS
         MVC   DATE,DATADAY           GROUP FOUND,GET DATE PORTION
         MVC   TIME,DATATIME          TIME PORTION
         L     2,TIME
         SLL   2,4                    REMOVE HIGH 4 BITS
         ST    2,TIME
         BAL   R2,FINDLIST            FIND THE LIST FOR THE GROUP
         B     FINDGRP                FIND THE REST OF THE GROUP
         DFHEJECT
FINDLIST DS    ØH
         ST    R2,R2SAVE             STORE RETURN ADDRESS
         MVC   LFLAG,=CL1'N'
         MVC   LID,=XL22'Ø'           START AT FIRST RECORD
         EXEC CICS READNEXT FILE('DFHCSD') INTO(CSDREC)  RIDFLD(LID)   X
           RESP(RESPONSE) REQID(1) KEYLENGTH(14)
FINDL    DS    ØH
         MVC   LIDTYPE,=XL2'ØØØD'     FILL IN RECORD TYPE
         MVC   LIDNAME,=CL8'        '
         EXEC CICS READNEXT FILE('DFHCSD') INTO(CSDREC)  RIDFLD(LID)   X
           RESP(RESPONSE) REQID(1) KEYLENGTH(14)
         CLC   RESPONSE,DFHRESP(ENDFILE) END OF FILE?
         BE    ENDFILEL
         CLC   RESPONSE,DFHRESP(NOTFND)  DOES THE RECORD EXIST?
         BE    ENDFILEL
         CLC   RESPONSE,DFHRESP(NORMAL)  UNEXPECTED ERROR?
         BNE   ENDFILEL
         CLC   LIDTYPE,=XL2'ØØØD'     IS IT A LIST?
         BNE   FINDL2                 READ NEXT
COMPLIST CLC   LIDNAME,RIDGROUP       IS GROUP ENTRY ON THE LIST?
         BNE   FINDL                  FIND NEXT
         MVC   LFLAG,=CL1'Y'
BUILDMSG MVC   OUTDATE,=XL7'4Ø2Ø2Ø4B2Ø2Ø2Ø'
         ED    OUTDATE,DATE           EDIT DATE
         OI    OUTDATE+1,X'FØ'        ENSURE PRINTABILITY
         OI    OUTDATE+2,X'FØ'
         OI    OUTDATE+3,X'4B'
         MVC   OUTTIME,=XL9'4Ø2Ø2Ø7A2Ø2Ø7A2Ø2Ø'
         ED    OUTTIME,TIME           EDIT TIME
         OI    OUTTIME+1,X'FØ'        ENSURE PRINTABILITY
         OI    OUTTIME+2,X'FØ'
         MVC   DISPLAY(8),RIDGROUP    BUILD MSG
         MVC   DISPLAY+2Ø(8),LIDGROUP
         MVC   DISPLAY+4Ø(7),OUTDATE
         MVC   DISPLAY+5Ø(9),OUTTIME
         BAL   R1,SENDMSG
         CLC   LFLAG,=CL1'E'
         BNE   SKIPREST               SKIP THE REST OF ENTRIES IN GROUP
         L     R2,R2SAVE              GET RETURN ADDRESS
         BR    R2                     RETURN
FINDL2   CLC   LIDTYPE,=XL2'ØØØ6'     IS IT A GROUP?
         BNE   SKIPREST               SKIP THE REST OF ENTRIES IN GROUP
         MVC   LIDNAME,=CL8'        ' CLEAR
         EXEC CICS READNEXT FILE('DFHCSD') INTO(CSDREC)  RIDFLD(LID)   X
           RESP(RESPONSE) REQID(1) KEYLENGTH(14)
         CLC LIDTYPE,=XL2'ØØØD'       IS IT A LIST?
         BE  COMPLIST                 CHECK IF GROUP FOUND IN LIST?
SKIPREST DS  ØH
         MVC LIDSEQNO,=XL4'FFFFFFFF'  SKIP THE REST OF RECORDS
         B   FINDL                    FIND NEXT LIST
*
TRAN     DS ØH
         MVC HTYPE,=XL2'1388'
         B FINDNAME
TERM     DS ØH
         MVC HTYPE,=XL2'123A'
         B FINDNAME
TYPE     DS ØH
         MVC HTYPE,=XL2'11CB'
         B FINDNAME
SESS     DS ØH
         MVC HTYPE,=XL2'1ØED'
         B FINDNAME
PROG     DS ØH
         MVC HTYPE,=XL2'ØFAØ'
         B FINDNAME
PROF     DS ØH
         MVC HTYPE,=XL2'ØBB8'
         B FINDNAME
MAPS     DS ØH
```

```
         MVC HTYPE,=XL2'Ø3E8'
         B FINDNAME
FILE     DS ØH
         MVC HTYPE,=XL2'Ø32Ø'
         B FINDNAME
LSRP     DS ØH
         MVC HTYPE,=XL2'Ø28A'
         B FINDNAME
CONN     DS ØH
         MVC HTYPE,=XL2'Ø1F4'
         B FINDNAME
*
ENDFILEL DS   ØH
         CLC  LFLAG,=CL1'N'
         BNE  ENDFILER
         MVC  LFLAG,=CL1'E'
         MVC  LIDGROUP,=CL8' -NIL-  '
         B    BUILDMSG
ENDFILER L    R2,R2SAVE
         BR   R2
ENDFILE  DS   ØH
         EXEC CICS ENDBR  FILE('DFHCSD') REQID(1)
*        MVC  MSGØ2,=CL8'ENDFILE'
         B    RETURNX
NOTFOUND DS   ØH
*        MVC  MSGØ2,=CL8'NOTFOUND'
         B    RETURNX
ERRORS   DS   ØH
*        MVC  MSGØ2,=CL8'ERROR'
         B    RETURNX
SENDMSG  ST   R1,R1SAVE                    STORE RETURN ADDRESS
         EXEC CICS SEND TEXT FROM(DISPLAY) LENGTH(79) FREEKB ERASE     X
           ACCUM PAGING
         L    R1,R1SAVE
         BR   R1 RETURN
RETURNX  DS   ØH
         EXEC CICS SET FILE('DFHCSD') CLOSED
*        EXEC CICS SEND TEXT FROM(MSG) LENGTH(8Ø) FREEKB ERASE        X
*          ACCUM PAGING
RETURNY  EXEC CICS SEND PAGE
RETURN   EXEC CICS RETURN
*
*  DATA DEFINITIONS
*
RIDF      DS   ØF                    USE FOR GROUP KEY
RIDGROUP  DS   CL8                   GROUP NAME
RIDSEQNO  DS   F                     RECORD SEQUENCE NUMBER
RIDTYPE   DS   H                     ENTRY TYPE
RIDNAME   DS   CL8                   RESOURCE NAME
```

```
*
LID       DS   ØF                     USE FOR LIST KEY
LIDGROUP  DS   CL8                    GROUP NAME
LIDSEQNO  DS   F                      RECORD SEQUENCE NUMBER
LIDTYPE   DS   H                      ENTRY TYPE
LIDNAME   DS   CL8                    RESOURCE NAME
*
CSDREC    DS   ØCL522                 RECORD RETURN
CSDGROUP  DS   CL8                    GROUP NAME
CSDSEQNO  DS   F                      RECORD SEQUENCE NUMBER
CSDTYPE   DS   H                      ENTRY TYPE
CSDNAME   DS   CL8                    RESOURCE NAME
DATADAY   DS   CL6
DATATIME EQU   DATADAY+2,4
          DS   CL494                  REMAINING PORTION OF RECORD
*
R1SAVE    DS   F                      RETURN ADDRESS
R2SAVE    DS   F                      RETURN ADDRESS
RSCNAME   DS   CL8' '                 RESOURCE NAME
HTYPE     DS   XL2
RESPONSE  DS   F                      RESPONSES TO CICS COMMANDS
LFLAG     DS   CL1'N'                 LIST FLAG
*
TEXTLEN   DS   H                      DATA RECEIVE LENGTH
INPUT     DS   ØCL8Ø                  DATA RECEIVE
          DS   CL5
STYPE     DS   CL4' '                 RESOURCE TYPE
SDATA     DS   CL71
*
DISPLAY  DS  CL8Ø' '
ERROR1   DC  CL8Ø'SYNTAX OF COMMAND: <TRANID> <TYPE> (NAME)'
ERROR2   DC  CL8Ø' <TYPE>: CONNection, LSRPool, FILE, MAPSet, PROFile,'
ERROR3   DC  CL8Ø' PROGram, SESSions, TYPEterm, TERMinal, TRANSaction'
*
DATE     DC  F'Ø'
TIME     DC  2F'Ø'
OUTDATE  DS  CL7
OUTTIME  DS  CL9
MSG      DS  ØCL79
MSGØ1    DC  CL7'STATUS:'
MSGØ2    DS  CL73' '
         END CSDPGM
```

Kah Soon Ho
Senior Systems Support Analyst
Public Bank (Malaysia)

Changing CEDA defaults

Below is an update to *Changing CEDA defaults*, published in *CICS Update* Issue 104, July 1994.This is the CICS 5.3. usermod to provide the same functions as described in the article.

```
++USERMOD(LT53001).
++VER(C150) FMID(HCI5300)
 /*
 *********************************************************************
 * LT53001  - Usermod to force CEDA Define to Userdefine             *
 *********************************************************************
 */.
++HOLD(LT53001) FMID(HCI5300) SYSTEM REASON(ACTION) DATE(01001)
  COMMENT(

         After LT53001 has been APPLY'd it will be impossible
         for any resources to be CEDA DEFINE'd. CEDA USERDEF is forced.
         As a result, a group named USERDEF must be created in each
         DFHCSD. This group must contain a resource for each type
         named USER.
         The definitions in the group USERDEF will be used as the
         default settings when a user defines a new resource.

         ==============================================================

         Action when IBM PTF does not PRE or SUP LT53001
         -----------------------------------------------

         If an IBM PTF will not APPLY because it does not PRE or
         SUP LT53001 then;

         o RESTORE and REJECT this USERMOD.
         o APPLY the IBM PTF.
         o Alter the source for this USERMOD so that it has the
           IBM PTF as a PRE-req. Review changes caused by the IBM
           PTF and alter this USERMOD in accordance.
         o RECIEVE and APPLY this USERMOD.

  ).
++ZAP (DFHESP19).
NAME    ANALYZE
VER     000CCE 58A0,C0A0
VER     000CD2 D50B,A000,80BB
VER     000CE0 47F0,3D0C
```

```
REP      ØØØCCE 582Ø,CØAØ
REP      ØØØCD2 D5ØB,2ØØØ,8ØBB
REP      000CE0 47F0,3C38
```

J P Lemmon
Lemon-tree (UK)

Health check-up for the CICS subsystem

There are numerous CICS applications successfully executing around the globe. A periodic ‘health check’ of these online applications can help identify performance problems and you can perform pro-active maintenance before these start posing serious problems for the application or use more than the required amount of system resources to get the work done.

A few things listed here can help you do a quick check-up of your CICS system. This can also help you identify both problems that already exist and potential problems.

CICS shutdown statistics form the basis for our action. There is a wealth of information available in the shutdown statistics that can be harvested to analyse and ensure optimal performance of the CICS subsystem.

Shutdown statistics can be gathered using the IBM-supplied CICS utility, DFHSTUP, which uses the CICS SMF record (SMF 110) to analyse the information and report it. For a complete description of this utility refer to the *CICS Operations and Utility Guide*.

This article does not discuss database-related issues (databases other than VSAM files) as a part of the CICS system health check because these form a separate topic by themselves.

FILE REQUEST INFORMATION

As we all know, no I/O is the best I/O. In this section of the CICS shutdown statistics, a key figure to watch out for is the number of EXCP requests on the file. All files with high EXCP counts are likely to be performance bottlenecks. The top 20-30 percent of the files with the highest EXCP count should be carefully examined for the type of access requests (get/browse/update/add/delete). Some files are used just for add requests. In many cases, it so happens that these are defined as VSAM KSDS clusters, which have an additional overhead of maintaining an index. Not only that, if the records being added are not written in ascending key sequence then excessive I/Os result because of CI and CA splits. For better performance these should be allocated as VSAM ESDS clusters using NSR access and ‘number of strings=1’.

Files showing a high number of CI and CA splits should be analysed to arrive at an initial loading strategy to minimize the splits and/or alter the CI size and FREESPACE parameters. Although shutdown statistics do not provide the CI/CA split information, LISTCAT can be used to gather this information. Information should also be gathered for unused alternate indexes. If the path shows a high EXCP count but no get/browse requests, it is likely to be redundant. At one of the installations where I worked, I found an alternate index was being built but was not used because of a change in functionality. Such alternate indexes should be reviewed with developers to find out whether they are needed by the application or not. If not, they should be removed.

Small and heavily read files are good candidates for data tables.

Waits on strings/buffers on critical files can also lead to serious performance problems and appropriate buffers/strings should be increased to minimize the waits.

LSRPOOLS AND DATA TABLES

Files with a high read to write ratio with fairly random access are good candidates for LSRPOOLS. Files with a low read to write ratio should not be allocated to LSRPOOLS. LSR buffers for these files would have to be externalized and therefore these would not show any major improvement by using LSRPOOL buffers. Using the information from the file request statistics above, those files to be placed in the

LSRPOOL can be identified and the LSRPOOL can be allocated appropriately for the files. The LSRPOOL look-aside ratio is the key thing to watch out for. It is calculated using following formula:

