In this issue

3   CWA transaction affinity issues
12  A pattern matching algorithm
18  Displaying CPU usage by TCB
30  CICS/MVS 2.1.2 to CICS/ESA 4.1 migration – part 2
34  CICS message suppression and re-routing
48  CICS news

© Xephon plc 1999
CICS Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: xephon@compuserve.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: 940 455 7050

Contributions
Articles published in CICS Update are paid for at the rate of £170 ($250) per 1000 words and £90 ($140) per 100 lines of code for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our Notes for Contributors.

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

Editor
Robert Burgess

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.

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.

© Xephon plc 1999. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.
CWA transaction affinity issues

Do you need more than one CWA per CICS AOR? If so this article will be of interest to you.

HISTORY
More and more large companies are merging their data centres – and therefore their CICS applications. If you have more than one application running in a CICS AOR, and each one is using a different mapping of the CWA, then you will run into problems.

SOLUTIONS
To provide a solution, you could run every application in its own AOR. Alternatively, you could set up one huge CWA, segmented into all possible CWA mappings, and re-compile all of your application code to point to the relevant CWA mapping. Neither of these solutions is particularly elegant.

My solution is relatively simple and allows for (virtually) any number of application CWAs. However, it does rely on strict transaction naming standards for each application and a means of relating the transaction to a particular CWA.

THE TECHNIQUE
The technique requires the following components:

- A system CWA to anchor the addresses of the application CWAs and the address of the translate table, and to store the CICS SYSID.
- A translate table to translate transaction to application CWA.
- An EXEC interface ‘out’ exit to process EXEC CICS ADDRESS CWA calls.
- A PLT program to allocate and anchor application CWA areas, to load and anchor the translate table, and to enable the EXEC interface ‘out’ exit.
SYSTEM CWA

You should set up a common system CWA to be used by all CICS regions. Then define a copy book for the system CWA containing addresses for each of the application CWAs.

The following code is an extract from the beginning of a typical system CWA DSECT:

```
CWADSECT DSECT
*     CWAUCWA1 DS F APPL CWA ADDRESS 1
*     CWAUCWA2 DS F APPL CWA ADDRESS 2
*     CWAUCWA3 DS F APPL CWA ADDRESS 3
*     CWACTRTA DS F ADDRESS OF CWA TRT.
*     CWASYSID DS CL4 CICS SYSID
```

..................
 ..................
 ..................
 ..................

Other fields in the CWA DSECT
 ..................
 ..................


TRANSLATE TABLE

You should set up a translate table to relate transactions to application CWAs.

The following code is an extract from a translate table:

```
GAC   CMBCRT RT TYPE=ENTRY,
     LOGIC=GAC,
     LOGICLEN=3
     SYSID=*,
     APCWANUM=1
*  
GA     CMBCRT RT TYPE=ENTRY,
     LOGIC=GA,
     LOGICLEN=2
     SYSID=TD1S,
     APCWANUM=3
*  
G      CMBCRT RT TYPE=ENTRY,
     LOGIC=G,
     LOGICLEN=1,
```
This translate table contains entries for the following:

- All transactions starting ‘GAC’ in any CICS region will use application CWA 1.
- All transactions starting with ‘GA’ running in a CICS region with SYSID TD1S will use application CWA 3.
- All transactions starting with ‘G’ in any CICS region will use application CWA 2.

MACRO TO GENERATE THE TRANSLATE TABLE

```
MACRO

CMBCRTT &LOGIC=, LOGICAL-ID X
     &SYSID=, SYS-ID OF CICS REGION X
     &APC WANUM=, USER CWA ADDR NUMBER X
     &LOGICLEN=, LENGTH OF LOGICAL ENTRY X
     &TYPE=.

.*
.* INITIAL/FINAL/ENTRY
.*
AIF ('&TYPE' EQ 'INITIAL').TRTINIT
AIF ('&TYPE' EQ 'FINAL').TRTFIN
AIF ('&TYPE' EQ 'ENTRY').TRT1
MNOTE 8,'PARM ERROR - TYPE IN ERROR'
MEXIT

.TR T1  ANOP
AIF ('&LOGIC' NE ' ').TRT2
MNOTE 8,'PARM ERROR - LOGIC MUST BE ENTERED'

.TR T2  ANOP
AIF ('&LOGICLEN' NE ' ').TRT3
MNOTE 8,'PARM ERROR - LOGIC ENTRY LENG MUST BE ENTERED'

.TR T3  ANOP
AIF ('&SYSID' NE ' ').TRT4
MNOTE 8,'PARM ERROR - SYSID MUST BE ENTERED'

.TR T4  ANOP
DC   CL4'&LOGIC'
DC F'&LOGICLEN'
DC   CL4'&SYSID'

.TR T5  ANOP
AIF ('&APC WANUM' NE ' ').TRT5A
```
LOAD VERSION OF THE TRANSLATE TABLE

When the translate table is assembled and linked using the above macro, the following Assembler code is generated to create the load version of the table:

```
DC      CL4'GAC'
DC      F'3'
DC      CL4'**'
DC      F'Ø'
DC      CL4'GA'
DC      F'2'
DC      CL4'TD1S'
DC      F'8'
DC      CL4'G'
```

PLT PROGRAM

A PLTPI program is needed to GETMAIN shared storage areas for each application CWA and anchor the address of each in the system CWA.

It will also load and anchor the translate table in the system CWA and enable the EXEC interface exit ZXEIOUT.

***********************************************************************
* PROGRAM NAME:  CWAPLTPG                                           *
* DESCRIPTION:  MAINLINE CODE THAT RUNS AT CICS                    *
*               INITIALIZATION TO GETMAIN                           *
*               APPLICATION CWA AREAS,                              *
*               LOAD A TRANSLATE TABLE AND                        *
*               ENABLE EXEC INTERFACE OUT EXIT                    *
***********************************************************************
* REGISTER AND OTHER EQUATES                                       *
********************************************************************************
REQU
CWAABAR   EQU 8
********************************************************************************
* CWA DSECT.                                                      *
********************************************************************************
CWADSECT
********************************************************************************
* WORKING STORAGE DEFINITIONS                                      *
********************************************************************************
DFHEISTG  DSECT
INITVAL   DS    X
WSMESS    DS    CL5Ø                 CSMT MESSAGE FIELD
********************************************************************************
* MAINLINE CODE                                                    *
********************************************************************************
CWAPLTPG DFHEIENT CODEREG=(11),DATAREG=(10),EIBREG=9
B     MAINSØØØ             BRANCH TO MAINLINE
DC    C'.','C'CICS 4.1'     SYSTEM-ID.
DC    C'.','C'CWAPLTPG'     PROGRAM SOURCE NAME
DC    C'.','C'V=Ø1,SML=Ø1'
DC    C'.','C'PLT FOR TRANSACTION/CWA AFFINITY'
DC 'C'. 'C'&SYSDATE' DATE ASSEMBLED.
DC 'C'. 'C'&SYSTIME' TIME ASSEMBLED.
************************************************************
* GET ADDRESS OF SYSTEM CWA
************************************************************
MAINS000 DS 0H
EXEC CICS ADDRESS CWA(CWABAR)
USING CWADSECT,CWABAR ADDRESS THE CWA.
************************************************************
* GET THE CICS SYSID AND PLUG INTO THE CWA FOR LATER
************************************************************
MAINS010 DS 0H
EXEC CICS ASSIGN SYSID(CWASYSID)
************************************************************
* GET AND ANCHOR THREE 4K USER CWA AREAS
************************************************************
MAINS010 DS 0H
MVI INITVAL,X'00'
EXEC CICS GETMAIN SET(R4) SHARED BELOW FLENGTH(4096) INITIMG(INITVAL).
ST R4,CWAUCWA1
EXEC CICS GETMAIN SET(R4) SHARED BELOW FLENGTH(4096) INITIMG(INITVAL).
ST R4,CWAUCWA2
EXEC CICS GETMAIN SET(R4) SHARED BELOW FLENGTH(4096) INITIMG(INITVAL).
ST R4,CWAUCWA3
************************************************************
* LOAD THE TRANSACTION AFFINITY TABLE AND ANCHOR ITS ADDRESS IN THE SYSTEM CWA.
************************************************************
MAINS040 DS 0H
EXEC CICS LOAD PROGRAM('CWACTRT') HOLD SET(R5)
ST R5,CWACTRTA
************************************************************
* ENABLE THE EXEC INTERFACE EXIT XEIOUT
************************************************************
EXEC CICS ENABLE PROGRAM('ZXEIOUT') EXIT('XEIOUT') START
MAINS100 DS 0H
MVC WSMESS(50),WDCMESS1
EXEC CICS WRITEQ TD QUEUE('CSMT') FROM(WSMESS)
MAINS999 DS 0H
EXEC CICS RETURN
************************************************************
* CONSTANTS USED IN THIS PROGRAM
************************************************************
WDCMESS1 DC CL80'PLTCWAPG-I01-TRANSACTION/CWA AFFINITYX 'NOW ENABED'
LTORG
END
EXEC INTERFACE EXIT CODE

TITLE 'ZXEIOUT - EXEC INTERFACE OUT EXIT'
*********************************************************************************************
* PROGRAM NAME: ZXEIOUT
*
* DESCRIPTION: THIS PROGRAM RUNS WHENEVER AN EXEC INTERFACE COMMAND HAS BEEN EXECUTED.
* IT IS USED TO TRAP ALL CALLS TO GET THE CWA ADDRESS
*********************************************************************************************

* AMENDMENT HISTORY
*
* AUTHOR: AAAAAAAAAAAAAAAAAAA
* DATE: DD/MM/YY
* DESCRIPTION: DDDDDDDDDDDDDDDDDDDDDDDDDDDDD
* IDENTIFIER: RRRR
*********************************************************************************************

* REGISTER EQUATES USED BY THIS PROGRAM
*********************************************************************************************
REQU
DFHEIBR EQU R10
*********************************************************************************************

* DSECTS FOR THIS PROGRAM
*********************************************************************************************
DFHUEXIT TYPE=EP,ID=XEIOUT
COPY DFHEIBLK
CWA DSECT CWA DSECT
*********************************************************************************************

* DSECTS FOR EXEC COMMAND INTERFACE PLIST
*********************************************************************************************
EIPLIST DSECT
EICOMND DS CL2
DS CL3
EIPARM1 DS CL1
EIPARM2 DS CL1
EIPARM3 DS CL1
EIPARM4 DS CL1
*********************************************************************************************