```
    Look-aside ratio = Look asides /(Look-asides + Buffer reads)
```

The closer to 1.00 the better it is. The look-aside ratio forms the basis for tuning the LSRPOOLS. The look-aside ratio goes up if the buffer size is increased. Of course as real storage is a constrained resource, there is an upper limit to the number of buffers that can be assigned to the LSRPOOL. Because the look-aside ratio is no exception to the law of diminishing returns, adding more pools beyond a certain point is not fruitful. Separate buffers for data and indexes are recommended for large files with semi-random access so that the data control intervals do not monopolize the LSR buffers by discarding the index control intervals. Separate buffers would reduce I/O for a frequently used index CI. Alternatively, compound buffers can be used by standardizing the CI sizes of data and index components of the VSAM cluster. Small and heavily read files can be defined as data tables because the path length to read a record from a data table is smaller than that from an LSRPOOL.

TRANSACTION MANAGER

Take a look at the transaction manager statistics. Things are not in good shape if we see that the MAXTASKS limit is being reached frequently. Transactions would queue up when the MAXTASK limit is reached. The gravity of the problem depends on how many times the MAXTASK limit is hit. Ideally this should be 0, but for some applications an occasional occurrence of this may not be a major concern. However, if this occurs in a CICS region during non-peak activity periods then it certainly needs attention. The remedy would be to increase MAXTASK appropriately in the SIT.

STORAGE MANAGER

Storage manager statistics show us the current limit and maximum usage of the DSA and the EDSA. The appropriate DSA limit should be increased if the peak utilization is approaching the maximum limit.

Not doing so could result in CICS releasing storage cushions, which means spending CPU cycles on non-application work, impacting performance. Time cushions released show up in the storage manager statistics and should ideally be 0, however having this at 0 is not sufficient. A healthy system would never run with a nearly 100 percent-utilized DSA or the EDSA. What if you implement new functionality, which adds a few more programs and a couple of additional files (which means more access control blocks) resulting in increased utilization of the EDSA? The CICS region may now go short on storage. It would therefore be nice to ensure that the above situation does not arise by having sufficient buffers for the DSA/ EDSA, and monitoring it after every implementation.

TEMPORARY STORAGE AND TRANSIENT DATA QUEUES

Temporary storage is mainly used by applications as a scratch pad area, but at many installations one sees all the TSQs allocated to auxiliary storage. If the statistics indicate a zero or very low utilization of main storage then there is a potential for performance improvement by moving the non-recoverable TSQs to MAIN storage, thereby reducing I/Os to an auxiliary dataset. If the statistics show too many queue extensions, it means that CICS is spending its resources doing ‘non-application’ work like GETMAIN and FREEMAIN. Should this happen at your installation, SIT parameter TSMGSET should be appropriately increased to reduce queue extensions. Watch out for buffer or string waits on the queues, if any. These waits can be eliminated/minimized by increasing the number of buffers/strings allocated.

JOURNALS

Important statistics to look out for are the buffer full condition and waits on archive. If the buffer full condition occurs, the buffer size should be increased in the journal control table. If waits on archive are experienced, consider increasing the log size or reducing the archival frequency; verify that, if the logs are archived to tape, tape mounts are performed quickly. Also have a look at the service class or dispatching priority (depending on whether the system is running in WLM goal mode or ICS/IPS configuration) of the journal job. If this is very low, the job may not get the required resources during busy hours. Care should also be taken not to make it too high or CICS response may suffer when the journal job executes.

REDUNDANT PROGRAMS/TRANSACTIONS/TERMINALS

Some programs/transactions become redundant over a period of time. Examining the 'attach count' can identify these (which would be 0 for unused transactions) in the TRANSACTION STATISTICS, and check the 'times used' count in the PROGRAM details. If too many of these are around, then they should be removed from the CICS system tables and CSD because these could result in increased 'non-application' resource usage in terms of real storage to hold these entries and CPU cycles to search table entries.

Pranav Sampat
Cognizant Technology Solutions (USA)

Determining the library using PINQPGM – revisited

I worked with some people who had a problem after migrating to OS/390 Version 2.8+ with the program PINQPGM published in *Determining the library using PINQPGM*, published in *CICS Update*, Issue 152, July 1998. Below is an updated program.

PINQPGM

```
//TRN       EXEC PGM=DFHEAP1$,
//          REGION=4Ø96K
//STEPLIB   DD DSN=CICS41Ø.SDFHLOAD,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSPUNCH  DD DSN=&&SYSCIN,
//          DISP=(,PASS),UNIT=SYSALLDA,
//          DCB=BLKSIZE=4ØØ,
//          SPACE=(4ØØ,(4ØØ,1ØØ))
//SYSIN     DD *
```

```
******************************************************************
*   WRITTEN BY CHORNG S. (JACK) HWANG  199Ø @ SDG&E              *
*     RETROFITTED FOR CICS V4.1        1997 @ HARRIS BANK        *
*   WRITTEN AND DISTRIBUTED AS IS, NO WARRANTIES EITHER          *
*     EXPRESSED OR IMPLIED                                       *
*                        JACK HWANG    CSHWANG@HOTMAIL.COM       *
*   UPDATE/RETROFIT FOR OS/39Ø V2.8+                             *
*          BY JOE BARNES   2/8/2ØØ1    JOE_BARNES@SECURA.NET     *
*             TOM N THOMAS 5/14/2ØØ1   THOMAST@ATTGLOBAL.NET     *
******************************************************************
          PRINT NOGEN
          TITLE 'PINQPGM - FIND DFHRPL FOR PROGRAM'
NEWLINE   EQU   X'15'
STFIELD   EQU   X'1D'
          COPY  DFHAID
          COPY  DFHBMSCA
          DCBD  DSORG=PO,DEVD=DA
          IEFTIOT1
          IEZDEB
          IHAPSA
          IKJTCB
*
DFHEISTG  DSECT
HEADERA   DS    CL5
HEADERT   DS    CL8
          DS    CL24
HEADERC   DS    CL6
HEADERS   DS    CL4
          DS    CL24
HEADERD   DS    CL8
HEADERNL  DS    CL2
PGMNAMCA  DS    CL5
PGMNAMC   DS    CL8'PGMNAME:'
PGMNAMA   DS    CL2
PGMNAM    DS    CL8
PGMNAMEA  DS    CL2
HEADERLE  EQU   *-HEADERA
CURSOR    DS    H
RECVLEN   DS    H
TEXTLEN   DS    H
TEXTPTR   DS    F
*
ABSTIME   DS    D
DSNAME    DS    CL44
CONCAT    DS    CL4
DDNAME    DS    CL8
BALSAVE   DS    F
TCBSAVE   DS    F
*
BLDLAREA  DS    CL2Ø
```

```
REGSTORE  DS    16F
MVSREGSA  DS    18F
RSTORE59  DS    5F
*
TEXTOUT   DS    CL256
*
* REGISTER USAGE TABLE
*    RØ   WORK REG
*    R1   WORK REG
*    R2   WORK REG
*    R3   BASE REG FOR CODE
*    R4   BASE REG FOR CODE
*    R5   WORK REG
*    R1Ø  BASE REG FOR RECEIVED DATA
*    R11  BASE REG FOR EIB
*    R12  BASE REG FOR WORKAREA
*    R13  MVS SAVE AREA
*
          PRINT GEN
PINQPGM  AMODE 31
PINQPGM  RMODE ANY
PINQPGM  DFHEIENT CODEREG=(3,4),DATAREG=(12)
         CLI   EIBAID,DFHCLEAR      IS THIS CLEAR?
         BE    RETURN               YES, RETURN AND END
         CLI   EIBAID,DFHPF3        PF3?
         BE    RETURN               YES, RETURN AND END
         CLI   EIBAID,DFHPF15       PF3?
         BE    RETURN               YES, RETURN AND END
         OC    PGMNAM,=CL8' '       CLEAR PGMNAM
         EXEC CICS RECEIVE SET(1Ø) LENGTH(TEXTLEN)
*
*                      NEW CODE
TRANS    EQU   *                    PREPARE TRANSLATE
         STM   5,9,RSTORE59         BE CAREFULLY, SAFE REGISTERS
         XR    5,5                  CLEAR R5
         XR    6,6                  CLEAR R6
         LH    5,TEXTLEN            LOAD SLIP
         LA    7,TABØ1
         LA    8,TRANS1
         LR    9,1Ø                 LOAD INPUT
TRANS1   EQU   *                    TRANSLATE
         IC    6,Ø(Ø,9)             CHARACTER FROM INPUT
         IC    6,Ø(6,7)             TRANSLATE CHARACTER
         STC   6,Ø(Ø,9)             RETURN TO INPUT
         LA    9,1(Ø,9)             LOAD ADR NEXT
         BCTR  5,8                  TRANSLATE NEXT OR END
         LM    5,9,RSTORE59         RELOAD REGISTERS - FINISHED
*              END OF  NEW CODE
*
         CLC   Ø(4,1Ø),EIBTRNID     IS THIS UNFORMATTED?
         BE    SENDINIT             YES, GO SEND INITIAL
         LH    2,TEXTLEN            GET LENGTH OF TEXT
         SH    2,=H'3'              SUBTRACT 3 TO BYPASS FIRST SA
         BNP   DOPGMNAM             NOT > Ø, GO DO PROCESS
         LA    1,PGMNAM             GET STARTING ADDRESS OF PGMNAM
         LA    1Ø,3(1Ø)             BUMP PAST SA
PGMNAML  DS    ØH
         MVC   Ø(1,1),Ø(1Ø)         MOVE IN PGMNAM
         LA    1,1(1)               GO TO NEXT BYTE TO MOVE TO
         LA    1Ø,1(1Ø)             GO TO NEXT BYTE TO MOVE FROM
         BCT   2,PGMNAML            GO DO NEXT BYTE
         B     DOPGMNAM             GO PROCESS
*
* PROCESS PGMNAM FOUND
DOPGMNAM DS    ØH
         MVC   TEXTOUT(DSNAMEL),DSNAMES MOVE IN SEND TEXT
*
         MVC   DDNAME,=CL8'DFHRPL' GET DFHRPL GUY FIRST
         BAL   1,PROCESSØ
         LA    1Ø,TEXTOUT         GET ADDRESS OF OUTPUT AREA
         MVC   DFHRPLO-DSNAMES(L'DFHRPLO,1Ø),DSNAME MOVE DSNAME
         MVC   CONCATDO-DSNAMES(L'CONCATDO,1Ø),CONCAT MOVE CONCAT #
*
*        MVC   DDNAME,=CL8'STEPLIB' NOW GET STEPLIB GUY
         MVC   DDNAME,=XL8'ØØØØØØØØØØØØØØØØ' NOW GET STEPLIB GUY
         BAL   1,PROCESSØ
         LA    1Ø,TEXTOUT         GET ADDRESS OF OUTPUT AREA
         MVC   STEPLIBO-DSNAMES(L'STEPLIBO,1Ø),DSNAME MOVE DSNAME
         MVC   CONCATSO-DSNAMES(L'CONCATSO,1Ø),CONCAT MOVE CONCAT #
*
         MVC   TEXTLEN,=AL2(DSNAMEL) MOVE SEND LENGTH
         B     PROCESS2           GO SEND IT
*
PROCESSØ DS    ØH
         ST    1,BALSAVE          STORE RETURN ADDRESS
         MVC   CONCAT,=CL4' '
DDNLOOP  DS    ØH
         USING PSA,Ø
         L     1,PSATOLD          GET TCB'S ADDRESS
         USING TCB,1
TCBLOOP  DS    ØH
         ST    1,TCBSAVE
         SR    2,2                CLEAR R2
         ICM   2,15,TCBDEB        GET FIRST DEB ADDRESS
         BZ    NORPL              INDICATE DFHRPL NOT FOUND
         L     5,TCBTIO           GET TIOT ADDRESS
         DROP  1
         USING DEBBASIC,2
DEBLOOP  DS    ØH
         SR    1,1                CLEAR 1
         ICM   1,7,DEBDCBB        GET DCB ADDRESS
         BZ    NEXTDEB            ZERO, GO GET NEXT DEB
         USING IHADCB,1
         LH    1Ø,DCBTIOT         GET OFFSET INTO TIOT FOR THIS ENTRY
         AR    1Ø,5               GET TRUE TIOT ENTRY
         USING TIOENTRY,1Ø
         CLC   TIOEDDNM,DDNAME    DDNAME FOUND?
         BE    PROCESS
*        MVC   CSHWTO+2Ø(8),TIOEDDNM
*        MVC   CSHWTO+3Ø(8),DDNAME
*SHWTO   WTO   'PINQPGM                                      '
*        EXEC  CICS DELAY
NEXTDEB  DS    ØH
         SR    1,1                CLEAR 1
         ICM   1,7,DEBDEBB        GET NEXT DEB ADDRESS
         BZ    NORPL              INDICATE DFHRPL NOT FOUND
         LR    2,1                GET DEB ADDRESS
         B     DEBLOOP            GO GET'EM TIGER
         DROP  1,2
*
NORPL    DS    ØH
         L     2,TCBSAVE          GET TCB'S ADDRESS
         USING TCB,2
         SR    1,1
         ICM   1,15,TCBBACK       GET NEXT TCB
         DROP  2
         BZ    TCBLOOPD           NO, CONTINUE TO PROCESS
         C     1,PSATOLD          SEE IF WE'VE HIT END
         BNE   TCBLOOP
TCBLOOPD DS    ØH
         MVC   DSNAME,=CL44'DCB NOT FOUND   '
         B     PROCESS1
*
PROCESS  DS    ØH
         STM   Ø,15,REGSTORE      STORE REGISTERS
         LA    13,MVSREGSA        GET ADDRESS OF MVS SA
*
         CLC   PGMNAM,=CL8'*RSETRPL' RESET RPL?
         BNE   NORSTRPL           NO, BYPASS CLOSE/OPEN DFHRPL
         LR    5,1                SAVE DCB ADDRESS
         CLOSE ((5))
         OPEN  ((5))
         B     SENDRRPL
*
NORSTRPL DS    ØH
         MVC   BLDLAREA(2),=H'1'  INDICATE 1 ENTRY
         MVC   BLDLAREA+2(2),=H'14' 14 BYTE ENTRY
         MVC   BLDLAREA+4(8),PGMNAM MOVE IN PROGRAM NAME
         BLDL  (1),BLDLAREA       GO DO BLDL
         LM    Ø,14,REGSTORE      STORE REGISTER
         LTR   15,15              TEST 15
         BNZ   NOMEMBER           NOT FOUND
*
         USING IHADCB,1
         LH    1Ø,DCBTIOT         GET OFFSET INTO TIOT FOR THIS ENTRY
         DROP  1
         L     1,PSATOLD
         USING TCB,1
         L     5,TCBTIO           GET TIOT ADDRESS
         AR    1Ø,5               GET TRUE TIOT ENTRY
         DROP  1
*
         SR    1,1                CLEAR 1
         ICM   1,1,BLDLAREA+15    GET CONCATENATION NUMBER
         CVD   1,ABSTIME          CONVERT TO DECIMAL
         UNPK  CONCAT+1(3),ABSTIME+6(2) UNPACK
         OI    CONCAT+3,C'Ø'      FORCE X'FØ'
         MVI   CONCAT,C'+'
DSNAMELP DS    ØH
         CH    1,=H'1'            COMPARE WITH H'1'
         BL    DSNFOUND           LOW, FOUND DSNAME
         BCTR  1,Ø                SUBTRACT COUNT BY ONE
         SR    Ø,Ø                CLEAR RØ
         IC    Ø,TIOELNGH         GET TIOE LENGTH
         AR    1Ø,Ø               BUMP UP TO NEXT TIOT ENTRY
         B     DSNAMELP
DSNFOUND DS    ØH
         SR    1,1                CLEAR 1
*        ICM   1,7,TIOEJFCB       GET JFCB TOKEN
         LA     5,EPA                  GET ADDRESS OF THE EPA
         ST     5,SWEPAPTR             INITIALIZE EPA POINTER
         USING  ZB5Ø5,5                ESTABLISH ADDRESSABILITY TO EPA
*        XC     SWAEPA,SWAEPA          INITIALIZE THE EPA
***********************************************************************
*IF THE LONGER 28-BYTE EPAL IS GENERATED (UNAUTH=YES), THE INSTRUCTION
*TO INITIALIZE THE EPA IS:
         XC     SWAEPAX,SWAEPAX
***********************************************************************
*        USING  TIOT1,1                ESTABLISH ADDRESSABILITY TO TIOT
         MVC    SWVA,TIOEJFCB          MV SVA OF JFCB INTO EPA
         SWAREQ FCODE=RL,EPA=SWEPAPTR,MF=(E,SWAPARMS),UNAUTH=YES  JFCB
         L      1,SWBLKPTR             SET THE POINTER TO THE JFCB
         USING  INFMJFCB,1             ESTABLISH ADDRESSABILITY TO JFCB
         MVC   DSNAME,Ø(1)        NO OFFSET IN INFMJFCB
         B     PROCESS1
*
NOMEMBER DS    ØH
         MVC   DSNAME,=CL44'PROGRAM NOT FOUND IN CONCATENATION'
*
PROCESS1 DS    ØH
```

```
         L     1,BALSAVE            GET RETURN ADDRESS
         BR    1                    RETURN
*
PROCESS2 DS    ØH
         EXEC CICS SEND TEXT FROM(TEXTOUT) LENGTH(TEXTLEN) ERASE
*
SENDINIT DS    ØH
         XC    TEXTLEN,TEXTLEN      CLEAR TEXT LENGTH
         LA    Ø,TEXTOUT            GET ADDRESS OF OUTPUT TEXT
         ST    Ø,TEXTPTR            STORE ADDRESS OF OUTPUT TEXT
         MVI   HEADERA,STFIELD      MOVE IN START FIELD
         MVI   HEADERA+1,DFHBMASK   MOVE IN ASKIP
         MVI   HEADERA+2,DFHSA      MOVE IN SET ATTRIBUTE
         MVI   HEADERA+3,DFHCOLOR   MOVE IN COLOR
         MVI   HEADERA+4,DFHTURQ    MOVE IN COLOR TURQUIS
         MVC   HEADERC,=CL6'SYSID=' INDICATE SYSID
         EXEC CICS ASSIGN SYSID(HEADERS)
         EXEC CICS ASKTIME ABSTIME(ABSTIME)
         EXEC CICS FORMATTIME ABSTIME(ABSTIME)                         X
               TIME(HEADERT) TIMESEP MMDDYY(HEADERD) DATESEP
         MVI   HEADERNL,NEWLINE     MOVE NEW LINE AFTER LINE1
         MVI   HEADERNL+1,NEWLINE   MOVE NEW LINE AFTER LINE1
         MVC   PGMNAMCA,HEADERA     MOVE IN DEFAULT DISPLAY ATTRIBUTE
         MVC   PGMNAMC,=CL8'PGNNAME:'
         MVI   PGMNAMA,STFIELD      MOVE IN START FIELD
         MVC   PGMNAMA+1(1),=AL1(DFHBMUNP+DFHBMFSE+DFHBMBRY)
         MVI   PGMNAMEA,STFIELD     MOVE IN START FIELD
         MVI   PGMNAMEA+1,DFHBMASK  MOVE IN ASKIP
         LH    1,TEXTLEN            GET TEXT LENGTH
         LA    1,HEADERLE(1)        ADD LENGTH OF HEADER
         STH   1,TEXTLEN            STORE TEXT LENGTH
         L     1,TEXTPTR            GET OUTPUT LOCATION
         MVC   Ø(HEADERLE,1),HEADERA MOVE OUTPUT LINE
         LA    1,HEADERLE(1)        BUMP UP MVC LENGTH
         ST    1,TEXTPTR
SENDTEXT DS    ØH
         EXEC CICS SEND TEXT FROM(TEXTOUT) LENGTH(TEXTLEN)             X
               FREEKB CURSOR(=AL2(171))
RETURNX  DS    ØH
         EXEC CICS RETURN TRANSID(EIBTRNID)
RETURN   DS    ØH
         EXEC CICS SEND CONTROL ERASE FREEKB
RETURNR  DS    ØH
         EXEC CICS RETURN
SENDRRPL DS    ØH
         LA    5,RRPLLEN
         STH   5,TEXTLEN
         EXEC CICS SEND TEXT FROM(RRPLOUT) LENGTH(TEXTLEN)             X
               ERASE FREEKB
         B     RETURNR
```

```
*
DSNAMES  DC     XL6'151515151515'
         DC    AL1(STFIELD,DFHBMASK,DFHSA,DFHCOLOR,DFHTURQ)
         DC    C' DFHRPL: '
DFHRPLO  DS    CL44'THIS IS SUPPOSED TO BE THE DATASET NAME'
         DC    C'   CONCAT: '
CONCATDO DS    CL4
         DC    XL6'1515'
         DC    AL1(STFIELD,DFHBMASK,DFHSA,DFHCOLOR,DFHTURQ)
         DC    C'STEPLIB: '
STEPLIBO DS    CL44
         DC    C'   CONCAT: '
CONCATSO DS    CL4
DSNAMEL  EQU   *-DSNAMES
*
*             MORE NEW STUFF
         DS    ØF
TABØ1    EQU   *                   TRANSLATE FROM UPPER TO LOWER
         DC    X'4Ø'               TRANSLATE X'ØØ' TO X'4Ø'
         DC    127AL1(*-TABØ1)
         DC    X'8ØC1C2C3C4C5C6C7C8C98A8B8C8D8E8F'   A-I
         DC    X'9ØD1D2D3D4D5D6D7D8D99A9B9C9D9E9F'   J-R
         DC    X'AØA1E2E3E4E5E6E7E8E9AAABACADAEAF'   S-Z
         DC    8ØAL1(*-TABØ1)
RRPLOUT  DC    AL1(STFIELD,DFHBMASK,DFHSA,DFHCOLOR,DFHTURQ)
         DC    C'PINQPGM - CLOSE/OPEN DFHRPL COMPLETED'
RRPLLEN  EQU   *-RRPLOUT
*
SWEPAPTR DS    F
EPA      DS    CL28
SWAPARMS SWAREQ MF=L
         CVT DSECT=YES
         IEFJESCT
         IEFZB5Ø5 LOCEPAX=YES
         PRINT NOGEN
         IEFJFCBN
         END
//ASM       EXEC PGM=IEV9Ø,
//          REGION=4Ø96K,
//          PARM='NODECK,OBJECT,XREF(SHORT)'
//SYSLIB    DD DSN=CICS41Ø.SDFHMAC,DISP=SHR
//          DD DSN=SYS1.MACLIB,DISP=SHR
//          DD DISP=SHR,DSN=SYS1.AMODGEN
//SYSUT1    DD UNIT=SYSALLDA,SPACE=(17ØØ,(4ØØ,4ØØ))
//SYSUT2    DD UNIT=SYSALLDA,SPACE=(17ØØ,(4ØØ,4ØØ))
//SYSUT3    DD UNIT=SYSALLDA,SPACE=(17ØØ,(4ØØ,4ØØ))
//SYSLIN    DD DSN=&&LOADSET,
//          UNIT=SYSALLDA,DISP=(,PASS),
//          SPACE=(4ØØ,(1ØØ,1ØØ))
//SYSPRINT  DD SYSOUT=*
```

```
//SYSIN     DD DSN=&&SYSCIN,DISP=(OLD,DELETE)
//COPYLINK  EXEC PGM=IEBGENER,COND=(7,LT,ASM)
//SYSUT1    DD DSN=CICS410.SDFHMAC(DFHEILIA),DISP=SHR
//SYSUT2    DD DSN=&&COPYLINK,DISP=(NEW,PASS),
//          DCB=(LRECL=80,BLKSIZE=400,RECFM=FB),
//          UNIT=SYSALLDA,SPACE=(400,(20,20))
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DUMMY
//LKED      EXEC PGM=IEWL,REGION=4096K,
//          PARM='LIST,XREF',COND=(7,LT,ASM)
//SYSLIB    DD DSN=CICS410.SDFHLOAD,DISP=SHR
//SYSLMOD   DD DISP=SHR,DSN=CICS410.SDFHLOAD(PINQPGM)
//SYSUT1    DD UNIT=SYSALLDA,DCB=BLKSIZE=1024,
//          SPACE=(1024,(200,20))
//SYSPRINT  DD SYSOUT=*
//SYSLIN    DD DSN=&&COPYLINK,DISP=(OLD,DELETE)
//          DD DSN=&&LOADSET,DISP=(OLD,DELETE)
//          DD DDNAME=SYSIN
//*
```

Chorng S (Jack) Hwang
Principal
HSA Systems (USA)

Monitoring system logger activity online

With the launch of CICS Transaction Server for OS/390, the MVS system logger has become the important feature for using CICS/TS successfully. Many companies have already migrated to CICS/TS, but a lot of migration work is still to be done before CICS/ESA goes out of service.

A very important task when running CICS/TS is to monitor the logger set up for CICS. Therefore IBM provided the batch program IXGRPT1, which is supplied in SYS1.SAMPLIB. A CICS systems programmer should be very familiar with this program in order to understand whether CICS's system logs, DFHLOG and DFHSHUNT, are well defined and tuned. IXGRPT1 is an excellent window into the MVS system logger. The input for IXGRPT1 are the SMF88 records.

However, during my work at IBM's CICS support group I have had a lot of contact with customers running CICS/TS who don't know about IXGRPT1. This inevitably leads to problems. A lot of people find it difficult to interpret IXGRPT1 output and to understand the figures and the critical situations.

To give an alternative to IXGRPT1 I wrote program IXGRPTC (C stands for CICS) and program IXGSMF8. Both programs run under CICS. The first program displays the local log streams used by CICS (see below):

```
         JOURNALNAME STREAMNAME                 TYPE    STATUS
         -----------------------------------------------------
         DFHJØ2      CICS.IV4A53A1.DFHJØ2       MVS     ENABLED
         DFHLGLOG    CICSUSER.IV5A53A1.DFHLGLOG MVS     ENABLED
         DFHLOG      CICS.IV4A53A1.DFHLOG       MVS     ENABLED
         DFHSHUNT    CICS.IV4A53A1.DFHSHUNT     MVS     ENABLED


NOTE: PUT THE CURSOR ON A STREAMNAME AND PRESS ENTER KEY

                                             SYSID=53A1 APPLID=IV4A53A1
PF 3 END
```

You can easily select a log stream by putting the cursor on a log stream name and pressing enter. Now the second program will be invoked to display the SMF88 interval records on screen (see below) for the previously selected log stream:

```
                                            SMF INTERVAL:     1Ø /    33
-------------- PRODUCT SECTION -----------------------------------------
MVS OPERATION SYSTEM NAME:   MCEVS4             RELEASE:     SP6.1.Ø
-------------- LOG STREAM SECTION --------------------------------------
LOG STREAM NAME:   CICS.IV4A53A1.DFHLOG    TOD-TIME: 2ØØ1/Ø5/Ø7 13:ØØ:ØØ
# WRITES INVOKED            :            13.42Ø
BYT WRITTN BY USERS IXGWRITES :        69.247.11Ø
MIN. BLOCKLEN IN SMF INTERVAL :                12Ø (INITIALIZED TO
X"7FFFFFFF" IF NO SMF ACTIVITY OCCURS WITHIN AN SMF INTERVAL.)
MAX. BLOCKLEN IN SMF INTERVAL :             1Ø.276
-------------- STRUCTURE (INTERIM STORAGE) SECTION ---------------------
--------------------- (DASD) ----------------------------
STRUCTURE NAME: LOG_DFHLOG_ØØ1
BYT WRITTN TO INTERIM STORAGE  :        71.ØØØ.32Ø  BYT WRITTN TO DASD
:          64.578.371
BYT DELETD INTERIM ST W/O DASD :         5.246.898  BYT DELETD INTERIM
ST W/DASD   :          64.Ø81.571
# DELETES W/O DASD WRITE       :               986  # DELETS W/WRITE
:              12.42Ø
# WRITES COMPLETED - TYPE 1    :        12.124 (TYPE1 = LOG STREAM
CONTENTS CAN REMAIN IN STRUCTURE. NO NEED TO MOVE DATA.)
# WRITES COMPLETED - TYPE 2    :         1.2Ø6 (TYPE2 = LOG STREAM IS
FILLING THE STRUCTURE. LOGGER STARTS OFFL. ASYNC.)
# WRITES COMPLETED - TYPE 3    :              87 (TYPE3 = SPACE USED IN
THE STRUCTURE IS CRITICAL BUT DOES NOT EXCEED 100%.)
------------- EVENTS SECTION ---------------------------------------
DASD SHFT :     136     STRC FULL :       3      OFFLOADS  :     104
(NUMBER OF SUCCESSFUL OFFLOADS)
REBLD INI :       0     STG THLD  :       0      OFFL.90%  :     244
(NO.OF SUSUCCESSFUL OFFLOADS DUE TO STRUC.REACHING 90% FULL
REBLD CMP :       0     STG FULL  :       0      IXGOFFLD  :       0
(NUMBER OF TIMES AN OFFLOAD WAS REQUESTED VIA IXGOFFLD SERV
--------------------------------------------------------------------