* CODE STARTS HERE
*********************************************************************************************
* SOME ENTRY REQUIREMENTS
*********************************************************************************************
ZXEIOUT CSECT
STM R14,R12,12(R13) SAVE REGISTERS IN RSA
LR R11,R15
USING ZXEIOUT,R11 ALLOCATE BASE REGISTER
*********************************************************************************************
* R1 POINTS AT THE PARAMETER LIST
* ON ENTRY TO THIS EXIT PROGRAM
* SO WE NEED TO MAP IT
*********************************************************************************************
USING DFHUEPAR,R1 AND MAP PARM LIST

*******************************
*        NOW BRANCH TO MAINLINE
******************************************************************************
B MAINS000 BYPASS EYE-CATCHER
******************************************************************************
* EYE CATCHER
******************************************************************************

DC C'.','C'CICS4.1.0' SYSTEM-ID.
DC C'.','C'ZXEIOUT' PROGRAM-ID.
DC C'.','C'V=01, ML=00' PROGRAM VERSION.
DC C'.','C'EXEC INTERFACE OUT EXIT'
DC C'.','C'S. HIGGINS' WRITTEN BY.
DC C'.','C'&SYSDATE' DATE ASSEMBLED.
DC C'.','C'&SYSTIME' TIME ASSEMBLED.
DC C'.',' END OF EYE-CATCHER

******************************************************************************
* GET SUPPLIED EIB
* AND CHECK FOR EXEC CICS ADDRESS COMMAND
******************************************************************************

MAINS000 DS $H
L R10,UEPEXEC8
CLC EIBFN,=X'0202'
BNE MAINS999

******************************************************************************
* GET EXEC INTERFACE PARAMETER LIST
* FOR AN 'EXEC CICS ADDRESS'
******************************************************************************

MAINS100 DS $H
L R9,UEPARG LOAD ADDRESS OF PLIST
L R8,0(R9) LOAD 1ST ADDR
USING EIPLIST,R8 AND MAP WITH DSECT
CLC EIPARM1,=X'02' IS PARM1 AN ADDR CWA
BNE MAINS120 NO - TRY NEXT PARM
L R7,4(R9) YES - GET ADDR OF CWA ADDR
B MAINS200 AND PROCESS IT.

MAINS120 DS $H
CLC EIPARM2,=X'02' IS PARM2 AN ADDR CWA
BNE MAINS140 NO - TRY NEXT PARM
L R7,8(R9) YES - GET ADDR OF CWA ADDR
B MAINS200 AND PROCESS IT.

MAINS140 DS $H
CLC EIPARM3,=X'02' IS PARM3 AN ADDR CWA
BNE MAINS160 NO - TRY NEXT PARM
L R7,12(R9) YES - GET ADDR OF CWA ADDR
B MAINS200 AND PROCESS IT.

MAINS160 DS $H
CLC EIPARM4,=X'02' IS PARM4 AN ADDR CWA
BNE MAINS999 NO - EXIT
L R7,16(R9) YES - GET ADDR OF CWA ADDR
**PROCESS CWA ADDRESS**

**1ST PROCESS THE CWA TRT TABLE TO GET THE APPLICATIONS CWA.**

```
MAIN200  DS  $0H
  L  R6,0(,R7)  GET SYSTEM CWA ADDR
  USING CMBCWA,R6  AND MAP IT
  ICM  R8,15,CWACTRTA  ANY CWA TRT ADDRESS ?
  BZ  MAINS999  NO EXIT

MAIN220  DS  $0H
  CLC  0(4,R8),EIBTRNID  GONE PAST ENTRY?
  BH  MAINS999  YES - DEFAULT TO SYSTEM CWA
  L  R5,4(,R8)  NO - GET LOGICAL ENTRY LENGTH
  BCTR  R5,0  DECREMENT LENGTH FOR CLC
  EX  R5,COMPARE  COMPARE ENTRY WITH TRANID
  BNE  MAINS240  YES - GO AND PROCESS
  CLI  8(R8),C'**'  ANY SYSID SPECIFIED
  BE  MAINS300  NO - USE DEFAULT
  CLC  8(4,R8),CWASYSID  YES - SYSIDS MATCH?
  BE  MAINS300  YES - GO PROCESS
  B  MAINS999  NO - USE DEFAULT

MAIN240  DS  $0H
  LA  R8,16(,R8)  NO - BUMP UP TO NEXT ENTRY
  B  MAINS220  AND CHECK IT OUT

MAIN300  DS  $0H
  L  R3,12(,R8)  GET USER CWA NUMBER ( $0,4,8)
  LA  R9,CWAUCWA1  GET USER CWA ADDR 1
  LA  R9,0(R3,R9)  BUMP UP TO OUR CWA ADDR
  MVC  0(4,R7),0(R9)  MOVE INTO SUPPLIED CWA ADDR

**MAINLINE CODE EXIT**

```

```
MAIN999  DS  $0H
  L  R13,UEPEPSA  POINT REG 13 AT RSA
  LM  R14,R12,12(R13)
  BR  R14

**END OF MAINLINE**

```

**COMPARE**

```
CLC  0(0,R8),EIBTRNID  DOES TRAN-ID MATCH THIS ENTRY
  LTORG
  END
```

_________

Simon Higgins
Blackbox Design Services (UK) © Xephon 1999
A pattern matching algorithm

The EXCI interface, introduced with CICS Version 4, gives access to CICS resources that were previously unavailable. We use EXCI to send CEMT commands from batch jobs to CICS to change CICS resources, eg open/close files.

Unfortunately, IBM revoked the usage of the CEMT programming interface at the same time that it introduced EXCI. So we had to write our own CEMT program, accessible from a batch program via EXCI, returning the CICS RESP value as a return-code to the batch job.

It is not difficult to write a ‘user’ CEMT program to vary a single resource. You must analyse the incoming command string and build the appropriate SPI command. For example, if the CICS server program receives the string:

```plaintext
CEMT SET PROGRAM(KO767A) PHASEIN
```

the program must create the corresponding SPI command:

```plaintext
EXEC CICS SET PROGRAM(KO767A) PHASEIN
```

It is more difficult to vary families of resources using the wildcard symbols ‘*’ and ‘+’, where ‘*’ represents any number of character, including none, and ‘+’ represents a single character. For example:

- ‘KO*’ – all identifiers beginning with KO.
- ‘*767*’ – all identifiers containing the characters 767.
- ‘A+’ – all 2-character identifiers starting with A.
- ‘++A*’ – all identifiers with A as the third character.

To allow the use of such generic resource names with the same syntax as the CICS CEMT transaction, I have written a little routine in COBOL that determines whether the resource name matches the search pattern.

The program is called with three parameters – the resource name, the search pattern, and a result flag. After the call, the result flag must be tested. The flag returns a high-value if the resource name matches the search pattern and a low-value if it does not.
For example, to NEWCOPY all programs starting with ‘KO’, you must code the following loop in your main program:

```
1 PGM-NAME PIC X(8).
1 PATTERN PIC X(8).
1 RESULT PIC X.
   88 MATCH VALUE HIGH-VALUE.
   88 NOMATCH VALUE LOW-VALUE.
.
MOVE 'KO*' TO PATTERN
EXEC CICS INQUIRE PROGRAM START
END-EXEC
EXEC CICS INQUIRE PROGRAM (PGM-NAME) NEXT RESP(RC)
END-EXEC
*
PERFORM UNTIL RC = DFHRESP(END)
   CALL 'GENERIC' USING PGM-NAME
      PATTERN
      RESULT
END-CALL
   IF MATCH
   THEN
      EXEC CICS SET PROGRAM (PGM-NAME) PHASEIN
      END-EXEC
   END-IF
   EXEC CICS INQUIRE PROGRAM (PGM-NAME) NEXT RESP(RC)
   END-EXEC
END-PERFORM
*
EXEC CICS INQUIRE PROGRAM END
END-EXEC
```

The subroutine consists logically of two COBOL programs, GENERIC and SUCHE. It should work in every CICS version that allows the SPI commands ‘inquire program’ and ‘set program’; the subroutines have no CICS statements.

**GENERIC**

```cobol
IDENTIFICATION DIVISION.
PROGRAM-ID. GENERIC.
DATA DIVISION.
WORKING-STORAGE SECTION.
  1 TT.
  2 T PIC X(1) OCCURS 8 TIMES.
```
1 ETAB.
  2 AE PIC 9(2).
  2 XE OCCURS 8 TIMES.
      3 EL PIC 9(2).
      3 ED PIC X(8).
1 X.
  2 XL PIC 9(2).
  2 XD PIC X(8).
  2 XT PIC X(1) OCCURS 8 TIMES REDEFINES XD.
1 Y.
  2 YL PIC 9(2).
  2 YD PIC X(8).
  2 YT PIC X(1) OCCURS 8 TIMES REDEFINES YD.
* O = OFFSET
  1 O PIC S9(4) BINARY VALUE ZERO.
  1 I PIC 9(4) VALUE ZERO.
  1 J PIC 9(4) VALUE ZERO.
  1 K PIC 9(4) VALUE ZERO.
  1 L PIC 9(4) VALUE ZERO.
  1 S PIC 9(4) VALUE ZERO.
  1 Z PIC 9(8) BINARY VALUE ZERO.
LINKAGE SECTION.
* C = CHARACTER STRING; P = PATTERN; E = RESULT FLAG
  1 C PIC X(8).
  1 P PIC X(8).
  1 E PIC X(1).
  88 MATCH VALUE HIGH-VALUE.
  88 NOMATCH VALUE LOW-VALUE.
PROCEDURE DIVISION USING C P E.
*    SPACES AS SEARCH PATTERN ARE NOT VALID
  IF P = SPACES
      THEN
      SET NOMATCH TO TRUE
      GOBACK
  END-IF
*  SET MATCH TO TRUE
  INITIALIZE ETAB
*  SPLIT SEARCH PATTERN INTO SUBSTRINGS
  UNSTRING P DELIMITED ALL '*' OR ALL ' ' INTO
      ED OF XE(1) COUNT EL OF XE(1)
      ED OF XE(2) COUNT EL OF XE(2)
      ED OF XE(3) COUNT EL OF XE(3)
      ED OF XE(4) COUNT EL OF XE(4)
      ED OF XE(5) COUNT EL OF XE(5)
      ED OF XE(6) COUNT EL OF XE(6)
      ED OF XE(7) COUNT EL OF XE(7)
      ED OF XE(8) COUNT EL OF XE(8)
  TALLYING AE
  END-UNSTRING
  IF P(8:1) = '*'
THEN
ADD 1 TO AE
END-ADD
END-IF
* MOVE PROGRAM-NAME TO LINKAGE FIELD X (WITH LENGTH)
  MOVE C TO XD
  MOVE ZERO TO XL
  INSPECT XD TALLYING XL FOR CHARACTERS BEFORE SPACE
  MOVE LOW-VALUES TO TT
* STARTING MATCH
  MOVE 1 TO Z
  SET MATCH TO TRUE
*
PERFORM VARYING I FROM 1 BY 1 UNTIL I > AE OR NOMATCH
  MOVE ED OF XE(I) TO YD
  MOVE EL OF XE(I) TO YL
  CALL 'SUCHE' USING X Y Z E O
  END-CALL
  IF MATCH
    THEN
      IF (I = AE AND YL > Ø)
        THEN
          COMPUTE S = XL - YL + 1
          END-COMPUTE
          MOVE HIGH-VALUE TO T(I)
          PERFORM VARYING J FROM 1 BY 1 UNTIL J > YL
          IF YT(J) = '+' OR XT(J) = YT(J)
            THEN
              CONTINUE
            ELSE
              Move LOW-VALUE TO T(I)
          END-IF
          ADD 1 TO S
          END-ADD
      END-PERFORM
      ELSE
        IF (I = 1 AND YL > Ø)
          THEN
            MOVE HIGH-VALUE TO T(I)
            PERFORM VARYING J FROM 1 BY 1 UNTIL J > YL
            IF YT(J) = '+'
              OR XT(J) = YT(J)
              THEN
                CONTINUE
              ELSE
                Move LOW-VALUE TO T(I)
            END-IF
            ADD 1 TO S
            END-ADD
        END-PERFORM
        ELSE
        END-ELSE
      ELSE
      END-ELSE
    END-IF
MOVE HIGH-VALUE TO T(I)
COMPUTE Z = Z + O + YL
END-COMPUTE
END-IF
END-IF
END-PERFORM
* END MATCH
SET MATCH TO TRUE
PERFORM VARYING I FROM 1 BY 1 UNTIL I > AE OR NOMATCH
IF T(I) = LOW-VALUE
THEN
SET NOMATCH TO TRUE
END-IF
END-PERFORM
*
GOBACK.
END PROGRAM GENERIC.