PF  3 RETURN     7 UP         8 DOWN       9 FIRST I.  10 MIDLE I.  11
LAST I.       (I. = SMF INTERVAL)
PF 13 DASD SHFT 14 OFFLOAD   15 STG FULL  16 STG THLD  17 STR FULL  18
OFFL.90%               23 AVERAGE   24 TOTALS
```

Because of the huge amount of data in one interval, session property screen size 27x132 is mandatory.

This method has some advantages, including:

1 Online access to SMF88 records without needing to run batch jobs.

2 All variables for a single SMF interval on a screen.

3 Every value prefixed by a description and some suffixed with an explanation.

4 Different colours for quantity variables (in green, eg number of bytes written to logger) and event variables (in red, eg structure full).

5 Exception monitoring with PF keys (PF13 - PF18); eg PF13 = show me the interval with the highest number of DASD shifts (DASD shift = allocating a new offload dataset), or PF17 = display the interval with structure full condition on maximum.

The input for the CICS programs are the same SMF88 records as used by the batch version, but the organization format is different. I use a VSAM KSDS cluster instead of ESDS datasets. The data is copied by a third program (batch part) IXGRPTB (B stands for batch) when an SMF dataset switch occurs. Only CICS logstreams are selected by program IXGRPTB. A clean-up routine deletes the records from files that are older than a specified time in days. You should use the same jobstream as provided for IXGRPT1 in the Redbook *CICS Transaction*

Server for OS/390: Version 1 Release 2 Implementation Guide (SG24-2234-00) on page 100/101. Replace the last step by program IXGRPTB. The DD statement for input is SMF88IN, for output SMF88OT.

Checklist:

1 Define the KSDS cluster with INDEXED, KEYS(36 100) – for example:

```
//DEFINE    EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE CICS53.CICS.IXGSMF88
 SET LASTCC=Ø
 SET MAXCC=Ø
 DEFINE CLUSTER(NAME(CICS53.CICS.IXGSMF88)-
          INDEXED -
          CYL(5 2)-
          SHR(3,3)-
          FREESPACE(1Ø 1Ø)-
          REUSE  -
          KEYS(36 1ØØ)-
          RECORDSIZE(276 32756) )
/*
//
```

2 Copy SMF88 records with IXGRPTB at every SMF dataset switch – for example:

```
//* UNLOAD SMF DATA SET CONTAINING CICS DATA
//SMFDUMP EXEC PGM=IFASMFDP
//INDD1    DD DISP=SHR,BUFNO=2Ø,DSN=SYS1.MAN1
//OUTDD1   DD  DSN=&&TEMP,DISP=(NEW,PASS),SPACE=(CYL,(12,5)),UNIT=SYSDA
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
   INDD(INDD1,OPTIONS(DUMP))
   OUTDD(OUTDD1,TYPE(88))
/*
//* COPIES SMF RECORDS TYPE 88 ONLY
//COPYSEL EXEC PGM=SORT,REGION=1Ø24K
//SYSOUT  DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SORTIN  DD DSN=&&TEMP,DISP=(OLD,PASS)
//SORTOUT DD DSN=&&TEMP1,DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(12,5))
//SYSIN   DD *
  OPTION COPY
  INCLUDE COND=(6,1,BI,EQ,X'58')
/*
//* SORT EQCH SMF INPUT BY TIMESTAMP AND LOGSTREAM NAME
```

```
//SORT1    EXEC PGM=SORT
//SYSOUT   DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SORTIN   DD DSN=&&TEMP1,DISP=(OLD,DELETE)
//SORTOUT  DD DSN=&&TEMP2,DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(5,1))
//SYSIN    DD *
  OPTION VLSHRT
  SORT  FIELDS=(133,8,BI,A,
                1Ø5,26,CH,A)
     INCLUDE COND=(23,2,BI,EQ,X'ØØØ1')
/*
//* EXECUTE PL/I PROGRAM IXGRPTB
//IXGRPTB  EXEC PGM=IXGRPTB,PARM='ØØ3'  /* CLEANUP INTERAL IN DAYS */
//STEPLIB  DD DISP=SHR,DSN=user.loadlib
//SYSPRINT DD SYSOUT=*
//SMF88IN  DD DISP=(OLD,PASS),DSN=&&TEMP2
//SMF88OT  DD DISP=SHR,DSN=CICS53.CICS.IXGSMF88
```

3 CICS program definition for IXGRPTC and IXGSMF8 with Language LE370.

4 CICS transaction definition for IXGC (IXGRPTC) and IXG8 (IXGSMF8).

5 CICS mapset definition for IXGMAPSM.

6 CICS file definition for file IXGSMF88 with RECORDFormat V and OPERATIONs Browse and READ.

Because CICS uses this file as read-only you can update the file from batch without problems. To get all the data before using the transaction, you should close and reopen the file in CICS. A better way is to use a file owning region and update the file via EXCI. The best way is to use SMSVSAM.

Readers who wish to discuss the material in this article further may contact me via e-mail, at ewoerner@de.ibm.com.

IXGRPTB

```
*PROCESS LANGLVL(OS,SPROG);
IXGRPTB :PROCEDURE (MVSPARMSTRING) OPTIONS(MAIN);
DCL PLIXOPT CHAR(2ØØ) VAR INIT('SYSTEM(MVS),NOEXECOPS') STATIC
                                                      EXTERNAL;
%INCLUDE SMF88STR;
%INCLUDE CEEIBMAW;
%INCLUDE CEEIBMCT;
DCL MVSPARMSTRING CHAR(1ØØ) VAR;
DCL 1 FILLØ4 UNAL BASED(ADDR(MVSPARMSTRING)),
      2 FILLØ5  CHAR(2),
      2 CLEANUP_INTERVAL PIC'999';
DCL CLEANUP_INTERVAL_BIN FIXED BIN(15);
DCL IXGR1A OPTIONS(ASSEMBLER INTER) ENTRY(CHAR(8), CHAR(24));
DCL SMF88IN FILE RECORD INPUT;
DCL SMF88OT FILE RECORD KEYED ENV(VSAM,SIS,V);
DCL WORKAREA CHAR(32756) VAR;
DCL TIMEDATE_CHAR_88 CHAR(24);
DCL 1 FILLØ2 BASED(ADDR(TIMEDATE_CHAR_88)),
      2 TODTIME CHAR(8),
      2 FILLØ3  CHAR(8),
      2 TODDATE CHAR(8);
DCL TIMEDATE_JULIAN_88 FIXED BIN(31);
DCL TIMEDATE_JULIAN_CUR FIXED BIN(31);
DCL TIMEDATE_SECONDS_CUR FLOAT DEC(16);
DCL TIMEDATE_GREGORN_CUR CHAR(17);
DCL 1 LGSEGMENT BASED(SMF88LOF),
      2 FILLØ1    CHAR(8),
      2 KEY,                   /* KEY FOR VSAM KSDS            */
        3 KEY_PART1 CHAR(26),  /* =>  SMF88LSN                 */
        3 KEY_PART2 CHAR(2),   /* =>  SMF88LFL                 */
        3 KEY_PART3 CHAR(8);   /* =>  SMF88LTD                 */
DCL SYSPRINT FILE;
DCL (CSTG,LENGTH,SUBSTR,ONCODE,POINTERADD,STRING,VERIFY,
     PLIRETC)  BUILTIN;
DCL (EOF,STRING_FOUND) BIT(1) INIT('Ø'B);
DCL (TRUE) BIT(1) STATIC INIT('1'B);
DCL (FALSE) BIT(1) STATIC INIT('Ø'B);
DCL (C,D,E,I) FIXED BIN(31) INIT(Ø);
DCL P PTR;
DCL X CHAR(4) BASED(P);  /* DSECT FOR DFHJ*, DFHL*, DFHS* */
DCL 1 FC,                          /* FEEDBACK TOKEN */
      2 MSGSEV     FIXED BIN(15),
      2 MSGNO      FIXED BIN(15),
      2 FLAGS,
        3 CASE     BIT(2),
        3 SEVERITY BIT(3),
        3 CONTROL  BIT(3),
      2 FACID      CHAR(3),        /* FACILITY ID */
      2 ISI        FIXED BIN(31);  /* INSTANCE-SPECIFIC INFORMATION */
ON ENDFILE(SMF88IN) EOF='1'B;
ON ENDFILE(SMF88OT) EOF='1'B;
ON KEY(SMF88OT)
   BEGIN;
 /*  PUT SKIP LIST('ON KEY CONDITION RAISED, ONCODE=',
                    ONCODE());        */
     IF  ONCODE = 52    /* 52 = DUPLICATE KEY */
       THEN
            D = D + 1;  /* BUMP DUPLICATE RECORD COUNTER */
 /*         PUT SKIP LIST ('DUPLICATE KEY = '||STRING(KEY));   */
```

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

```
     END;
/* **************************************************************** */
/* IF VSAM CLUSTER IS EMPTY, OPEN IN SEQUENTIAL MODE, CLOSE AND     */
/* REOPEN IN DIRECT MODE.                                           */
/* **************************************************************** */
ON UNDEFINEDFILE(SMF88OT)
   BEGIN;
     PUT SKIP LIST('ON UNDEFINEDFILE CONDITION RAISED, ONCODE=',
                    ONCODE());
     IF ONCODE = 82   /* 82 = DATA SET NEVER LOADED */
       THEN
         BEGIN;
           OPEN FILE(SMF88OT) OUTPUT SEQUENTIAL;
           KEY_PART2 = 'ØØØØ'X;
           CALL FORMAT_TOD;
           SUBSTR(WORKAREA,1,LENGTH(WORKAREA)) = REAL_RECORD;
           WRITE FILE (SMF88OT) FROM (WORKAREA)
                 KEYFROM(STRING(KEY));
           CLOSE FILE(SMF88OT);
           OPEN FILE (SMF88OT) OUTPUT DIRECT;
         END;
   END;
/* **************************************************************** */
/* GET CLEANUP INTERVAL FROM MVS PARMS (IF AVAILABLE ).             */
/* **************************************************************** */
IF VERIFY(SUBSTR(MVSPARMSTRING,1,3),'Ø123456789') = Ø /* NUMERIC? */
   THEN CLEANUP_INTERVAL_BIN = CLEANUP_INTERVAL; /* YES. */
   ELSE CLEANUP_INTERVAL_BIN = 3;    /* NO. DEFAULT IS 3 DAYS */
PUT SKIP DATA( CLEANUP_INTERVAL_BIN );
/* **************************************************************** */
/* SELECT LOGSTREAMS WITH QUALIFIER DFHL*, DFHS* AND DFHJ*          */
/* **************************************************************** */
OPEN FILE(SMF88IN);
READ FILE(SMF88IN) INTO(WORKAREA);
SUBSTR(REAL_RECORD,1,LENGTH(WORKAREA)) = WORKAREA;
OPEN FILE(SMF88OT) OUTPUT DIRECT;
DO WHILE(¬EOF);
   /* IS IT A CICS LOGSTREAM ? */
   P = ADDR(SMF88LSN);
   STRING_FOUND = FALSE;
A: DO I = 1 TO 23;  /* LENGTH OF LOG STREAM NAME - 4 + 1 */
      IF X = 'DFHL' | X = 'DFHS' | X = 'DFHJ'
         THEN DO;
                 STRING_FOUND = TRUE;
                 LEAVE A;
              END;
      P = POINTERADD(P,1);
   END A;
   /* PROCESS RECORD */
   IF STRING_FOUND = TRUE
      THEN DO;
```

```
              C = C + 1;
              KEY_PART2 = '0000'X;
              CALL FORMAT_TOD;
              SUBSTR(WORKAREA,1,LENGTH(WORKAREA)) = REAL_RECORD;
              WRITE FILE(SMF88OT) FROM(WORKAREA)
                    KEYFROM(STRING(KEY));
           END;
   /* NEXT READ */
   READ FILE(SMF88IN) INTO(WORKAREA);
   SUBSTR(REAL_RECORD,1,LENGTH(WORKAREA)) = WORKAREA;
END;
CLOSE FILE(SMF88IN);
CLOSE FILE(SMF88OT);
/*******************************************************************/
/* CLEANUP - DELETE ALL RECORDS WITH A CREATION DATE GT 10         */
/*******************************************************************/
EOF = FALSE;
/* GET CURRENT DATE IN JULIAN DATE FORMAT */
CALL CEELOCT(TIMEDATE_JULIAN_CUR,TIMEDATE_SECONDS_CUR,
             TIMEDATE_GREGORN_CUR,FC);
OPEN FILE(SMF88OT) SEQUENTIAL UPDATE;
READ FILE(SMF88OT) INTO(WORKAREA); /* FIRST READ */
DO WHILE(¬EOF);
   SUBSTR(REAL_RECORD,1,LENGTH(WORKAREA)) = WORKAREA;
   CALL IXGR1A(SMF88LTD,TIMEDATE_CHAR_88);
   /* CONVERT SMF DATE FORMAT TO JULIAN DATE FORMAT */
   CALL CEEDAYS(SUBSTR(TIMEDATE_CHAR_88,17,8),'YYYYMMDD',
                TIMEDATE_JULIAN_88,FC);
   IF TIMEDATE_JULIAN_CUR - TIMEDATE_JULIAN_88 > CLEANUP_INTERVAL_BIN
      THEN DO;
               DELETE FILE(SMF88OT);    /* DELETE THE LAST RECORD -*/
               E = E + 1;               /* IT'S OLDER THAN 10 DAYS */
           END;
   READ FILE(SMF88OT) INTO(WORKAREA);  /* NEXT READ */
END;
CLOSE FILE(SMF88OT);
/*******************************************************************/
/* ISSUE MESSAGES, SET RETURN-CODE AND RETURN TO MVS               */
/*******************************************************************/
PUT SKIP LIST('COUNTER RECORDS ALL  :'); PUT DATA(C);
PUT SKIP LIST('COUNTER RECORDS DUPL.:'); PUT DATA(D);
PUT SKIP LIST('COUNTER RECORDS DEL. :'); PUT DATA(E);
IF C = 0 THEN CALL PLIRETC(4);  /* NO RECORDS LOADED */
IF D > 0 THEN CALL PLIRETC(8);  /* SOME DUPLICATE RECORDS */
IF D>0 & D=C THEN CALL PLIRETC(12); /* ALL RECORDS DUPLICATE */
RETURN;
FORMAT_TOD: PROC;
CALL IXGR1A(SMF88LTD,TIMEDATE_CHAR_88);
SMF88PNM = TODDATE;
SMF88LIT = TODTIME;
END FORMAT_TOD;
END IXGRPTB;
```

IXGRPTC