SUCHE

IDENTIFICATION DIVISION.
PROGRAM-ID. SUCHE.
DATA DIVISION.
WORKING-STORAGE SECTION.
1 I    PIC S9(4) BINARY.
1 J    PIC S9(4) BINARY.
1 K    PIC S9(4) BINARY.
1 XI   PIC S9(4) BINARY.
1 YI   PIC S9(4) BINARY.
1 TT.
   2 T PIC X(1) OCCURS 8 TIMES.
LINKAGE SECTION.
* X = CHARACTER STRING
* Y = PATTERN
1 X.
   2 XL PIC 9(2).
   2 XD PIC X(1) OCCURS 8 TIMES.
1 Y.
   2 YL PIC 9(2).
   2 YD PIC X(1) OCCURS 8 TIMES.
* Z = POINTER IN CHARACTER STRING
* E = RESULT FLAG
* O = OFFSET IF FOUND
1 Z    PIC 9(8) BINARY.
1 E    PIC X.
   88 FOUND VALUE HIGH-VALUE.
   88 NOT-FOUND VALUE LOW-VALUE.
1 O    PIC S9(4) BINARY.
PROCEDURE DIVISION USING X Y Z E O.
* YL = Ø MEANS '**

  IF YL = Ø
    THEN
      SET FOUND TO TRUE
      MOVE +Ø TO O
      GOBACK
    END-IF

  SET NOT-FOUND TO TRUE
  MOVE -1 TO O

  IF ((XL - Z + 1) < YL) OR (XL > 8) OR (YL > 8)
  THEN
    GOBACK
  END-IF

  PERFORM VARYING I FROM Z BY 1 UNTIL ((I + YL - 1) > XL)
  OR FOUND
  ADD 1 TO O
  END-ADD
  PERFORM VARYING J FROM 1 BY 1
  UNTIL J > YL OR FOUND
  MOVE LOW-VALUES TO TT
  PERFORM VARYING K FROM Ø BY 1
  UNTIL K = YL
  COMPUTE XI = K + I
  END-COMPUTE
  COMPUTE YI = K + 1
  END-COMPUTE
  IF YD(YI) = '+' OR YD(YI) = XD(XI)
  THEN
    MOVE HIGH-VALUE TO T(YI)
  END-IF
  END-PERFORM
  SET FOUND TO TRUE
  PERFORM VARYING K FROM 1 BY 1
  UNTIL K > YL OR NOT-FOUND
  IF T(K) = LOW-VALUE
  THEN
    SET NOT-FOUND TO TRUE
  END-IF
  END-PERFORM
  END-PERFORM
  END-PERFORM

  GOBACK.
  END PROGRAM SUCHE.
Displaying CPU usage by TCB

When analysing the CPU usage of our CICS regions for a capacity planning exercise, we compared the real CPU usage by the CICS address space (from SMF and RMF data sources) with ‘transaction level’ CPU data (provided by CMF and DB2 accounting). We noticed that there was quite a difference between these two sources (even with capture ratios applied to the CMF and DB2 data). Therefore, there must be something else consuming CPU in our CICS.

By using an on-line MVS monitor (OMEGAMON in our case), we noticed that the CICS address space had a lot of TCBs. We wanted to know how the total CPU consumed by the CICS address space was distributed across these multiple TCBs. This could help us relate CICS CMF data and DB2 accounting data to the total CICS RMF or SMF data, and thus provide a better capacity plan for our CICS regions.

The DTCB transaction consists of a map (IPPDTCB), a COBOL program (IPPCDTCB), and an Assembler subroutine (IPPCGTCB). The Assembler subroutine collects the CPU usage by TCB for the address space in which it is running. In fact, as the routine is not using any CICS services, IPPCGTCB may be used in any address space. For example, we also used it to see CPU usage by TCB in an IDMS region.

The Assembler routine collects CPU usage by TCB for the address space it is running in:

```
IPPCGTCB TITLE 'IPPCGTCB - GET CPUTIMES BY TCB'
    SPACE 2
IPPCGTCB AMODE 31
IPPCGTCB RMODE ANY
    SPACE 2
**********************************************************************
*        THIS PROGRAM IS CALLED VIA STANDARD LINKAGE : (COBOL EXAMPLE)
*        CALL 'IPPCGTCB' USING ASCBREC - MAPPED BY ASCBREC DSECT
*                        TCBAREC.- MAPPED BY TCBAREC DSECT
*        ONLY ASCBMAX IS NEEDED AS INPUT ... ALL OTHER FIELDS ARE OUTPUT
*        FROM THIS ROUTINE ...
*        OPM : - PASSED RECORDS NEED NOT BE FW ALIGNED ...
```
* - RETURN VALUE Ø : ALL OK
* - RETURN VALUE 4 : MORE TCBS UNDER THIS ASCB THAN ALLOWED
* - RETURN VALUE 8 : SOMETHING BAD HAPPENED

**********************************************************************
EJECT
RØ   EQU   Ø
R1    EQU   1
R2    EQU   2
R3    EQU   3
R4    EQU   4
R5    EQU   5
R6    EQU   6
R7    EQU   7
R8    EQU   8
R9    EQU   9
R10   EQU   10
R11   EQU   11
R12   EQU   12
R13   EQU   13
R14   EQU   14
R15   EQU   15
RPARAM EQU   3
RCNT   EQU   3
RASCOREC EQU   4
RTCBAREC EQU   5
RPSA    EQU   6
RASCHE EQU   6
RASXB   EQU   6
RTCB    EQU   7
RRB     EQU   8
RCDE    EQU   9
RW      EQU   10
RW2     EQU   11
EJECT

* SAVE REGISTERS AND OBTAIN PARAMETERS PASSED

IPPCGTCB CSECT ,
SAVE (14,12),,IPPCGTCB.&SYSDATE..&SYSTIME SAVE REGISTERS
LR    R12,R15 COPY BASE REGISTER
USING IPPCGTCB,R12 ESTABLISH ADDRESSABILITY
LR    RPARAM,R1 POINT TO PARAMETER LIST ...
XR    R15,R15 CLEAR RETURN CODE ...
USING PARAMDS,RPARAM ADDRESS THE PARAMETER ADDRESSES
L     RASCHE,RASCHEPTR LOAD POINTER TO ASCB RECORD
USING ASCBREC,RASCHE PTR ADDRESS THE ASCBREC
L     RTCBAREC,TCBAPTR LOAD POINTER TO TCB ARRAY
DROP  RPARAM
TIME  BIN
STCM  RØ,B'1111',ASCBTIM SAVE THE TIME REQUEST WAS EXECUTED

USING TCBAREC,RTCBAREC ADDRESS THE TCBAREC
XR RPSA,RPSA PSA FROM LOC X'ØØØ' TO X'FFF'
USING PSA,RPSA ADDRESS THE PSA
L RASCB,PSAAOLD CURRENT ASCB ADDRESS
DROP RPSA
USING ASCB,RASCB ADDRESS THE ASCB

* —— PROCESS ASCB DATA :  ———
*    TCB CPU AT ADDRESS SPACE LEVEL
*    SRB CPU TIME AT ADDRESS SPACE LEVEL
* ——

* - TCB TIME
LM RW,RW2,ASCBEJST LOAD ASCB TCB CPU TIME
SRDL RW,12 DIVIDE BY 4096
D RW,=F'1000' AND BY 1000 - CPU TIME IN MILLISECS
STCM RW2,B'1111',ASCBTCB SAVE IN ASCBREC

* - SRB TIME
LM RW,RW2,ASCBSRBT LOAD ASCB SRB CPU TIME
SRDL RW,12 DIVIDE BY 4096
D RW,=F'1000' AND BY 1000 - CPU TIME IN MILLISECS
STCM RW2,B'1111',ASCBSRB SAVE IN ASCBREC

* - TOT # OF TCBS
L RASXB,ASCBASXB ACCESS THE ASXB CONTROL BLOCK
DROP RASCB
USING ASXB,RASXB ADDRESS THE ASXB
L RTCB,ASXBFTCB LOAD THE FIRST TCB
LH RCNT,ASXBTCBS # OF TCBS IN THIS ASCB
ICM RW,B'1111',ASCBMAX LOAD MAX # OF ENTRIES TO BE RETURNED
CR RCNT,RW MORE TCBS THAN MAX
BNH SIZEOK
LA R15,4 INDICATE TRUNCATION OF DATA
LR RCNT,RW PREVENT OVERFLOW
SIZEOK STCM RCNT,B'1111',ASCBSR#TCB SAVE IN ASCBREC
DROP RASXB
USING TCB,RTCB ADDRESS THE TCB STRUCTURE

* —— PROCESS TCB DATA ———

XR RW,RW
ST RW,ASCBSUM CLEAR SUMMARY FIELD
XR RPSA,RPSA RE-ADDRESS PSA
USING PSA,RPSA ADDRESS THE PSA
TCBLOOP LTR RCNT,RCNT TILL ALL PROCESSED
BZ ALLPROC
LTR RTCB,RTCB AND NOT END-OF-CHAIN IN THE TCBS
BZ ALLPROC

* - ADDRESS OF THE TCB - RETURN IT IN READABLE FORM
ZAP WFIELD,=P'Ø' INIT THE WFIELD
STCM RTCB,B'1111',WFIELD LOAD THE BINARY ADDRESS
UNPK OFIELD(9),WFIELD CONVERT HEX TO READABLE
MVI OFIELD+8,C' ' ...
TR OFIELD(8),TABHEX AND MAKE UGLY ONES READABLE
MVC TCBADDR,OFIELD SAVE ADDRESS IN TCBAREC
STCM RTCB,B'1111',TCBATCB KEEP ADDRESS FOR THE SORT

* - ADDRESS OF THE TCB
MVI TCBFLAGS,C' ' RE-INIT THE FLAGS
MVC TCBFLAGS+1(L'TCBFLAGS-1),TCBFLAGS+1

* - TCB TIME
LM RW,RW2,TCBTTIME LOAD TCB CPU TIME
SRDL RW,12 DIVIDE BY 4906
D RW,=F'1000' AND BY 1000 - CPU TIME IN MILLISECS
STCM RW2,B'1111',TCBCPUT SAVE IN TCBAREC
ICM RW,B'1111',ASCBSUM LOAD PREVIOUS SUM FROM ASCBREC
AR RW,RW2 ADD CURRENT
STCM RW,B'1111',ASCBSUM SAVE NEW SUM IN ASCBREC

* - FLAG BYTES
C RTCB,PSATOLD IS THIS THE CURRENT TCB
BNE CHKACTI
MVI TCBFLAGS,C'**' INDICATE CURRENT TCB

CHKACTI TM TCBXSCT1,TCBACTIV IS THIS TCB ACTIVE ON A CPU ?
BNO NOTACTI
MVI TCBFLAGS+1,C'**' INDICATE ACTIVE ON A CPU

* - PROGRAM BEING RUN UNDER THE TCB
NOTACTI LA RW,255 PREVENT LOOP
L RRB,TCHRBP CURRENT REQUEST BLOCK
USING RBBASIC,RRB ADDRESS THE RB STRUCTURE
L RW2,RBLINK LOAD THE RBLINK ADDRESS
SLL RW2,8 RB'S ARE 24-BIT ADDRESSES SO ..
SRL RW2,8 CLEAR HIGH ORDER BYTE

PGMLOOP LTR RW,RW PREVENT LOOP
BZ ENDLOOP
CR RTCB,RW2 IS THIS THE ONE
BE ENDLOOP
LR RRB,RW2 LOAD PREVIOUS REQUEST BLOCK
L RW2,RBLINK LOAD THE NEXT RBLINK ADDRESS
SLL RW2,8 RB'S ARE 24-BIT ADDRESSES SO ..
SRL RW2,8 CLEAR HIGH ORDER BYTE
BCTR RW,RØ MINUS ONE
B PGMLOOP
ENDLOOP CR RTCB,RW2 DID WE HAVE A HIT ...
BNE NOHIT
L RCDE,RBCDE CURRENT CONTENTS DIRECTORY ENTRY
USING CDENTRY,RCDE ADDRESS THE CDE
MVC TCBPROG,CDNAME PROGRAM RUNNING UNDER THIS TCB

NOHIT LA RTCBAREC,TCBLEN(RTCBAREC) NEXT IN TCB ARRAY
L RTCB,TCBTCB ADDRESS NEXT TCB
BCTR RCNT,RØ ONE MORE PROCESSED
B TCBLOOP

ALLPROC DS ØH
DROP RPSA
RETURN (14,12),RC=(15) RETURN TO CALLING PROGRAM
SPACE 2
Now we have collected the CPU data, let’s show the usage by TCB.
The DTCB transaction invokes the COBOL program IPPCDTCB. IPPCDTCB calls the IPPCGTCB routine, and displays the data on the IPPDTCB map. The IPPDTCB map produces the screen layout shown in Figure 1.
<table>
<thead>
<tr>
<th>PROGRAM</th>
<th>ADDRESS</th>
<th>CPU TIME</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHKETCB</td>
<td>Ø0ADB2EØ</td>
<td>1112.35</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ACF95UB</td>
<td>Ø0ADB68</td>
<td>510.79</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DSN2EXT3</td>
<td>Ø0AES318</td>
<td>327.93</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DSN2EXT3</td>
<td>Ø0AAC3Ø</td>
<td>61.73</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DFH5SERV</td>
<td>Ø0ABB75Ø</td>
<td>28.72</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DFHKETCB</td>
<td>Ø0ADB2FB8</td>
<td>8.20</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DSN2EXT3</td>
<td>Ø0AAS2Ø</td>
<td>7.49</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>E2ØSTSK</td>
<td>Ø0AB23Ø</td>
<td>3.96</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DSN2EXT3</td>
<td>Ø0AAS6Ø</td>
<td>1.17</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>IEE5B6Ø</td>
<td>Ø0AFDØ</td>
<td>0.49</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DFH5IP</td>
<td>Ø0AF94Ø</td>
<td>0.34</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DFH5KMR</td>
<td>Ø0AB66Ø</td>
<td>0.27</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DFH5KTSK</td>
<td>Ø0ADB3CBC</td>
<td>0.26</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DFH@SERV</td>
<td>Ø0AB4CØ</td>
<td>0.22</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ACF95UB</td>
<td>Ø0AAB88</td>
<td>0.12</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

COMMAND ===>
F3-END       F5-DISP-E
F6-MODE-D    F8-FORWARD

**Figure 1: Screen layout**

You can sort the display by issuing the SORT command, followed by PROGRAM, ADDRESS, or CPUTIME. Default SORT order is by CPUTIME. The NOSORT command displays the data in the sequence delivered by the IPPGTB program. PF key 6 allows you to toggle the display between delta mode (CPU usage between two samples), or total mode (CPU usage since CICS start-up). When using PF key 5 you can alter the graphical representation:

- **MODE-C** shows the bars as TCB CPU compared with the total CPU used.
- **MODE-E** shows the bars as TCB CPU compared with the elapsed time of the CICS started task.

If you run the DTCB transaction on a colour-capable terminal (which I suppose most people do now), the active TCB (the one running the IPPCGGTB program) will be coloured red. Other TCBs that are also
active on a CPU are coloured yellow. If your CICS region is pretty busy, you’ll notice that often more than one TCB in the CICS address space is actually using the CPU.

The ASCB-TCB/SRB field contains the TCB and SRB time for the address space, as collected in the ASCB control block. The SUM field is the sum of CPU time for all TCBs. Normally these values should be quite close to each other. Differences may exist if TCBs are created and destroyed in the CICS address space.

As the IPPCGTCB program accesses MVS control blocks, it must be assembled with the current versions of your MVS system. We are currently running this with CICS Version 4.1 and OS/390 Version 1 Release 2.

In our case, we could split up the TCBs in three large parts:

- Real CICS TCBs (DFHKETCB, DFHSIP, DFHKSMGR, etc)
- The DB2 attachment TCBs (DSN2EXT3)
- Some external stuff (ACF99SUB, E20XSTSK, etc).

We saw that a considerable amount of CPU in our production CICS was consumed by the ACFF9SUB TCB. This is CPU used by our security product.

The CICS map source follows:

```
IPPDTCB DFHMSD TYPE=&SYSPARM,MODE=INOUT,LANG=COBOL, DATA=FIELD,TERM=3270,TIOAPFX=YES,STORAGE=AUTO,
            MAPATTS=(COLOR,HILIGHT),DSATTS=(COLOR,HILIGHT)

* HEADER LINE 1
TCBPROG DFHMDF POS=(01,01),LENGTH=08,ATTRB=(ASKIP)
TCBTIT1 DFHMDF POS=(01,10),LENGTH=40,ATTRB=(ASKIP),COLOR=NEUTRAL
TCBNETN DFHMDF POS=(01,52),LENGTH=08,ATTRB=(ASKIP)
TCBAPPL DFHMDF POS=(01,61),LENGTH=08,ATTRB=(ASKIP)
TCBDATE DFHMDF POS=(01,70),LENGTH=10,ATTRB=(ASKIP)
* HEADER LINE 2
TCBMAPN DFHMDF POS=(02,01),LENGTH=08,ATTRB=(ASKIP)
TCBTIT2 DFHMDF POS=(02,10),LENGTH=40,ATTRB=(ASKIP),COLOR=NEUTRAL
TCBTERM DFHMDF POS=(02,52),LENGTH=04,ATTRB=(ASKIP)
TCBUSER DFHMDF POS=(02,61),LENGTH=08,ATTRB=(ASKIP)
TCBTIME DFHMDF POS=(02,70),LENGTH=10,ATTRB=(ASKIP)
* THE DATA PORTION ...
```
* THE TITLE ...  
DFHMDF POS=(Ø4,Ø1),LENGTH=79,ATTRB=(ASKIP,BRT), INITIAL="PROGRAM ADDRESS CPU TIME PCT1 20 30 X 40 50 60 70 80 90 100",COLOR=PINK
* 1234567890123456789012345678901234567890
DFHMDF POS=(Ø5,Ø1),LENGTH=79,ATTRB=(ASKIP,BRT), INITIAL="______________________________",COLOR=PINK
* THE DATA LINES ...
TCBROW1 DFHMDF POS=(Ø6,Ø1),LENGTH=79,ATTRB=(ASKIP)
TCBROW2 DFHMDF POS=(Ø7,Ø1),LENGTH=79,ATTRB=(ASKIP)
TCBROW3 DFHMDF POS=(Ø8,Ø1),LENGTH=79,ATTRB=(ASKIP)
TCBROW4 DFHMDF POS=(Ø9,Ø1),LENGTH=79,ATTRB=(ASKIP)
TCBROW5 DFHMDF POS=(1Ø,Ø1),LENGTH=79,ATTRB=(ASKIP)
TCBROW6 DFHMDF POS=(11,Ø1),LENGTH=79,ATTRB=(ASKIP)
TCBROW7 DFHMDF POS=(12,Ø1),LENGTH=79,ATTRB=(ASKIP)
TCBROW8 DFHMDF POS=(13,Ø1),LENGTH=79,ATTRB=(ASKIP)
TCBROW9 DFHMDF POS=(14,Ø1),LENGTH=79,ATTRB=(ASKIP)
TCBROWA DFHMDF POS=(15,Ø1),LENGTH=79,ATTRB=(ASKIP)
TCBROWB DFHMDF POS=(16,Ø1),LENGTH=79,ATTRB=(ASKIP)
TCBROWC DFHMDF POS=(17,Ø1),LENGTH=79,ATTRB=(ASKIP)
TCBROWD DFHMDF POS=(18,Ø1),LENGTH=79,ATTRB=(ASKIP)
TCBROWE DFHMDF POS=(19,Ø1),LENGTH=79,ATTRB=(ASKIP)
TCBROWF DFHMDF POS=(20,Ø1),LENGTH=79,ATTRB=(ASKIP)
* MESSAGE LINE 21
TCBMESS DFHMDF POS=(21,Ø1),LENGTH=3Ø,ATTRB=(ASKIP,BRT),INITIAL=' ',COLOR=RED
TCBACT1 DFHMDF POS=(21,32),LENGTH=48,ATTRB=(ASKIP,BRT),INITIAL=' ',COLOR=RED
* COMMAND LINE 22
DFHMDF POS=(22,Ø1),LENGTH=12,ATTRB=(ASKIP), INITIAL='COMMAND ===>'
TCBCOMM DFHMDF POS=(22,14),LENGTH=4Ø,ATTRB=(BRT,UNPROT,IC)
* TCBPFS LINE 23
TCBPFS01 DFHMDF POS=(23,Ø1),LENGTH=12,ATTRB=(ASKIP)
TCBPFS03 DFHMDF POS=(23,14),LENGTH=12,ATTRB=(ASKIP)
TCBPFS05 DFHMDF POS=(23,27),LENGTH=12,ATTRB=(ASKIP)
TCBPFS07 DFHMDF POS=(23,4Ø),LENGTH=12,ATTRB=(ASKIP)
TCBPFS09 DFHMDF POS=(23,53),LENGTH=12,ATTRB=(ASKIP)
TCBPFS11 DFHMDF POS=(23,66),LENGTH=12,ATTRB=(ASKIP)
* TCBPFS LINE 24
TCBPFS02 DFHMDF POS=(24,Ø1),LENGTH=12,ATTRB=(ASKIP)
TCBPFS04 DFHMDF POS=(24,14),LENGTH=12,ATTRB=(ASKIP)
TCBPFS06 DFHMDF POS=(24,27),LENGTH=12,ATTRB=(ASKIP)
TCBPFS08 DFHMDF POS=(24,4Ø),LENGTH=12,ATTRB=(ASKIP)
TCBPFS10 DFHMDF POS=(24,53),LENGTH=12,ATTRB=(ASKIP)
TCBPFS12 DFHMDF POS=(24,66),LENGTH=12,ATTRB=(ASKIP)
* DFHMSD TYPE=FINAL
END