```
*PROCESS MACRO SYSTEM(CICS) LANGLVL(SPROG) XREF(FULL);
 IXGRPTC: PROC(COMPTR) OPTIONS(MAIN);
 /*******************************************************************/
 /* DISPLAY SMF88 DATA ONLINE                                      */
 /*******************************************************************/
 %INCLUDE IXGMAPS;   /* DSECT GENERATED BY BMS */
 %INCLUDE (DFHAID);
 DCL COMPTR PTR;
 DCL XRESP FIXED BIN(31);
 DCL XABSTIME DEC FIXED(15);
 DCL XSYSID CHAR(4), XAPPLID CHAR(8);
 DCL (ADDR,CHAR,CSTG,STG,LOW,HIGH,SUBSTR,LENGTH) BUILTIN;
 DCL STR CHAR(32767) BASED;
 DCL I,J,K,C FIXED BIN(15); /* I,J,K FOR GENERAL PURPOSES, C=CURSOR */
 DCL REQJOUR CHAR(8);
 DCL XJOURNALNAME CHAR(8),
     XSTREAMNAME  CHAR(26),
     XSTATUS      FIXED BIN(31), CSTATUS CHAR(8),
     XTYPE        FIXED BIN(31), CTYPE   CHAR(8);
 DCL XCOMMAREA CHAR(CSTG(XCOMMAREA_DATA)) INIT(
                LOW(CSTG(XCOMMAREA_DATA)))  CONTROLLED;
 DCL 1 XCOMMAREA_DATA UNAL BASED(COMPTR),
       2 EYECATCHER       CHAR(8),
       2 TAB_IDX          FIXED BIN(15),
       2 TAB_STRNM (6:17) CHAR(26),
       2 NEXT_FUNCTION    CHAR(2Ø);
 DCL END_MESSAGE CHAR(4Ø) INIT('IXGRPTC TERMINATED');
 DCL 1 TO_IXGSMF8_CA,
       2 FILLØ1    CHAR(8) INIT('IXGSMF8'),
       2 FILLØ2    CHAR(2Ø) INIT('FIRST_INVOCATION'),
       2 STRNM     CHAR(26),
       2 LSFLAGS   CHAR(2),
       2 TIMESTAMP CHAR(8),
       2 MAXITEM      FIXED BIN(15),
       2 LASTITEM     FIXED BIN(15),
       2 TSQNAME      CHAR(8),
       2 ACCUM_TAB (2Ø),
         3 ACCUMULATOR  FLOAT BIN(64),
       2 MAX_TAB (22),
         3 MAXIMUM      FLOAT BIN(64),
         3 TSQITEM      FIXED BIN(15);
 DCL SCREEN_LINES FIXED BIN(15);
 DCL SCREEN_COLS  FIXED BIN(15);
 IF EIBCALEN=Ø
    THEN DO;
             ALLOCATE XCOMMAREA;
             COMPTR=ADDR(XCOMMAREA);
             EYECATCHER='IXGRPTC';
```

```
            NEXT_FUNCTION = '***';
        END;
SELECT(EIBAID);
    WHEN(DFHPF3)  IF EIBCALEN > Ø
                     THEN
                          NEXT_FUNCTION = 'RETURN_TO_CICS';
    WHEN(DFHENTER)
      DO; IF EIBCALEN>Ø  THEN
          DO;
              C=EIBCPOSN/8Ø;
              IF (TAB_IDX>5 & C>TAB_IDX) | C<6 | C>17
                 THEN DO;
                         EXEC CICS SEND MAP ('INVCURS')
                                   MAPSET('IXGMAPS')
                                   RESP(XRESP);
                         GOTO RETURN_TO_CICS;
                      END;
              EXEC CICS ASSIGN ALTSCRNHT(SCREEN_LINES)
                               ALTSCRNWD(SCREEN_COLS)
                               RESP(XRESP);
              IF SCREEN_LINES < 27 | SCREEN_COLS < 132
                 THEN DO;
                         EXEC CICS SEND MAP ('INVSCRN')
                                   MAPSET('IXGMAPS')
                                   RESP(XRESP);
                         GOTO RETURN_TO_CICS;
                      END;
              /* START READING SMF88 FILE */
              STRNM=TAB_STRNM(C);
              LSFLAGS=LOW(LENGTH(LSFLAGS));
              TIMESTAMP=LOW(LENGTH(TIMESTAMP));
              MAXITEM=Ø; LASTITEM=Ø;
              EXEC CICS RETURN TRANSID('IXG8') IMMEDIATE
                        COMMAREA(TO_IXGSMF8_CA)
                        LENGTH(CSTG(TO_IXGSMF8_CA))
                        RESP(XRESP);
          END;
      END;
    OTHERWISE DO;
                 EXEC CICS SEND MAP ('INVKEY')
                           MAPSET('IXGMAPS')
                           RESP(XRESP);
                 GOTO RETURN_TO_CICS;
              END;
END;
SELECT (NEXT_FUNCTION);
    WHEN ('RETURN_TO_CICS') DO;
                               EXEC CICS SEND TEXT
                               FROM(END_MESSAGE)
                               ERASE LAST
                               RESP(XRESP);
                               EXEC CICS RETURN;  /* STOP RUN */
                            END;
    OTHERWISE;
END;  /* END SELECT */
LØ1Ø:      /* SEND FIRST MAP */
/* CLEAR MAP */
SUBSTR(ADDR(IXGMAP1O)->STR,1,STG(IXGMAP1O))=LOW(STG(IXGMAP1O));
TAB_IDX=5; /* SET TAB_IDX TO THE 5TH. LINE ON SCREEN */
/* VARIABLES INTO MAP */
EXEC CICS ASSIGN SYSID(XSYSID) APPLID(XAPPLID);
SYSIDO='SYSID='||XSYSID||' APPLID='||XAPPLID;
EXEC CICS INQUIRE JOURNALNAME START RESP(XRESP);
DO I=1 TO 12 UNTIL(XRESP=DFHRESP(END));
   EXEC CICS INQUIRE JOURNALNAME(XJOURNALNAME) NEXT STATUS(XSTATUS)
             STREAMNAME(XSTREAMNAME) TYPE(XTYPE) RESP(XRESP);
   IF XRESP¬=DFHRESP(NORMAL) THEN LEAVE;
   SELECT(XSTATUS);
       WHEN (DFHVALUE(ENABLED))  CSTATUS='ENABLED';
       WHEN (DFHVALUE(DISABLED)) CSTATUS='DISABLED';
       WHEN (DFHVALUE(FAILED))   CSTATUS='FAILED';
       OTHERWISE                 CSTATUS='   ???';
   END;
   SELECT(XTYPE);
       WHEN (DFHVALUE(MVS))    CTYPE='MVS';
       WHEN (DFHVALUE(SMF))    CTYPE='SMF';
       WHEN (DFHVALUE(DUMMY))  CTYPE='DUMMY';
       OTHERWISE               CTYPE='   ???';
   END;
   LSNO(I)=(9)' '||XJOURNALNAME||(4)' '||XSTREAMNAME||
           (2)' '||CTYPE||CSTATUS;
   /* SAVE THE STREAMNAME IN TAB */
   TAB_IDX=TAB_IDX+1; TAB_STRNM(TAB_IDX)=XSTREAMNAME;
END;
/* SEND MAP */
EXEC CICS SEND MAP('IXGMAP1') MAPSET('IXGMAPS') FROM(IXGMAP1O)
          ERASE RESP(XRESP);
RETURN_TO_CICS:
EXEC CICS RETURN TRANSID(EIBTRNID) COMMAREA(XCOMMAREA_DATA)
          LENGTH(CSTG(XCOMMAREA_DATA)) RESP(XRESP);
END IXGRPTC;
```

IXGSMF8

```
*PROCESS MACRO SYSTEM(CICS) LANGLVL(SPROG) XREF(FULL);
 IXGSMF8: PROC(COMPTR) OPTIONS(MAIN);
 /*******************************************************************/
 /* READ SMF88 FILE AND DISPLAY SMF88 VARIABLES ON SCREEN           */
 /*******************************************************************/
```

```
%INCLUDE IXGMAPS;   /* DSECT GENERATED BY BMS */
%INCLUDE SMF88STR;
%INCLUDE (DFHAID);
%INCLUDE (DFHBMSCA);
DCL (COMPTR,P) PTR;
DCL NULL_CA CHAR(1), ZERO FIXED BIN(15) INIT(Ø);
DCL XLENGTH FIXED BIN(15);
DCL XRESP FIXED BIN(31);
DCL XABSTIME DEC FIXED(15);
DCL XSYSID CHAR(4), XAPPLID CHAR(8);
DCL LONG_FLOAT_BIN FLOAT BINARY(64) BASED;
DCL WRK_BIN    FIXED BIN(31);
DCL WRK_PACKED FIXED DEC(15);
DCL WRK_FLOAT  FLOAT BIN(64);
DCL SYSPRINT FILE;
DCL (ADDR,CHAR,CSTG,STG,LOW,SUBSTR,STRING,LENGTH,FLOAT,BIN,MIN,
     UNSPEC,LBOUND,HBOUND) BUILTIN;
DCL STR CHAR(32767) BASED;
DCL I,J,K,C FIXED BIN(15); /* I,J,K FOR GENERAL PURPOSES, C=CURSOR */
DCL STRNM_SAVE CHAR(CSTG(STRNM));
DCL 1 XCOMMAREA BASED(COMPTR),
      2 EYECATCHER CHAR(8),
      2 NEXT_FUNCTION CHAR(2Ø),
      2 XKEY,
        3 STRNM      CHAR(26),
        3 LSFLAGS    CHAR(2),
        3 TIMESTAMP  CHAR(8),
      2 MAXITEM      FIXED BIN(15),
      2 CURRITEM     FIXED BIN(15),
      2 TSQNAME      CHAR(8),
      2 ACCUM_TAB (2Ø),
        3 ACCUMULATOR  FLOAT BIN(64),
      2 MAX_TAB (22),
        3 MAXIMUM      FLOAT BIN(64),
        3 TSQITEM      FIXED BIN(15);
  /* --------------------------------------------------------------- */
DCL SMF88SWB_FLOAT BINARY(64) FLOAT; /* BYT WRITTN TO INTERIM STOR. */
DCL SMF88LDB_FLOAT BINARY(64) FLOAT; /* BYT WRITTN TO DASD          */
DCL SMF88SIB_FLOAT BINARY(64) FLOAT; /* BYT DELETD INT.W/O DASD     */
DCL SMF88SAB_FLOAT BINARY(64) FLOAT; /* BYT DELETD INTERIM ST W/DASD*/
DCL SMF88LWB_FLOAT BINARY(64) FLOAT; /*                             */
  /* --- CONSTANTS ------------------------------------------------- */
DCL CONST_ZERO_BIN15      FIXED BINARY (15) STATIC INIT(Ø);
DCL CONST_SIGNIF_DIGITS   FIXED BINARY (15) STATIC INIT(14);
DCL CONST_MAX_EXP         FIXED BINARY (15) STATIC INIT(16);
DCL SPACE                 CHAR(1)           STATIC INIT(' ');
DCL SMF88LWI_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(1),
    SMF88LWB_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(2),
    SMF88SWB_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(3),
    SMF88LDB_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(4),
    SMF88SIB_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(5),
    SMF88SAB_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(6),
    SMF88SII_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(7),
    SMF88SAI_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(8),
    SMF88SC1_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(9),
    SMF88SC2_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(1Ø),
    SMF88SC3_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(11),
    SMF88EDS_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(12),
    SMF88ERI_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(13),
    SMF88ERC_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(14),
    SMF88ESF_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(15),
    SMF88ETT_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(16),
    SMF88ETF_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(17),
    SMF88EOA_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(18),
    SMF88EFS_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(19),
    SMF88EDO_ACCUM_TABIDX FIXED BIN(15) STATIC INIT(2Ø);
DCL SMF88LWI_MAX_TABIDX FIXED BIN(15) STATIC INIT(1),
    SMF88LWB_MAX_TABIDX FIXED BIN(15) STATIC INIT(2),
    SMF88SWB_MAX_TABIDX FIXED BIN(15) STATIC INIT(3),
    SMF88LDB_MAX_TABIDX FIXED BIN(15) STATIC INIT(4),
    SMF88SIB_MAX_TABIDX FIXED BIN(15) STATIC INIT(5),
    SMF88SAB_MAX_TABIDX FIXED BIN(15) STATIC INIT(6),
    SMF88SII_MAX_TABIDX FIXED BIN(15) STATIC INIT(7),
    SMF88SAI_MAX_TABIDX FIXED BIN(15) STATIC INIT(8),
    SMF88SC1_MAX_TABIDX FIXED BIN(15) STATIC INIT(9),
    SMF88SC2_MAX_TABIDX FIXED BIN(15) STATIC INIT(1Ø),
    SMF88SC3_MAX_TABIDX FIXED BIN(15) STATIC INIT(11),
    SMF88EDS_MAX_TABIDX FIXED BIN(15) STATIC INIT(12),
    SMF88ERI_MAX_TABIDX FIXED BIN(15) STATIC INIT(13),
    SMF88ERC_MAX_TABIDX FIXED BIN(15) STATIC INIT(14),
    SMF88ESF_MAX_TABIDX FIXED BIN(15) STATIC INIT(15),
    SMF88ETT_MAX_TABIDX FIXED BIN(15) STATIC INIT(16),
    SMF88ETF_MAX_TABIDX FIXED BIN(15) STATIC INIT(17),
    SMF88EOA_MAX_TABIDX FIXED BIN(15) STATIC INIT(18),
    SMF88EFS_MAX_TABIDX FIXED BIN(15) STATIC INIT(19),
    SMF88EDO_MAX_TABIDX FIXED BIN(15) STATIC INIT(2Ø),
    SMF88LAB_MAX_TABIDX FIXED BIN(15) STATIC INIT(21),
    SMF88LIB_MIN_TABIDX FIXED BIN(15) STATIC INIT(22);
%PAGE;
/*******************************************************************/
/*                   MAIN TASK CONTROL                             */
SELECT(EIBAID);
    WHEN(DFHENTER);
    WHEN(DFHPF3)   NEXT_FUNCTION = 'RETURN_TO_IXGC';
    WHEN(DFHPF7)   NEXT_FUNCTION = 'PROCESS_PREV_ITEM';
    WHEN(DFHPF8)   NEXT_FUNCTION = 'PROCESS_NEXT_ITEM';
    WHEN(DFHPF9)   NEXT_FUNCTION = 'PROCESS_FIRST_ITEM';
    WHEN(DFHPF1Ø)  NEXT_FUNCTION = 'PROCESS_MIDLE_ITEM';
    WHEN(DFHPF11)  NEXT_FUNCTION = 'PROCESS_LAST_ITEM';
    WHEN(DFHPF13)  NEXT_FUNCTION = 'DASD_SHFT_MAX';
    WHEN(DFHPF14)  NEXT_FUNCTION = 'OFFLOAD_MAX';
    WHEN(DFHPF15)  NEXT_FUNCTION = 'STG_FULL';
```

```
        WHEN(DFHPF16)  NEXT_FUNCTION = 'STG_THLD';
        WHEN(DFHPF17)  NEXT_FUNCTION = 'STR_FULL';
        WHEN(DFHPF18)  NEXT_FUNCTION = 'OFFL_90%';
        WHEN(DFHPF23)  NEXT_FUNCTION = 'AVERAGE';
        WHEN(DFHPF24)  NEXT_FUNCTION = 'SUMMARY';
        OTHERWISE      NEXT_FUNCTION = 'INVALID_PFKEY';
    END;
    SELECT(NEXT_FUNCTION);
        WHEN('FIRST_INVOCATION ') DO;
                                      CALL PROC_INIT;
                                      CALL PROC_READ_FILE;
                                      NEXT_FUNCTION = '???';
                                  END;
        WHEN('PROCESS_FIRST_ITEM') DO;
                                   CURRITEM = 1;
                                   CALL DISPLAY_SMF88_INTERVAL(CURRITEM);
                                   END;
        WHEN('PROCESS_MIDLE_ITEM') DO;
                                   CURRITEM = MAXITEM / 2;
                                   CALL DISPLAY_SMF88_INTERVAL(CURRITEM);
                                   END;
        WHEN('PROCESS_LAST_ITEM')  DO;
                                   CURRITEM = MAXITEM;
                                   CALL DISPLAY_SMF88_INTERVAL(MAXITEM);
                                   END;
        WHEN('RETURN_TO_IXGC   ') DO;
                                    EXEC CICS DELETEQ TS QUEUE(TSQNAME)
                                           RESP(XRESP);
                                    EXEC CICS RETURN TRANSID('IXGC')
                                           IMMEDIATE
                                           COMMAREA(NULL_CA)
                                           LENGTH(ZERO)
                                           RESP(XRESP);
                                  END;
        WHEN('PROCESS_NEXT_ITEM')  DO;
                                     IF CURRITEM = MAXITEM
                                         THEN CURRITEM = 1;
                                         ELSE IF CURRITEM < MAXITEM
                                             THEN CURRITEM = CURRITEM + 1;
                                    CALL DISPLAY_SMF88_INTERVAL(CURRITEM);
                                  END;
        WHEN('PROCESS_PREV_ITEM')  DO;
                                     IF CURRITEM = 1
                                         THEN CURRITEM = MAXITEM;
                                         ELSE IF CURRITEM > 1
                                             THEN CURRITEM = CURRITEM - 1;
                                    CALL DISPLAY_SMF88_INTERVAL(CURRITEM);
                                  END;
        WHEN('DASD_SHFT_MAX') DO;
                                 CURRITEM=TSQITEM(SMF88EDS_MAX_TABIDX);
                                 CALL DISPLAY_SMF88_INTERVAL(CURRITEM);
```

```
                          END;
    WHEN('OFFLOAD_MAX') DO;
                           CURRITEM=TSQITEM(SMF88EOA_MAX_TABIDX);
                           CALL DISPLAY_SMF88_INTERVAL(CURRITEM);
                        END;
    WHEN('STG_FULL') DO;
                        CURRITEM=TSQITEM(SMF88ETF_MAX_TABIDX);
                        CALL DISPLAY_SMF88_INTERVAL(CURRITEM);
                     END;
    WHEN('STG_THLD') DO;
                        CURRITEM=TSQITEM(SMF88ETT_MAX_TABIDX);
                        CALL DISPLAY_SMF88_INTERVAL(CURRITEM);
                     END;
    WHEN('STR_FULL') DO;
                        CURRITEM=TSQITEM(SMF88LDB_MAX_TABIDX);
                        CALL DISPLAY_SMF88_INTERVAL(CURRITEM);
                     END;
    WHEN('OFFL_9Ø%') DO;
                        CURRITEM=TSQITEM(SMF88EFS_MAX_TABIDX);
                        CALL DISPLAY_SMF88_INTERVAL(CURRITEM);
                     END;
    WHEN('SUMMARY')  CALL PROC_SUMM;
    WHEN('AVERAGE')  CALL PROC_AVG;
    WHEN('INVALID_PFKEY') DO; CALL CLEAR_MAP;
                              MSGO='*** INVALID PF KEY ***. PRESS ' ||
                                   'ONE OF THE PF KEYS DISPLAYED ' ||
                                   'BELOW|';
                              MSGH=DFHREVRS;
                              CALL SEND_MAP;
                          END;
    OTHERWISE;
END;
EXEC CICS RETURN TRANSID(EIBTRNID) COMMAREA(XCOMMAREA)
          LENGTH(CSTG(XCOMMAREA)) RESP(XRESP);
%PAGE;
/********************************************************************/
/* READ THE REQUESTED SMF88 RECORDS INTO A TEMPORARY STORAGE AREA  */
PROC_READ_FILE: PROC OPTIONS(REENTRANT) REORDER;
STRNM_SAVE=STRNM;
TSQNAME=EIBTRNID||EIBTRMID;
MAXITEM=Ø; CURRITEM=Ø;
EXEC CICS DELETEQ TS QUEUE(TSQNAME)
          RESP(XRESP);
EXEC CICS STARTBR FILE('IXGSMF88')
          RIDFLD(XKEY) GTEQ
          RESP(XRESP);
EXEC CICS READNEXT FILE('IXGSMF88') SET(P) LENGTH(XLENGTH)
          RIDFLD(XKEY) RESP(XRESP);
DO WHILE(STRNM_SAVE=STRNM);
   IF XRESP¬=DFHRESP(NORMAL) THEN LEAVE;
   EXEC CICS WRITEQ TS QUEUE(TSQNAME) FROM(P->STR) LENGTH(XLENGTH)
```

```
             RESP(XRESP);
   MAXITEM=MAXITEM+1; CURRITEM=CURRITEM+1;
   /* MOVE TO REAL_RECORD AND ACCUMULATE THE VALUES */
   SUBSTR(REAL_RECORD,1,XLENGTH) = SUBSTR(P->STR,1,XLENGTH);
   CALL PROC_ACCUMULATE_AND_MAX;
   /* READ NEXT */
   EXEC CICS READNEXT FILE('IXGSMF88') SET(P) LENGTH(XLENGTH)
             RIDFLD(XKEY) RESP(XRESP);
END;
EXEC CICS ENDBR FILE('IXGSMF88')
          RESP(XRESP);
CALL DISPLAY_SMF88_INTERVAL(CURRITEM);
END PROC_READ_FILE;
%PAGE;
/*******************************************************************/
/* DISPLAY THE SMF88 INTERVAL ON CRT                               */
DISPLAY_SMF88_INTERVAL: PROC (INTV) OPTIONS(REENTRANT) REORDER;
DCL INTV FIXED BIN(15);
CALL CLEAR_MAP;
IF MAXITEM=Ø
   THEN DO;
           DO I=CSTG(STRNM) BY -1 TO 1;
              IF SUBSTR(STRNM_SAVE,I,1)¬=SPACE THEN LEAVE;
           END;
           MSGO=' *** NO DATA FOUND FOR LOGSTREAM "' ||
                SUBSTR(STRNM_SAVE,1,I) ||
                '" ON VSAM FILE. USE IXGRPTB TO COPY SMF88 DATA. ***';
           MSGH=DFHREVRS;
           GOTO EXIT_DISPLAY_SMF88_INTERVAL;
        END;
IF MAXITEM>Ø & MAXITEM=CURRITEM
   THEN DO;
           MSGO='NOTE: THIS IS THE MOST RECENT SMF INTERVAL. ' ||
                'YOU''LL SEE THE SMF INTERVAL COUNTER IN THE '||
                'UPPER-RIGHT CORNER.   (CURRENT / MAXIMUM) ';
           MSGC=DFHPINK;
        END;
EXEC CICS READQ TS QUEUE(TSQNAME) INTO(REAL_RECORD) ITEM(INTV)
          RESP(XRESP);
MCURINTVO = INTV;
MMAXINTVO = MAXITEM;
/* ************************************************ */
/*           PRODUCT SECTION                        */
SMF88SYNO = SMF88SYN;  /* MVS OPERATION SYSTEM NAME */
SMF88OSLO = SMF88OSL;  /* MVS RELEASE               */
/* ************************************************ */
/*           LOGSTREAM SECTION                      */
SMF88LSNO = SMF88LSN;  /* LOG STREAM NAME           */
SMF88LWIO = FLOAT(UNSPEC(SMF88LWI),32);  /* #WRITES INVOKED  */
SMF88LTDO = SUBSTR(SMF88PNM,1,4) || '/' ||    /* YYYY */
            SUBSTR(SMF88PNM,5,2) || '/' ||    /* MM   */
            SUBSTR(SMF88PNM,7,2) || SPACE ||  /* DD   */
            SUBSTR(SMF88LIT,1,2) || ':' ||  /* HH   */
            SUBSTR(SMF88LIT,3,2) || ':' ||  /* MM   */
            SUBSTR(SMF88LIT,5,2);           /* SS   */
SMF88LIBO = FLOAT(UNSPEC(SMF88LIB),32);  /* MIN.BLOCKLEN  */
SMF88LABO = FLOAT(UNSPEC(SMF88LAB),32);  /* MAX.BLOCKLEN  */
/* ------------- BYT WRITTN BY USERS IXGWRITES -------------- */
SMF88LWB_FLOAT = FLOAT(Ø);
CALL CONVERT_TO_FLOAT_AND_SUM(ADDR(SMF88LWB),ADDR(SMF88LWB_FLOAT));
SMF88LWBO  = SMF88LWB_FLOAT;
/* ************************************************ */
/*           STRUCTURE (INTERIM STORAGE) SECTION    */
SMF88STNO = SMF88STN;     /* STRUCTURE  NAME           */
/* ------------- BYT WRITTN TO INTERIM STORAGE -------------- */
SMF88SWB_FLOAT = Ø;
CALL CONVERT_TO_FLOAT_AND_SUM(ADDR(SMF88SWB),ADDR(SMF88SWB_FLOAT));
SMF88SWBO = SMF88SWB_FLOAT;
/* ------------- BYT WRITTN TO DASD ------------------------- */
SMF88LDB_FLOAT = Ø;
CALL CONVERT_TO_FLOAT_AND_SUM(ADDR(SMF88LDB),ADDR(SMF88LDB_FLOAT));
SMF88LDBO = SMF88LDB_FLOAT;
/* ------------- BYT DELETD INTERIM ST W/O DASD ------------- */
SMF88SIB_FLOAT = Ø;
CALL CONVERT_TO_FLOAT_AND_SUM(ADDR(SMF88SIB),ADDR(SMF88SIB_FLOAT));
SMF88SIBO = SMF88SIB_FLOAT;
/* ------------- BYT DELETD INTERIM ST W/DASD --------------- */
SMF88SAB_FLOAT = Ø;
CALL CONVERT_TO_FLOAT_AND_SUM(ADDR(SMF88SAB),ADDR(SMF88SAB_FLOAT));
SMF88SABO = SMF88SAB_FLOAT;
SMF88SIIO = FLOAT(UNSPEC(SMF88SII),32); /* # DELETES W/O DASD WRITE */
SMF88SAIO = FLOAT(UNSPEC(SMF88SAI),32); /* # DELETS W/WRITE         */
SMF88SC1O = FLOAT(UNSPEC(SMF88SC1),32); /* # WRITES COMPLETED TYPE1 */
SMF88SC2O = FLOAT(UNSPEC(SMF88SC2),32); /* # WRITES COMPLETED TYPE2 */
SMF88SC3O = FLOAT(UNSPEC(SMF88SC3),32); /* # WRITES COMPLETED TYPE3 */
/* ************************************************ */
/*           EVENTS SECTION                         */
SMF88EDSO = FLOAT(UNSPEC(SMF88EDS),32); /* DASD SHFT  */
SMF88ERIO = FLOAT(UNSPEC(SMF88ERI),32); /* REBLD INI  */
SMF88ERCO = FLOAT(UNSPEC(SMF88ERC),32); /* REBLD CMP  */
SMF88ESFO = FLOAT(UNSPEC(SMF88ESF),32); /* STRC FULL  */
SMF88ETTO = FLOAT(UNSPEC(SMF88ETT),32); /* STG THLD   */
SMF88ETFO = FLOAT(UNSPEC(SMF88ETF),32); /* STG FULL   */
SMF88EOAO = FLOAT(UNSPEC(SMF88EO ),32); /* OFFLOADS   */
SMF88EFSO = FLOAT(UNSPEC(SMF88EFS),32); /* OFFL.9Ø%   */
SMF88EDOO = FLOAT(UNSPEC(SMF88EDO),32); /* IXGOFFLD   */
SELECT(NEXT_FUNCTION);   /* SET MAP ATTRIBUTE TO REVERSE VIDEO */
  WHEN('PROCESS_PREV_ITEM')     PF7H=DFHREVRS;
  WHEN('PROCESS_NEXT_ITEM')     PF8H=DFHREVRS;
  WHEN('PROCESS_FIRST_ITEM')    PF9H=DFHREVRS;
  WHEN('PROCESS_MIDLE_ITEM')    PF1ØH=DFHREVRS;
  WHEN('PROCESS_LAST_ITEM')     PF11H=DFHREVRS;
  WHEN('DASD_SHFT_MAX') DO; SMF88EDSH=DFHREVRS; PF13H=DFHREVRS; END;
  WHEN('OFFLOAD_MAX')   DO; SMF88EOAH=DFHREVRS; PF14H=DFHREVRS; END;
  WHEN('STG_FULL')      DO; SMF88ETFH=DFHREVRS; PF15H=DFHREVRS; END;
  WHEN('STG_THLD')      DO; SMF88ETTH=DFHREVRS; PF16H=DFHREVRS; END;
  WHEN('STR_FULL')      DO; SMF88ESFH=DFHREVRS; PF17H=DFHREVRS; END;
  WHEN('OFFL_9Ø%')      DO; SMF88EFSH=DFHREVRS; PF18H=DFHREVRS; END;
  OTHERWISE;
END;
EXIT_DISPLAY_SMF88_INTERVAL:
CALL SEND_MAP;
RETURN;
END DISPLAY_SMF88_INTERVAL;
%PAGE;
PROC_ACCUMULATE_AND_MAX: PROC  OPTIONS(REENTRANT) REORDER;
/* SMF88LWI */
ACCUMULATOR(SMF88LWI_ACCUM_TABIDX) =
ACCUMULATOR(SMF88LWI_ACCUM_TABIDX) + FLOAT(UNSPEC(SMF88LWI),32);
IF MAXIMUM(SMF88LWI_MAX_TABIDX) <= FLOAT(UNSPEC(SMF88LWI),32) THEN
 DO;
   MAXIMUM(SMF88LWI_MAX_TABIDX) = FLOAT(UNSPEC(SMF88LWI),32);
   TSQITEM(SMF88LWI_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88LIB  (EXCEPTION: NOT MAX BUT MIN) */
IF MAXIMUM(SMF88LIB_MIN_TABIDX) >= FLOAT(UNSPEC(SMF88LIB),32) THEN
 DO;
   MAXIMUM(SMF88LIB_MIN_TABIDX) = FLOAT(UNSPEC(SMF88LIB),32);
   TSQITEM(SMF88LIB_MIN_TABIDX) = CURRITEM;
 END;
/* SMF88LAB  */
IF MAXIMUM(SMF88LAB_MAX_TABIDX) <= FLOAT(UNSPEC(SMF88LAB),32) THEN
 DO;
   MAXIMUM(SMF88LAB_MAX_TABIDX) = FLOAT(UNSPEC(SMF88LAB),32);
   TSQITEM(SMF88LAB_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88LWB  */
CALL CONVERT_TO_FLOAT_AND_SUM(ADDR(SMF88LWB),
          ADDR(ACCUMULATOR(SMF88LWB_ACCUM_TABIDX)));
IF MAXIMUM(SMF88LWB_MAX_TABIDX) <= WRK_FLOAT THEN
 DO;
   MAXIMUM(SMF88LWB_MAX_TABIDX) = WRK_FLOAT;
   TSQITEM(SMF88LWB_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88SWB  */
CALL CONVERT_TO_FLOAT_AND_SUM(ADDR(SMF88SWB),
          ADDR(ACCUMULATOR(SMF88SWB_ACCUM_TABIDX)));
IF MAXIMUM(SMF88SWB_MAX_TABIDX) <= WRK_FLOAT THEN
 DO;
   MAXIMUM(SMF88SWB_MAX_TABIDX) = WRK_FLOAT;
   TSQITEM(SMF88SWB_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88LDB  */
```