Now there is one thing left, the IPPCDTCB COBOL program displaying the data, collected by the IPPCGTCB Assembler program, on the map. We limit the total number of TCBs to 255 for our CICS region. (This seems to be a reasonable number to me!)

IDENTIFICATION DIVISION.
*-------------------------------------------------------------
* DISPLAY THE CPU USAGE BY TCB IN A CICS REGION ...
*-------------------------------------------------------------
PROGRAM-ID. IPPCDTCB.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
*-------------------------------------------------------------
* IMPORTANT : PICTURES SHOULD NOT BE MODIFIED.
*-------------------------------------------------------------
Ø1 IPPCGTCB PIC X(8) VALUE 'IPPCDTCB'.
Ø1 STORTIM PIC S9(15).
Ø1 STORDAT PIC S9(9).
Ø1 CPU-WORK PIC ZZZZZ9.999.
Ø1 NUM-WORK PIC 9999.
Ø1 ABSTIME PIC 9(15) COMP-3.
Ø1 CNT PIC S9(4) COMP VALUE Ø.
Ø1 CNTT PIC S9(4) COMP VALUE Ø.
Ø1 TCBROWS PIC X(79) VALUE SPACES.
Ø1 TCBCOLR PIC X VALUE SPACES.
Ø1 RECRESP PIC S9(8) COMP VALUE Ø.
Ø1 TCB-CPUTIME PIC S9(12)V999 COMP VALUE Ø.
Ø1 TCB-PCTU PIC S9(8) COMP VALUE Ø.
Ø1 TCBA-ROWS PIC X(26) VALUE SPACES.
Ø1 EXT-CNT PIC S9(8) COMP VALUE Ø.
Ø1 INT-CNT PIC S9(8) COMP VALUE Ø.
Ø1 SWSORT PIC X VALUE SPACES.
*-------------------------------------------------------------
* THE COBOL COPYBOOK FOR THE MAP
*-------------------------------------------------------------
COPY IPPDTCB.
COPY DFHAID.
COPY DFHBMSCA.
*-------------------------------------------------------------
* THE PSEUDO-CONVERSE COMMUNICATIONS AREA
*-------------------------------------------------------------
Ø1 COMMAREA.
Ø3 FILLER PIC X(8) VALUE 'IPPCDTCB'.
Ø3 FILLER PIC X(8) VALUE 'COMMAREA'.
Ø3 CNTS PIC S9(4) COMP VALUE Ø.
Ø3 CNTR PIC S9(4) COMP VALUE Ø.
Ø3 SW-PF7     PIC X     VALUE 'N'.
Ø3 SW-PF8     PIC X     VALUE 'N'.

* SW-MODE : T=TOTAL, D=DELTA
Ø3 SW-MODE    PIC X     VALUE 'T'.

* SW-DISP : C=TCBCPU/TOTCPU, E=TCBcpu/ELAPSED
Ø3 SW-DISP    PIC X     VALUE 'C'.

* SORT-FLD : PROGRAM, ADDRESS OR CPU TIME
Ø3 SORT-FLD   PIC X(8)  VALUE SPACES.

Ø3 ASCB-REC.
Ø5 ASCB-TCB   PIC S99999V999 COMP.
Ø5 ASCB-SRB   PIC S99999V999 COMP.
Ø5 ASCB-NUM   PIC S9(8) COMP.
Ø5 ASCB-SUM   PIC S99999V999 COMP.
Ø5 ASCB-MAX   PIC S9(8) COMP.
Ø5 ASCB-TIM   PIC S99999V999 COMP.

Ø3 TCBA-REC.
Ø5 TCBA-ROW OCCURS 256.
    Ø7 TCB-ADDR   PIC X(8).
    Ø7 TCB-PROG   PIC X(8).

*    Ø7 TCB-CPUT   PIC S9(8) COMP.
    Ø7 TCB-CPUT   PIC S99999V999 COMP.
    Ø7 TCB-FLG1   PIC X.
    Ø7 TCB-FLG2   PIC X.
    Ø7 TCB-ATCB   PIC S9(8) COMP.

Ø3 ASCB-REC-NEW.
Ø5 ASCB-TCB-NEW PIC S99999V999 COMP.
Ø5 ASCB-SRB-NEW PIC S99999V999 COMP.
Ø5 ASCB-NUM-NEW PIC S9(8) COMP.
Ø5 ASCB-SUM-NEW PIC S99999V999 COMP.
Ø5 ASCB-MAX-NEW PIC S9(8) COMP.
Ø5 ASCB-TIM-NEW PIC S99999V999 COMP.

Ø3 TCBA-REC-NEW.
Ø5 TCBA-ROW-NEW OCCURS 256.
    Ø7 TCB-ADDR-NEW PIC X(8).
    Ø7 TCB-PROG-NEW PIC X(8).

*    Ø7 TCB-CPUT-NEW PIC S9(8) COMP.
    Ø7 TCB-CPUT-NEW PIC S99999V999 COMP.
    Ø7 TCB-FLG1-NEW PIC X.
    Ø7 TCB-FLG2-NEW PIC X.
    Ø7 TCB-ATCB-NEW PIC S9(8) COMP.

Ø3 ASCB-REC-OLD.
Ø5 ASCB-TCB-OLD PIC S99999V999 COMP.
Ø5 ASCB-SRB-OLD PIC S99999V999 COMP.
Ø5 ASCB-NUM-OLD PIC S9(8) COMP.
Ø5 ASCB-SUM-OLD PIC S99999V999 COMP.
Ø5 ASCB-MAX-OLD PIC S9(8) COMP.
*Ø5 ASCB-TIM-OLD PIC S99999V99 COMP.

*Ø3 TCBA-REC-OLD.
*Ø5 TCBA-ROW-OLD OCCURS 256.
   Ø7 TCB-ADDR-OLD PIC X(8).
   Ø7 TCB-PROG-OLD PIC X(8).
*Ø7 TCB-CPUT-OLD PIC S9(8) COMP.
*Ø7 TCB-CPUT-OLD PIC S99999V999 COMP.
Ø7 TCB-FLG1-OLD PIC X.
Ø7 TCB-FLG2-OLD PIC X.
Ø7 TCB-ATCB-OLD PIC S9(8) COMP.

*Ø1 DTCB-WORKAREA.
Ø3 TX-QUIT-TO-CICS.
Ø5 FILLER PIC X(158) VALUE SPACES.
Ø5 TEXT-WORK PIC X(32) VALUE 'DTCB PROGRAM ENDED.'.

*LINKAGE SECTION.
*Ø1 DFHCOMMAREA PIC X(3276Ø).

*PROCEDURE DIVISION.
*
   MOVE SPACES TO TCBMESSO
*   MOVE SPACES TO TCBACTIO
*   EIBCALEN = Ø IS AT FIRST INVOCATION
*   IF EIBCALEN = Ø
      PERFORM STARTIT
   ELSE
      MOVE DFHCOMMAREA TO COMMAREA
      PERFORM REC-MAP
      END-IF
      PERFORM RET-TO-CICS .

*____________________________________________________________________
* WE PROCESS THE AID RECEIVED FROM THE MAP
*____________________________________________________________________
*
*REC-MAP.
*
   EXEC CICS RECEIVE MAP('IPPDTCB') MAPSET('IPPDTCB')
       RESP(RECRESP)
   END-EXEC
*
   IF RECRESP = DFHRESP(NORMAL) OR
      RECRESP = DFHRESP(MAPFAIL)
   EVALUATE EIBAID
      WHEN DFHENTER
      PERFORM PROC-ENTER
WHEN DFHPF8
   IF SW-PF8 = 'Y'
      PERFORM PROC-PF8
   ELSE
      PERFORM PROC-OTHER
   END-IF
WHEN DFHPF7
   IF SW-PF7 = 'Y'
      PERFORM PROC-PF7
   ELSE
      PERFORM PROC-OTHER
   END-IF
WHEN DFHCLEAR
   PERFORM ENDIT
WHEN DFHPF3
   PERFORM ENDIT
WHEN DFHPF5
   PERFORM PROC-PF5
WHEN DFHPF6
   PERFORM PROC-PF6
WHEN OTHER
   PERFORM PROC-OTHER
END-EVALUATE
ELSE
   PERFORM ENDIT
END-IF
.
*
PROC-OTHER.
   MOVE 'YOU HIT A BAD KEY DUMMY' TO TCBMESSO
.
*
PROC-PF5.
* SWAP BETWEEN DISPLAY MODES
   IF SW-DISP = 'C'
      MOVE 'F5-DISP-C' TO TCBPF050
      MOVE 'DISPLAY MODE : TCBCPU/ELAPSED' TO TCBMESSO
      MOVE 'E' TO SW-DISP
   ELSE
      MOVE 'F5-DISP-E' TO TCBPF050
      MOVE 'DISPLAY MODE : TCBCPU/TOT.CPU' TO TCBMESSO
      MOVE 'C' TO SW-DISP
   END-IF
*
*
Editor’s note: this article will be concluded next month.