```
CALL CONVERT_TO_FLOAT_AND_SUM(ADDR(SMF88LDB),
          ADDR(ACCUMULATOR(SMF88LDB_ACCUM_TABIDX)));
IF MAXIMUM(SMF88LDB_MAX_TABIDX) <= WRK_FLOAT THEN
 DO;
   MAXIMUM(SMF88LDB_MAX_TABIDX) = WRK_FLOAT;
   TSQITEM(SMF88LDB_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88SIB  */
CALL CONVERT_TO_FLOAT_AND_SUM(ADDR(SMF88SIB),
          ADDR(ACCUMULATOR(SMF88SIB_ACCUM_TABIDX)));
IF MAXIMUM(SMF88SIB_MAX_TABIDX) <= WRK_FLOAT THEN
 DO;
   MAXIMUM(SMF88SIB_MAX_TABIDX) = WRK_FLOAT;
   TSQITEM(SMF88SIB_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88SAB  */
CALL CONVERT_TO_FLOAT_AND_SUM(ADDR(SMF88SAB),
          ADDR(ACCUMULATOR(SMF88SAB_ACCUM_TABIDX)));
IF MAXIMUM(SMF88SAB_MAX_TABIDX) <= WRK_FLOAT THEN
 DO;
   MAXIMUM(SMF88SAB_MAX_TABIDX) = WRK_FLOAT;
   TSQITEM(SMF88SAB_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88SII */
ACCUMULATOR(SMF88SII_ACCUM_TABIDX) =
ACCUMULATOR(SMF88SII_ACCUM_TABIDX) + FLOAT(UNSPEC(SMF88SII),32);
IF MAXIMUM(SMF88SII_MAX_TABIDX) <= FLOAT(UNSPEC(SMF88SII),32) THEN
 DO;
   MAXIMUM(SMF88SII_MAX_TABIDX) = FLOAT(UNSPEC(SMF88SII),32);
   TSQITEM(SMF88SII_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88SAI */
ACCUMULATOR(SMF88SAI_ACCUM_TABIDX) =
ACCUMULATOR(SMF88SAI_ACCUM_TABIDX) + FLOAT(UNSPEC(SMF88SAI),32);
IF MAXIMUM(SMF88SAI_MAX_TABIDX) <= FLOAT(UNSPEC(SMF88SAI),32) THEN
 DO;
   MAXIMUM(SMF88SAI_MAX_TABIDX) = FLOAT(UNSPEC(SMF88SAI),32);
   TSQITEM(SMF88SAI_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88SC1 */
ACCUMULATOR(SMF88SC1_ACCUM_TABIDX) =
ACCUMULATOR(SMF88SC1_ACCUM_TABIDX) + FLOAT(UNSPEC(SMF88SC1),32);
IF MAXIMUM(SMF88SC1_MAX_TABIDX) <= FLOAT(UNSPEC(SMF88SC1),32) THEN
 DO;
   MAXIMUM(SMF88SC1_MAX_TABIDX) = FLOAT(UNSPEC(SMF88SC1),32);
   TSQITEM(SMF88SC1_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88SC2 */
ACCUMULATOR(SMF88SC2_ACCUM_TABIDX) =
ACCUMULATOR(SMF88SC2_ACCUM_TABIDX) + FLOAT(UNSPEC(SMF88SC2),32);
IF MAXIMUM(SMF88SC2_MAX_TABIDX) <= FLOAT(UNSPEC(SMF88SC2),32) THEN
```