Stan Adriaensen
Systems Engineer
Groupe Royale Belge/IPP A (Belgium)

© Xephon 1999
CICS/MVS 2.1.2 to CICS/ESA 4.1 migration – part 2

This month we conclude the article giving hints and tips to use during an upgrade from CICS/MVS 2.1.2 or lower to CICS/ESA 4.1.

MESSAGES AND CODES

The messages and codes have changed in CICS/ESA 4.1 to enhance the information given. The messages have gone from the format ‘DFHnnnn’ to ‘DFHxxnnnn’ (‘xx’ representing the active component or domain that caused the message to be issued).

If applications or automation software depend on the outcome of certain messages and the use of abend codes, they may require extra time to get ready for this release of CICS. The CICS message domain is responsible for message conversion to the domain-id format which started in CICS/ESA 3.1. Figure 5 shows a few examples of the converted messages.

<table>
<thead>
<tr>
<th>Old message</th>
<th>New message</th>
<th>Code</th>
<th>Component name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH0302</td>
<td>DFHKC0302</td>
<td>KC</td>
<td>Task control</td>
</tr>
<tr>
<td>DFH0401</td>
<td>DFHPC0401</td>
<td>PC</td>
<td>Program control</td>
</tr>
<tr>
<td>DFH1001</td>
<td>DFHTC1001</td>
<td>TC</td>
<td>Terminal control interface</td>
</tr>
<tr>
<td>DFH1516</td>
<td>DFHSI1516</td>
<td>SI</td>
<td>System initialization</td>
</tr>
<tr>
<td>DFH0310</td>
<td>DFHIC0310</td>
<td>IC</td>
<td>Interval control</td>
</tr>
<tr>
<td>DFH1601</td>
<td>DFHDU1601</td>
<td>DU</td>
<td>Dump domain</td>
</tr>
<tr>
<td>DFH2900</td>
<td>DFHJC2900</td>
<td>JC</td>
<td>Journal control</td>
</tr>
</tbody>
</table>

Figure 5: Examples of converted messages

CMAC – DISPLAY MESSAGES AND CODES

When using CMAC you can specify the abend code, message number, or the component identification plus the message number. You will
notice, after applying 75% of future APARs, you will have to ACTION updating CMAC. I gave up on this and always bypass this action. I have a current CD and use READIBM for messages and codes and all other manuals.

LPA
Use ‘LPA=YES’ because most of the CICS management modules are now in ELPA. Change the programs loaded from LPA to USELPACOPY(YES) in the RDO entry and be sure to remove all the LPA loaded modules from the CICS STEPLIB, otherwise they will not be taken from LPA.

The SIT PRVMOD parameter can still be used to name CICS management modules and user modules that should not use LPA.

You should review members IEAICSxx, IEAIPSxx (CICS performance group), CSVLLAxx (library lookaside), COFVLFxx (virtual lookaside) as possible alternatives.

You might want to change MVS’s IEFUSI to increase the storage available above the 16 MB line which affects the EDSASZE parameter in CICS/ESA 4.1.

APF-AUTHORIZED LIBRARIES
The following are APF-authorized libraries:

- SYS1.CICS410.SDFHLINK (also in LNKLSTxx).
- SYS1.CICS410.RCT.LOADLIB (my RCT LOADLIB that is link-listed above SDFHLINK).
- SYS1.CICS410.SDFHAUTH.
- SYS1.CICS410.SDFHLPA.

Note: if you are using the RCT LOADLIB, make sure someone else in your group does not assemble the RCT into a STEPLIB library. As mentioned above: STEPLIB first, then LNKLST.
SOME CONSIDERATIONS WITH TABLES

With tables, you should consider the following:

- **ALT** – this is obsolete.
- **DCT** – REUSE and RSL are obsolete (anything to do with internal security is obsolete), non-resident TD destinations are not supported.
- **FCT** – remove the CSD from FCT (it is now in the SIT override). LSRPOOLS is defined in the CSD. REUSE is obsolete. READ and ADD have replaced GET and NEWREC. RSL is obsolete.
- **JCT** – INPUT, JOUROPT, and RSL are obsolete. BUFSUV has been replaced.
- **MCT** – ACCOUNT and RECORD are obsolete.
- **NLT** – this is obsolete.
- **PCT** – this is obsolete (CSD RDO).
- **PLT** – this has multiple phases. Programs must be defined with EXECKEY(CICS), or risk potential AEZD abend.
- **PPT** – this is obsolete (CSD RDO).
- **SNT** – this is obsolete.
- **SRT** – ROUTINE and PROGRAM are obsolete.
- **TCT** – VTAM definitions are no longer allowed in the TCT.

THE SIT

Obsolete parameters (some introduced in CICS/ESA Version 3) are:

- **AMXT**
- **CDSASZE/CSCS**
- **CMXT/CMXTLIM**
- **ECDSASZE/ECSCS**
- **ERDSASZE/ERSCS**
• EUDSASZE/EUSCS
• SRDELAY
• MAXSMIR
• UDSASZE/USCS.

The MXT value must be set to a reasonable figure because the storage acquired for kernel stacks is based on the MXT value. The region can become constrained. If it’s set too low, performance can suffer. The ICVR default is zero and the RAPOOL default is two (this should be at least 30).

TRACE has changed, so check out the CETR transaction and the new trace format utility.

TRACE is replaced by the SIT parameters INTTR, GTFTR, SYSTR, STNTR, SPCTR, and USERTR.

AEYD is a new abend code. It abends a task that is requesting access to storage for which it is not authorized. I had this problem when a user link-edited his program as re-entrant when the program was in fact modifying storage areas within itself and we coded ‘RENTPGM=PROTECT’ for the program.

APPLICATION PROGRAMMING CONSIDERATIONS

COMMAREA

You should read Chapter 4 of the CICS/ESA Migration Guide. The address of the COMMAREA can be above or below the 16MB line.

You will get unpredictable results if the received COMMAREA does not match the expected length. Whether you LINK or XCTL, you should always specify the length of the COMMAREA.

The COMMAREA can be in CICS-key storage or USER-key storage if CICS is running with storage protection (READ-ONLY storage if obtained by an MVS GETMAIN).

There is a new facility to check for a positive, zero, or negative value (with an AEIV abend, a zero value is not assumed).
Year 2000 support in CICS/ESA V4.1

With FORMATTIME, date formats are YYYYMMDD, YYYYDDMM, YYYYDDD, DDMMYYYY, and MMDDYYYY. CICS/MVS 2.1.2 supports two-digit dates.

The EIBDATE function has not changed. It provides CICS Command Level access in the packed decimal form of 0CYYDDD+ where C is a century indicator. (20th century=0, 21st century=1, and 22nd century=2). For example, 25 December 1999 is ‘0099359+’ while 1 January 2000 is ‘0100001+’.

There are other considerations that I have not specified within this report. The on-line environment of each company is made up of many different unique components. These components or OEM software will have to be investigated by the migration team. My team was made up of four CICS systems programmers, one DBA, two application programmers, and one MVS part-time systems programmer.

Joe DiFranco
Senior Systems Programmer
Workplace Safety and Insurance Board of Ontario (Canada) © Xephon 1999

CICS message suppression and re-routing

INTRODUCTION

CICS generates a multitude of messages, some of which are useful and a lot which are not. The not-so-useful ones can be suppressed using the message domain global exit XMEOUT.

XMEOUT is called every time CICS is about to issue a message from the message domain. It receives information about where the message has come from (which domain) and where it is going (TD queue or console). It can re-route the message by changing the routing information or can suppress it by exiting with the appropriate return code.
IMPLEMENTATION
The program SPGXMEO has been written to be executed as global exit XMEOUT. The processing it carries out is driven by two tables that are loaded during PLTPI processing. They are:

- SPGXMEOD – this tells SPGXMEO which TD queues it is to be interested in.
- SPGXMEOM – this tells SPGXMEO which messages to look for and what to do with them.

The tables are loaded above the 16MB line by the program SPGXMEOP, which is run by CICS during PLTPI processing. It enables SPGXMEO as XMEOUT and places the addresses of SPGXMEOD and SPGXMEOM in XMEOUT’s global work area so that they can be accessed by SPGXMEO.

SPGXMEOP is also run during PLTSD processing to turn off message suppression during CICS closedown.

A program SPGXMEOR is provided that can be used to refresh the message and routing tables dynamically.

XMEODEST MACRO
This is used to generate the table of transient data queues for which messages are to be processed:

```
XMEODEST TYPE=<INITIAL|ENTRY|FINAL>,
    DEST=<transient data queue name>
```

There must be only one ‘TYPE=INITIAL’ and it must be the first entry. Similarly, there must be only one ‘TYPE=FINAL’ and it must be the last entry. There can be any number of ‘TYPE=ENTRY’ entries.

XMEOMESS MACRO
This is used to generate the table of messages to be processed and the action to be carried out:

```
XMEOMESS TYPE=<INITIAL|ENTRY|FINAL>,
    DOMAIN=<originating domain>,
    MESSAGE=<message number>.
```
DEST=<TD queue where message is to be sent>,
ROUTE=<Console where message is to be sent>

There must be only one ‘TYPE=INITIAL’ and it must be the first entry. Similarly, there must be only one ‘TYPE=FINAL’ and it must be the last entry. There can be any number of ‘TYPE=ENTRY’ entries.

The ‘DEST’ and ‘ROUTE’ parameters are mutually exclusive. If neither parameter is specified then the message is suppressed.

RDO DEFINITIONS

DEFINE PROGRAM(SPGXMEO) GROUP(MESSSUPP)
DESCRIPTION(MESSAGE SUPPRESSION EXIT)
   LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
   USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALLOCATION(ANY)
   EXECKEY(CICS) EXECUTIONSET(FULLAPI)

DEFINE PROGRAM(SPGXMEOD) GROUP(MESSSUPP)
DESCRIPTION(MESSAGE SUPPRESSION DESTINATION TABLE)
   LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
   USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALLOCATION(ANY)
   EXECKEY(CICS) EXECUTIONSET(FULLAPI)

DEFINE PROGRAM(SPGXMEOM) GROUP(MESSSUPP)
DESCRIPTION(MESSAGE SUPPRESSION MESSAGE TABLE)
   LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
   USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALLOCATION(ANY)
   EXECKEY(CICS) EXECUTIONSET(FULLAPI)

DEFINE PROGRAM(SPGXMEOP) GROUP(MESSSUPP)
DESCRIPTION(MESSAGE SUPPRESSION PLT PROGRAM)
   LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
   USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALLOCATION(ANY)
   EXECKEY(CICS) EXECUTIONSET(FULLAPI)

DEFINE PROGRAM(SPGXMEOR) GROUP(MESSSUPP)
DESCRIPTION(MESSAGE SUPPRESSION REFRESH PROGRAM)
   LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
   USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALLOCATION(ANY)
   EXECKEY(CICS) EXECUTIONSET(FULLAPI)

DEFINE TRANSACTION(XMER) GROUP(MESSSUPP)
   PROGRAM(SPGXMEOR) TWASIZE(Ø) PROFILE(DFHCICST)
   STATUS(ENABLED)
   TASKDATALOC(ANY) TASKDATAKEY(CICS) STORAGECLEAR(NO)
   RUNAWAY(SYSTEM) SHUTDOWN(DISABLED) ISOLATE(YES) DYNAMIC(NO)
   PRIORITY(1) TRANCLASS(DFHTCLØØ) DTIMOUT(NO) INDOUBT(BACKOUT)
   RESTART(NO) SPURGE(NO) TPURGE(NO) DUMP(YES) TRACE(YES)
   CONFDATA(NO) RESSEC(NO) CMDSEC(NO)
PLT ENTRIES

SPGXMEOP should be included in both PLT tables. It should be after DFHDELIM in the start-up PLT and before it in the shutdown table.

SPGXMEO

*---------------------------------------------------------------*
* *---------------------------------------------------------------*
* S P G X M E O * *---------------------------------------------------------------*
* = = = = = = = * *---------------------------------------------------------------*
* * MESSAGE SUPPRESSION AND RE-ROUTING EXIT * *---------------------------------------------------------------*
* *---------------------------------------------------------------*
* ---------------------------------------------------------------*
* THIS INSTRUCTION SETS UP THE DSECT FOR XMEOUT * *---------------------------------------------------------------*
* D F H U X I T T Y P E = E P , I D = X M E O U T * *---------------------------------------------------------------*
* ---------------------------------------------------------------*
* REGISTER EQUATES * *---------------------------------------------------------------*
* *---------------------------------------------------------------*
* RØ EQU Ø
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R1Ø EQU 1Ø
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
* * GLOBAL WORK AREA LAYOUT * *---------------------------------------------------------------*
* *---------------------------------------------------------------*
SPGXMEO_GWA DSECT
DEST_TABLE_ADDRESS DS F DESTINATION TABLE ADDRESS
MESSAGE_TABLE_ADDRESS DS F MESSAGE TABLE ADDRESS

* MESSAGE TABLE LAYOUT
*
MESSAGE_TABLE DSECT
MESSAGE_NO DS AL4 MESSAGE NUMBER
MESSAGE_DOM DS CL2 MESSAGE DOMAIN
       DS AL1 RESERVED
MESSAGE_ROUTE DS AL1 MESSAGE ROUTE
MESSAGE_DEST DS CL4 MESSAGE DESTINATION
MESSAGE_TABLE_LENGTH EQU *-MESSAGE_NO
*
SPGXMEO CSECT
SPGXMEO AMODE 31
SPGXMEO RMODE ANY
SAVE (14,12) SAVE REGISTERS
       LR R11,R15
       USING SPGXMEO,R11 SET UP PROGRAM BASE REGISTER
       LR R2,R1
       USING DFHUEPAR,R2 ADDRESS USER EXIT PARAMETER LIST
**********************************************************************
                          START.                                        *
**********************************************************************
*
       L     R3,UEPGAA GET GWA ADDRESS
       LTR   R3,R3 IF IT'S ZERO
       BZ    ALLOW_THIS_ONE THEN NO GWA SO GET OUT
       USING SPGXMEO_GWA,R3
*
       L     R4,UEPMTDQ GET TDQ ARRAY ADDRESS
       L     R5,UEPMNTD GET NUMBER OF TDQS ADDRESS
       LH    R5,Ø(R5) GET NUMBER OF TDQS
       LTR   R5,R5 DO WE HAVE ANY?
       BZ    ALLOW_THIS_ONE NO - NOTHING TO DO
CHECK_NEXT_TDQ DS ØH
       L     R6,DEST_TABLE_ADDRESS GET DEST TABLE ADDRESS
CHECK_NEXT_DEST DS ØH
       CLC  Ø(4,R6),HIGH_VALUES END OF TABLE?
       BE    POINT_TO_NEXT_TDQ YES - GO TO NEXT TDQ
       CLC  Ø(4,R6),Ø(R4) ARE WE INTERESTED?
       BE    DEST_TDQ_MATCH YES - GO AND CHECK THE MESSAGE
       LA    R6,4(R6) POINT TO NEXT DEST
       B     CHECK_NEXT_DEST GO AND CHECK IT
POINT_TO_NEXT_TDQ DS ØH
       LA    R4,4(R4) POINT TO NEXT TDQ
       BCT   R5,CHECK_NEXT_TDQ IF ANY LEFT THEN GO AND CHECK
       B     ALLOW_THIS_ONE
DEST_TDQ_MATCH DS ØH
       L     R4,UEPMDOM GET ADDRESS OF DOMAIN
       L     R5,UEPMNUM GET ADDRESS OF MESSAGE
L R6,MESSAGE_TABLE_ADDRESS GET MESSAGE TABLE ADDRESS USING MESSAGE_TABLE,R6
*
MESSAGE_LOOP DS ØH
*
CLC MESSAGE_NO,HIGH_VALUES END OF TABLE?
BE ALLOW_THIS_ONE YES - NOTHING TO DO
CLC MESSAGE_DOM,Ø(R4) IS IT FOR THE CURRENT DOMAIN
BE POSSIBLE_ONE YES - POSSIBLY INTERESTED
*
GET_NEXT_ENTRY DS ØH
*
LA R6,MESSAGE_TABLE_LENGTH(R6) POINT TO NEXT ENTRY
B MESSAGE_LOOP AND GO ROUND AGAIN
*
POSSIBLE_ONE DS ØH
CLC MESSAGE_NO,Ø(R5) IS THIS OUR MESSAGE
BNE GET_NEXT_ENTRY NO - GO TO CHECK NEXT ENTRY
*
CLI MESSAGE_ROUTE,X'ØØ' RE-ROUTE TO CONSOLE?
BNE ROUTE_TO_CONSOLE YES - GO TO IT
CLC MESSAGE_DEST,SPACES RE-ROUTE TO TDQ
BNE ROUTE_TO_TDQ YES - GO TO IT
LA R15,UERCBYP OTHERWISE JUST SUPPRESS IT
B THATS_ALL_FOLKS
*
ROUTE_TO_CONSOLE DS ØH
L R4,UEPMROU GET ADDRESS OF ROUTING CODES
L R5,UEPMNRC GET ADDRESS NUMBER OF ROUTING CODES
L R7,UEPMNTD GET ADDRESS NUMBER OF TDQS
IC R1,MESSAGE_ROUTE GET NEW ROUTING CODE
STC R1,Ø(R4) AND SET IT
LA R1,1 SET NUMBER OF ROUTING CODES...
STH R1,Ø(R5) ....TO 1
XR R1,R1 SET NUMBER OF DESTINATIONS....
STH R1,Ø(R7) ....TO ZERO
B ALLOW_THIS_ONE
ROUTE_TO_TDQ DS ØH
L R4,UEPMTDQ GET ADDRESS OF DEST TABLE
L R5,UEPMNTD GET ADDRESS NUMBER OF TDQS
L R7,UEPMNRC GET ADDRESS NUMBER OF ROUTING CODES
MVC Ø(4,R4),MESSAGE_DEST SET NEW DESTINATION
LA R1,1 SET NUMBER OF DESTS....
STH R1,Ø(R5) ....TO 1
XR R1,R1 SET NUMBER OF ROUTING CODES....
STH R1,Ø(R7) ....TO ZERO
ALLOW_THIS_ONE DS ØH
*
LA R15,UERCNORM SET THE RETURN CODE TO NORMAL @P2C
B THATS_ALL_FOLKS

*  ************************************************************
*  RESTORE REGISTERS, SET RETURN CODE, AND RETURN TO USER  *
*  EXIT HANDLER.                                          *
*  **********************************************************
*  THATS_ALL_FOLKS DS ØH
L       R13,UEPEPSA
RETURN (14,12),RC=(15)
LTORG
*
HIGH_VALUES DC AL4(-1)
SPACES      DC C'    '
*  
END

SPGXMEOD

XMEODEST TYPE=INITIAL
XMEODEST TYPE=ENTRY,DEST=CADL
XMEODEST TYPE=ENTRY,DEST=CDBC
XMEODEST TYPE=ENTRY,DEST=COUL
XMEODEST TYPE=ENTRY,DEST=CRDI
XMEODEST TYPE=ENTRY,DEST=CSCS
XMEODEST TYPE=ENTRY,DEST=CSDL
XMEODEST TYPE=ENTRY,DEST=CSFL
XMEODEST TYPE=ENTRY,DEST=CSKL
XMEODEST TYPE=ENTRY,DEST=CSML
XMEODEST TYPE=ENTRY,DEST=CSPL
XMEODEST TYPE=ENTRY,DEST=CSMT
XMEODEST TYPE=ENTRY,DEST=CSMT
XMEODEST TYPE=ENTRY,DEST=CSTL
XMEODEST TYPE=FINAL
END

SPGXMEOM

XMEOMESS TYPE=INITIAL
XMEOMESS TYPE=ENTRY,DOMAIN=FC,MESSAGE=Ø2ØØ
XMEOMESS TYPE=ENTRY,DOMAIN=FC,MESSAGE=Ø2Ø1
XMEOMESS TYPE=ENTRY,DOMAIN=KC,MESSAGE=Ø1Ø1
XMEOMESS TYPE=ENTRY,DOMAIN=ZC,MESSAGE=6966
XMEOMESS TYPE=ENTRY,DOMAIN=ZC,MESSAGE=6935
XMEOMESS TYPE=ENTRY,DOMAIN=ZC,MESSAGE=5966
XMEOMESS TYPE=ENTRY,DOMAIN=ZC,MESSAGE=3464
XMEOMESS TYPE=ENTRY,DOMAIN=ZC,MESSAGE=3461
XMEOMESS TYPE=ENTRY,DOMAIN=ZC,MESSAGE=3462
XMEOMESS TYPE=ENTRY,DOMAIN=ZN,MESSAGE=211Ø,DEST=LNK2
XMEOMESS TYPE=ENTRY,DOMAIN=AC,MESSAGE=2236,ROUTE=1
SPGXMEOP