```
 DO;
   MAXIMUM(SMF88SC2_MAX_TABIDX) = FLOAT(UNSPEC(SMF88SC2),32);
   TSQITEM(SMF88SC2_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88SC3 */
ACCUMULATOR(SMF88SC3_ACCUM_TABIDX) =
ACCUMULATOR(SMF88SC3_ACCUM_TABIDX) + FLOAT(UNSPEC(SMF88SC3),32);
IF MAXIMUM(SMF88SC3_MAX_TABIDX) <= FLOAT(UNSPEC(SMF88SC3),32) THEN
 DO;
   MAXIMUM(SMF88SC3_MAX_TABIDX) = FLOAT(UNSPEC(SMF88SC3),32);
   TSQITEM(SMF88SC3_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88EDS */
ACCUMULATOR(SMF88EDS_ACCUM_TABIDX) =
ACCUMULATOR(SMF88EDS_ACCUM_TABIDX) + FLOAT(UNSPEC(SMF88EDS),32);
IF MAXIMUM(SMF88EDS_MAX_TABIDX) <= FLOAT(UNSPEC(SMF88EDS),32) THEN
 DO;
   MAXIMUM(SMF88EDS_MAX_TABIDX) = FLOAT(UNSPEC(SMF88EDS),32);
   TSQITEM(SMF88EDS_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88ERI */
ACCUMULATOR(SMF88ERI_ACCUM_TABIDX) =
ACCUMULATOR(SMF88ERI_ACCUM_TABIDX) + FLOAT(UNSPEC(SMF88ERI),32);
IF MAXIMUM(SMF88ERI_MAX_TABIDX) <= FLOAT(UNSPEC(SMF88ERI),32) THEN
 DO;
   MAXIMUM(SMF88ERI_MAX_TABIDX) = FLOAT(UNSPEC(SMF88ERI),32);
   TSQITEM(SMF88ERI_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88ERC */
ACCUMULATOR(SMF88ERC_ACCUM_TABIDX) =
ACCUMULATOR(SMF88ERC_ACCUM_TABIDX) + FLOAT(UNSPEC(SMF88ERC),32);
IF MAXIMUM(SMF88ERC_MAX_TABIDX) <= FLOAT(UNSPEC(SMF88ERC),32) THEN
 DO;
   MAXIMUM(SMF88ERC_MAX_TABIDX) = FLOAT(UNSPEC(SMF88ERC),32);
   TSQITEM(SMF88ERC_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88ESF */
ACCUMULATOR(SMF88ESF_ACCUM_TABIDX) =
ACCUMULATOR(SMF88ESF_ACCUM_TABIDX) + FLOAT(UNSPEC(SMF88ESF),32);
IF MAXIMUM(SMF88ESF_MAX_TABIDX) <= FLOAT(UNSPEC(SMF88ESF),32) THEN
 DO;
   MAXIMUM(SMF88ESF_MAX_TABIDX) = FLOAT(UNSPEC(SMF88ESF),32);
   TSQITEM(SMF88ESF_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88ETT */
ACCUMULATOR(SMF88ETT_ACCUM_TABIDX) =
ACCUMULATOR(SMF88ETT_ACCUM_TABIDX) + FLOAT(UNSPEC(SMF88ETT),32);
IF MAXIMUM(SMF88ETT_MAX_TABIDX) <= FLOAT(UNSPEC(SMF88ETT),32) THEN
 DO;
   MAXIMUM(SMF88ETT_MAX_TABIDX) = FLOAT(UNSPEC(SMF88ETT),32);
   TSQITEM(SMF88ETT_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88ETF */
ACCUMULATOR(SMF88ETF_ACCUM_TABIDX) =
ACCUMULATOR(SMF88ETF_ACCUM_TABIDX) + FLOAT(UNSPEC(SMF88ETF),32);
IF MAXIMUM(SMF88ETF_MAX_TABIDX) <= FLOAT(UNSPEC(SMF88ETF),32) THEN
 DO;
   MAXIMUM(SMF88ETF_MAX_TABIDX) = FLOAT(UNSPEC(SMF88ETF),32);
   TSQITEM(SMF88ETF_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88EOA */
ACCUMULATOR(SMF88EOA_ACCUM_TABIDX) =
ACCUMULATOR(SMF88EOA_ACCUM_TABIDX) + FLOAT(UNSPEC(SMF88EO),32);
IF MAXIMUM(SMF88EOA_MAX_TABIDX) <= FLOAT(UNSPEC(SMF88EO),32) THEN
 DO;
   MAXIMUM(SMF88EOA_MAX_TABIDX) = FLOAT(UNSPEC(SMF88EO),32);
   TSQITEM(SMF88EOA_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88EFS */
ACCUMULATOR(SMF88EFS_ACCUM_TABIDX) =
ACCUMULATOR(SMF88EFS_ACCUM_TABIDX) + FLOAT(UNSPEC(SMF88EFS),32);
IF MAXIMUM(SMF88EFS_MAX_TABIDX) <= FLOAT(UNSPEC(SMF88EFS),32) THEN
 DO;
   MAXIMUM(SMF88EFS_MAX_TABIDX) = FLOAT(UNSPEC(SMF88EFS),32);
   TSQITEM(SMF88EFS_MAX_TABIDX) = CURRITEM;
 END;
/* SMF88EDO */
ACCUMULATOR(SMF88EDO_ACCUM_TABIDX) =
ACCUMULATOR(SMF88EDO_ACCUM_TABIDX) + FLOAT(UNSPEC(SMF88EDO),32);
IF MAXIMUM(SMF88EDO_MAX_TABIDX) <= FLOAT(UNSPEC(SMF88EDO),32) THEN
 DO;
   MAXIMUM(SMF88EDO_MAX_TABIDX) = FLOAT(UNSPEC(SMF88EDO),32);
   TSQITEM(SMF88EDO_MAX_TABIDX) = CURRITEM;
 END;
END PROC_ACCUMULATE_AND_MAX;
%PAGE;
PROC_INIT: PROC;
DO I = 1 TO 2Ø;
   ACCUMULATOR(I) = FLOAT(Ø);
END;
DO I = 1 TO 22;
   MAXIMUM(I) = FLOAT(Ø);
   TSQITEM(I) = 1;
END;
END PROC_INIT;
%PAGE;
 CONVERT_TO_FLOAT_AND_SUM : PROC
                 (SOURCE_STRING_PTR, FLOAT_ACCUM_PTR);
    DECLARE
      SOURCE_STRING_PTR POINTER,
      SOURCE_STRING BIT(64) BASED(SOURCE_STRING_PTR), /* INPUT: FORMAT
                                          IS ASM LONG FLOATING POINT */
      FIRST_BYTE    BIT(8)  BASED(SOURCE_STRING_PTR), /* EXPONENT OF
                                         ASM LONG FLOATING POINT    */
      TARGET_STRING BIT(64),             /*    TEMP 64 BIT WORKAREA */
      FLOAT_ACCUM_PTR POINTER,
      FLOAT_ACCUM BINARY(64) FLOAT BASED(FLOAT_ACCUM_PTR); /* OUTPUT:
                                         PL/I FORMAT 64-BIT FLOAT   */
    DECLARE
      CHARACTERISTIC            BINARY(15),
      DIGITS_TO_MOVE            BINARY(15),
      NUM_BITS_TO_MOVE          BINARY(15),
      SOURCE_START_SUBSCRIPT    BINARY(15),
      TARGET_START_SUBSCRIPT    BINARY(15);
    CHARACTERISTIC = BIN (FIRST_BYTE);
    IF (CHARACTERISTIC = CONST_ZERO_BIN15) THEN
      DO;
           /* INPUT FIELD IS Ø, NOTHING TO SUM */
      END;
    ELSE
      DO;  /* INPUT FIELD IS NON-ZERO */
          CHARACTERISTIC = CHARACTERISTIC - 64; /* REMOVE EXCESS-64 */
             DO;  /* CONVERT FLOATING POINT SOURCE TO BIT STRING */
               TARGET_STRING = ''B;        /*         CLEAR TARGET */
               SOURCE_START_SUBSCRIPT = LENGTH (FIRST_BYTE) + 1;
               DIGITS_TO_MOVE =
                 MIN (CHARACTERISTIC, CONST_SIGNIF_DIGITS);
               NUM_BITS_TO_MOVE = (DIGITS_TO_MOVE)*4;
               TARGET_START_SUBSCRIPT =
                    ( (CONST_MAX_EXP - CHARACTERISTIC) * 4) + 1;
               SUBSTR
                (TARGET_STRING,
                 TARGET_START_SUBSCRIPT,NUM_BITS_TO_MOVE) =
               SUBSTR
                (SOURCE_STRING,
                 SOURCE_START_SUBSCRIPT,NUM_BITS_TO_MOVE);
               FLOAT_ACCUM = FLOAT_ACCUM + FLOAT(TARGET_STRING,64);
             END; /* CONVERT FLOATING POINT SOURCE TO BIT STRING */
            END;  /* INPUT IS NON-ZERO */
 END CONVERT_TO_FLOAT_AND_SUM;
%PAGE;
PROC_SUMM: PROC OPTIONS(REENTRANT) REORDER;
/* CLEAR MAP */
SUBSTR(ADDR(IXGMAP8O)->STR,1,STG(IXGMAP8O))=LOW(STG(IXGMAP8O));
EXEC CICS READQ TS QUEUE(TSQNAME) INTO(REAL_RECORD) ITEM(MAXITEM)
          RESP(XRESP);
MCURINTVO = MAXITEM;
MMAXINTVO = MAXITEM;
/* ********************************************** */
/*            PRODUCT SECTION                     */
SMF88SYNO = SMF88SYN;  /* MVS OPERATION SYSTEM NAME */
SMF88OSLO = SMF88OSL;  /* MVS RELEASE               */
/* ********************************************** */
/*           LOGSTREAM SECTION                    */
SMF88LSNO = SMF88LSN;  /* LOG STREAM NAME         */
SMF88LWIO = ACCUMULATOR(SMF88LWI_ACCUM_TABIDX);
SMF88LTDO = SUBSTR(SMF88PNM,1,4) || '/' ||    /* YYYY */
            SUBSTR(SMF88PNM,5,2) || '/' ||    /* MM   */
            SUBSTR(SMF88PNM,7,2) || SPACE ||  /* DD   */
            SUBSTR(SMF88LIT,1,2) || ':' ||  /* HH   */
            SUBSTR(SMF88LIT,3,2) || ':' ||  /* MM   */
            SUBSTR(SMF88LIT,5,2);           /* SS   */
SMF88LIBO = FLOAT(UNSPEC(SMF88LIB),32);  /* MIN.BLOCKLEN  */
SMF88LABO = FLOAT(UNSPEC(SMF88LAB),32);  /* MAX.BLOCKLEN  */
SMF88LWBO = ACCUMULATOR(SMF88LWB_ACCUM_TABIDX); /* BYT WRITTN SUM   */
SMF88STNO = SMF88STN;                           /* STRUCTURE  NAME  */
SMF88SWBO = ACCUMULATOR(SMF88SWB_ACCUM_TABIDX); /* BYT WRITTN INT.  */
SMF88LDBO = ACCUMULATOR(SMF88LDB_ACCUM_TABIDX); /* BYT WRITTN DASD  */
SMF88SIBO = ACCUMULATOR(SMF88SIB_ACCUM_TABIDX); /* BYT DEL W/O DASD */
SMF88SABO = ACCUMULATOR(SMF88SAB_ACCUM_TABIDX); /* BYT DEL W/DASD   */
SMF88SIIO = ACCUMULATOR(SMF88SII_ACCUM_TABIDX); /* #DEL W/O DASD    */
SMF88SAIO = ACCUMULATOR(SMF88SAI_ACCUM_TABIDX); /* #DEL W/WRITE     */
SMF88SC1O = ACCUMULATOR(SMF88SC1_ACCUM_TABIDX); /* #WRITES TYPE 1   */
SMF88SC2O = ACCUMULATOR(SMF88SC2_ACCUM_TABIDX); /* #WRITES TYPE 2   */
SMF88SC3O = ACCUMULATOR(SMF88SC3_ACCUM_TABIDX); /* #WRITES TYPE 3   */
SMF88EDSO = ACCUMULATOR(SMF88EDS_ACCUM_TABIDX); /* DASD SHRT        */
SMF88ERIO = ACCUMULATOR(SMF88ERI_ACCUM_TABIDX); /* REBLD INIT.      */
SMF88ERCO = ACCUMULATOR(SMF88ERC_ACCUM_TABIDX); /* REBLD CMP.       */
SMF88ESFO = ACCUMULATOR(SMF88ESF_ACCUM_TABIDX); /* STRC FULL        */
SMF88ETTO = ACCUMULATOR(SMF88ETT_ACCUM_TABIDX); /* STG THLD         */
SMF88ETFO = ACCUMULATOR(SMF88ETF_ACCUM_TABIDX); /* STG FULL         */
SMF88EOAO = ACCUMULATOR(SMF88EOA_ACCUM_TABIDX); /* OFFLOADS         */
SMF88EFSO = ACCUMULATOR(SMF88EFS_ACCUM_TABIDX); /* OFFL.9Ø%         */
SMF88EDOO = ACCUMULATOR(SMF88EDO_ACCUM_TABIDX); /* IXGOFFLD         */
PF24H = DFHREVRS;
CALL SEND_MAP;
RETURN;
END PROC_SUMM;
%PAGE;
PROC_AVG: PROC OPTIONS(REENTRANT) REORDER;
/* CLEAR MAP */
SUBSTR(ADDR(IXGMAP8O)->STR,1,STG(IXGMAP8O))=LOW(STG(IXGMAP8O));
EXEC CICS READQ TS QUEUE(TSQNAME) INTO(REAL_RECORD) ITEM(MAXITEM)
          RESP(XRESP);
MCURINTVO = MAXITEM;
MMAXINTVO = MAXITEM;
/* ************************************************ */
/*           PRODUCT SECTION                        */
SMF88SYNO = SMF88SYN;  /* MVS OPERATION SYSTEM NAME */
SMF88OSLO = SMF88OSL;  /* MVS RELEASE               */
/* ************************************************ */
/*           LOGSTREAM SECTION                      */
SMF88LSNO = SMF88LSN;  /* LOG STREAM NAME           */
SMF88LWIO = ACCUMULATOR(SMF88LWI_ACCUM_TABIDX) / MAXITEM;
```

```
SMF88LTDO = SUBSTR(SMF88PNM,1,4) || '/' ||    /* YYYY */
            SUBSTR(SMF88PNM,5,2) || '/' ||    /* MM   */
            SUBSTR(SMF88PNM,7,2) || SPACE ||  /* DD   */
            SUBSTR(SMF88LIT,1,2) || ':' ||  /* HH   */
            SUBSTR(SMF88LIT,3,2) || ':' ||  /* MM   */
            SUBSTR(SMF88LIT,5,2);           /* SS   */
SMF88LIBO = FLOAT(UNSPEC(SMF88LIB),32);  /* MIN.BLOCKLEN  */
SMF88LABO = FLOAT(UNSPEC(SMF88LAB),32);  /* MAX.BLOCKLEN  */
SMF88LWBO = ACCUMULATOR(SMF88LWB_ACCUM_TABIDX) / MAXITEM;
SMF88STNO = SMF88STN;
SMF88SWBO = ACCUMULATOR(SMF88SWB_ACCUM_TABIDX) / MAXITEM;
SMF88LDBO = ACCUMULATOR(SMF88LDB_ACCUM_TABIDX) / MAXITEM;
SMF88SIBO = ACCUMULATOR(SMF88SIB_ACCUM_TABIDX) / MAXITEM;
SMF88SABO = ACCUMULATOR(SMF88SAB_ACCUM_TABIDX) / MAXITEM;
SMF88SIIO = ACCUMULATOR(SMF88SII_ACCUM_TABIDX) / MAXITEM;
SMF88SAIO = ACCUMULATOR(SMF88SAI_ACCUM_TABIDX) / MAXITEM;
SMF88SC1O = ACCUMULATOR(SMF88SC1_ACCUM_TABIDX) / MAXITEM;
SMF88SC2O = ACCUMULATOR(SMF88SC2_ACCUM_TABIDX) / MAXITEM;
SMF88SC3O = ACCUMULATOR(SMF88SC3_ACCUM_TABIDX) / MAXITEM;
SMF88EDSO = ACCUMULATOR(SMF88EDS_ACCUM_TABIDX) / MAXITEM;
SMF88ERIO = ACCUMULATOR(SMF88ERI_ACCUM_TABIDX) / MAXITEM;
SMF88ERCO = ACCUMULATOR(SMF88ERC_ACCUM_TABIDX) / MAXITEM;
SMF88ESFO = ACCUMULATOR(SMF88ESF_ACCUM_TABIDX) / MAXITEM;
SMF88ETTO = ACCUMULATOR(SMF88ETT_ACCUM_TABIDX) / MAXITEM;
SMF88ETFO = ACCUMULATOR(SMF88ETF_ACCUM_TABIDX) / MAXITEM;
SMF88EOAO = ACCUMULATOR(SMF88EOA_ACCUM_TABIDX) / MAXITEM;
SMF88EFSO = ACCUMULATOR(SMF88EFS_ACCUM_TABIDX) / MAXITEM;
SMF88EDOO = ACCUMULATOR(SMF88EDO_ACCUM_TABIDX) / MAXITEM;
PF23H = DFHREVRS;
CALL SEND_MAP;
RETURN;
END PROC_AVG;
%PAGE;
CLEAR_MAP: PROC;
SUBSTR(ADDR(IXGMAP8O)->STR,1,STG(IXGMAP8O))=LOW(STG(IXGMAP8O));
RETURN;
END CLEAR_MAP;
%PAGE;
SEND_MAP: PROC;
EXEC CICS SEND MAP('IXGMAP8') MAPSET('IXGMAPS') FROM(IXGMAP8O)
          ALTERNATE ERASE RESP(XRESP);
RETURN;
END SEND_MAP;
END IXGSMF8;
```

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

Erhard Woerner
CICS Support Group
IBM (Germany)

CICS news

Landmark Systems has announced general availability of its new TMON for CICS/TS (VSE) monitoring tool, designed specifically to support CICS Transaction Server on VSE.

It contains the same functionality and features as its predecessor with, the company claims, a substantial reduction in overhead. Once the performance data is collected, it is transferred outside CICS for logging and analysis in order to avoid consuming critical system resources.

The package makes it possible to solve performance problems such as CICS lockouts, storage shortages, and poorly designed application code by providing the ability to monitor real-time transactions and resource utilization using either the internal CICS screens or a standalone batch partition.

It provides a centralized portal to collect and observe recent transaction activity and also provides a daily performance summary. Users can measure the performance impact of application or system changes within the CICS environment.

There's a high-level view of all CICS partitions and their resource utilization across VSE images, and users can navigate directly into any CICS or VSE platform. The product also collects historical statistics that can be used to analyse past problems and future trends.

For further information contact:
Landmark, 12700 Sunrise Valley Drive, Reston, VA 20191-5804, USA.
Tel: (703) 464 1300.URL: http://www.landmark.com/products/tmoncicsts.shtml.

* * *

IBM has announced Version 4.0 of its CICS Transaction Gateway (CTG), which includes support for the Java Developer's Toolkit (JDK) Version 1.3.

It also supports Linux on S/390 platforms and HP-UX 11.00 and has better support for Windows 2000, including a single new install package for NT and Windows 2000 and support for Windows 2000 Terminal Services.

Also, CICS Servers can now be accessed via TCP62 (except on OS/390) and there's additional EPI support, new Java sample programs, and RAS enhancements.

The product enables Java, JavaBean, C++, COM, and C applications to connect to CICS applications running on any CICS server. The CICS applications can be written in any supported language as LINKable programs or as 3270 transactions.

Version 4 enables gateway/server communication over TCP/IP, TCP62, SNA LU6.2, and memory-based protocols. It supports transactional interoperation where the invoking application may initiate a recoverable unit of work, which is coordinated with the actions of the target CICS application and the resources it accesses.

The new implementation of the TCP62 protocol is integrated with the base product. V4 also incorporates the major functions of V3 and CICS Universal Client V3.

For further information contact your local IBM representative.
URL: ttp://www.ibm.com/software/ts/cics.

* * *

xephon