* * * * * * * * * *
* SPGXMEOP * *
* * * * * * * * * *
* PLT PROGRAM TO START MESSAGE SUPPRESSION ON CICS STARTUP AND STOP * *
* IT BEFORE SHUTDOWN * *
* * * * * * * * * *
* * * * * * * * * *

DFHREGS

DFHEISTG

RESPONSE DS F CICS RESP
CICS_STATUS DS F CICS STATUS
GWA_LEN DS H GWA LENGTH

MESSAGE_AREA DS CL8Ø

MESSAGE_ID DS CL1Ø

MESSAGE_CICS DS CL8

MESSAGE_TEXT DS CL6Ø

SPGXMEOP DFHEIENT EIBREG=11,CODEREG=12,DATAREG=13

EXEC CICS HANDLE ABEND LABEL(ABENDED)

XR R2,R2
XR R3,R3

LA R9,Ø
BAL R10,ISSUE_MESSAGE

EXEC CICS INQUIRE SYSTEM CICSSTATUS(CICS_STATUS) X

RESPONSE,DFHRESP(NORMAL)

UNKNOWN_CICS_STATUS

CICS_STATUS,DFHVALUE(STARTUP)

CICS_IS_CLOSING

INVALID_CICS_STATUS
EXEC CICS LOAD PROGRAM('SPGXMEOD') SET(R2) HOLD RESP(RESPONSE)
CLC RESPONSE,DFHRESP(NORMAL)
BNE CANT_LOAD_SPGXMEOD

EXEC CICS LOAD PROGRAM('SPGXMEOM') SET(R3) HOLD RESP(RESPONSE)
CLC RESPONSE,DFHRESP(NORMAL)
BNE CANT_LOAD_SPGXMEOM

* ENABLE THE EXIT TO GET THE GWA
*
MVC GWA_LEN,GWALEN
EXEC CICS ENABLE
   PROGRAM('SPGXMEO')
   EXIT('XMEOUT')
   GALENGTH(GWA_LEN)
   RESP(RESPONSE)
CLC RESPONSE,DFHRESP(NORMAL)
BNE CANT_ENABLE_EXIT

* GET THE GWA ADDRESS
*
EXEC CICS EXTRACT EXIT
   PROGRAM('SPGXMEO')
   GALENGTH(GWA_LEN)
   GASET(4)
   RESP(RESPONSE)
CLC RESPONSE,DFHRESP(NORMAL)
BNE CANT_EXTRACT_EXIT
ST R2,Ø(R4)
ST R3,4(R4)

* START THE EXIT
*
EXEC CICS ENABLE
   PROGRAM('SPGXMEO')
   START
   RESP(RESPONSE)
CLC RESPONSE,DFHRESP(NORMAL)
BNE CANT_START_EXIT

* WE'RE DONE
*
THATS_ALL_FOLKS DS ØH
   LA R9,32
   BAL R10,ISSUE_MESSAGE
EXEC CICS RETURN

UNKNOWN_CICS_STATUS DS ØH
LA R9,4
BAL R10,ISSUE_MESSAGE
B THATS_ALL_FOLKS

* INVALID_CICS_STATUS DS ØH
LA R9,8
BAL R10,ISSUE_MESSAGE
B THATS_ALL_FOLKS

* CANT_LOAD_SPGXMEOD DS ØH
LA R9,12
BAL R10,ISSUE_MESSAGE
B THATS_ALL_FOLKS

* CANT_LOAD_SPGXMEOM DS ØH
EXEC CICS RELEASE PROGRAM('SPGXMEOD')
LA R9,16
BAL R10,ISSUE_MESSAGE
B THATS_ALL_FOLKS

* CANT_ENABLE_EXIT DS ØH
EXEC CICS RELEASE PROGRAM('SPGXMEOD')
EXEC CICS RELEASE PROGRAM('SPGXMEOM')
LA R9,20
BAL R10,ISSUE_MESSAGE
B THATS_ALL_FOLKS

* CANT_EXTRACT_EXIT DS ØH
EXEC CICS RELEASE PROGRAM('SPGXMEOD')
EXEC CICS RELEASE PROGRAM('SPGXMEOM')
EXEC CICS DISABLE PROGRAM('SPGXMEO') EXIT('XMEOUT')
LA R9,24
BAL R10,ISSUE_MESSAGE
B THATS_ALL_FOLKS

* CANT_START_EXIT DS ØH
EXEC CICS RELEASE PROGRAM('SPGXMEOD')
EXEC CICS RELEASE PROGRAM('SPGXMEOM')
EXEC CICS DISABLE PROGRAM('SPGXMEO') EXIT('XMEOUT')
LA R9,28
BAL R10,ISSUE_MESSAGE
B THATS_ALL_FOLKS

* ABENDED DS ØH
LTR R2,R2
BZ NO_SPGXMEOD
EXEC CICS RELEASE PROGRAM('SPGXMEOD')
NO_SPGXMEOD DS ØH
LTR R2,R2
BZ NO_SPGXMEOM
EXEC CICS RELEASE PROGRAM('SPGXMEOM')
NO_SPGXMEOM DS ØH
    EXEC CICS DISABLE PROGRAM('SPGXMEOM') EXIT('XMEOUT')
    LA R9,36
    BAL R10,ISSUE_MESSAGE
B THATS_ALL_FOLKS
*
* DISABLE THE EXIT DURING CICS CLOSEDOWN
*   CICS_IS_CLOSING DS ØH
*
    EXEC CICS DISABLE
    PROGRAM('SPGXMEOM')
    EXITALL
*
    EXEC CICS RELEASE PROGRAM('SPGXMEOD') RESP(RESPONSE)
*
    EXEC CICS RELEASE PROGRAM('SPGXMEOM') RESP(RESPONSE)
*
B THATS_ALL_FOLKS
*
   ISSUE_MESSAGE DS ØH
       MVI MESSAGE_AREA,C' '
       MVC MESSAGE_AREA+1(L'MESSAGE_AREA-1'),MESSAGE_AREA
       EXEC CICS ASSIGN APPLID(MESSAGE_CICS)
       L  R1,MESS_CODE_TABLE(R9)
       MVC MESSAGE_ID,Ø(R1)
       MVC MESSAGE_TEXT,Ø(R1)
       EXEC CICS WRITE OPERATOR TEXT(MESSAGE_AREA) RESP(RESPONSE)
       XR R9,R9
       BR R10
*
GWALEN DC AL2(8)
*
LTORG
*
MESS_CODE_TABLE DS ØF
   DC A(MESSØ)
   DC A(MESS4)
   DC A(MESS8)
   DC A(MESS12)
   DC A(MESS16)
   DC A(MESS2Ø)
   DC A(MESS24)
   DC A(MESS28)
   DC A(MESS32)
   DC A(MESS36)
*
MESSØ DC CL1Ø‘SPGXMEOPØØ’
TEXTØ DC CL6Ø‘SPGXMEOP PROCESSING STARTED’
* MESS4 DC CL1Ø‘SPGXMEOPØ4’
TEXT4 DC CL6Ø‘UNKNOWN CICS STATUS RECEIVED’
* MESS8 DC CL1Ø‘SPGXMEOPØ8’
TEXT8 DC CL6Ø‘NOT RUN BY PLTP1 OR PLTSD’
* MESS12 DC CL1Ø‘SPGXMEOP12’
TEXT12 DC CL6Ø‘UNABLE TO LOAD TDQ TABLE SPGXMEOD’
* MESS16 DC CL1Ø‘SPGXMEOP16’
TEXT16 DC CL6Ø‘UNABLE TO LOAD MESSAGE TABLE SPGXMEOM’
* MESS2Ø DC CL1Ø‘SPGXMEOP2Ø’
TEXT2Ø DC CL6Ø‘UNABLE TO ENABLE EXIT PROGRAM SPGXMEO’
* MESS24 DC CL1Ø‘SPGXMEOP24’
TEXT24 DC CL6Ø‘UNABLE TO EXTRACT EXIT PROGRAM SPGXMEO’
* MESS28 DC CL1Ø‘SPGXMEOP28’
TEXT28 DC CL6Ø‘UNABLE TO START EXIT PROGRAM SPGXMEO’
* MESS32 DC CL1Ø‘SPGXMEOP32’
TEXT32 DC CL6Ø‘SPGXMEOP PROCESSING ENDED’
* MESS36 DC CL1Ø‘SPGXMEOP36’
TEXT36 DC CL6Ø‘SPGXMEOP ABENDED’
*
END

SPGXMEOR

*---------------------------------------------------------------------*
*
*            SPGXMEOR
* = = = = = = = =
*
*---------------------------------------------------------------------*
*
DFHREGS
*
DFHEISTG
GWA_LEN DS H GWA LENGTH
*
SPGXMEOR DFHEIENT EIBREG=11,CODEREG=12,DATAREG=13
*
XR R2,R2

XR     R3,R3
*               EXEC CICS DISABLE PROGRAM('SPGXMEO') STOP
*               EXEC CICS RELEASE PROGRAM('SPGXMEOD')
*               EXEC CICS RELEASE PROGRAM('SPGXMEOM')
*               EXEC CICS SET PROGRAM('SPGXMEOD') NEWCOPY
*               EXEC CICS SET PROGRAM('SPGXMEOM') NEWCOPY
*               EXEC CICS LOAD PROGRAM('SPGXMEOD') SET(R2) HOLD
*               EXEC CICS LOAD PROGRAM('SPGXMEOM') SET(R3) HOLD
*               EXEC CICS EXTRACT EXIT PROGRAM('SPGXMEO') GALENGTH(GWA_LEN) X GASET(4)
*               ST     R2.Ø(R4)
               ST     R3.4(R4)
*               EXEC CICS ENABLE PROGRAM('SPGXMEO') START
*               EXEC CICS SEND TEXT FROM(MESSAGE) ERASE FREEKB
*               EXEC CICS RETURN
*               LTORG
*               MESSAGE DC    C'MESSAGE SUPPRESSION/RE_ROUTING TABLES REFRESHED'
*               END

XMEODEST
MACRO
XMEODEST &TYPE=,&DEST=
AIF   ('&TYPE' EQ 'INITIAL').INITIAL
AIF   ('&TYPE' EQ 'ENTRY').ENTRY
AIF   ('&TYPE' EQ 'FINAL').FINAL
MNOTE 8,'TYPE PARAMETER MUST BE INITIAL, ENTRY OR FINAL'
MEXIT
.CSECT
SPGXMEOD CSECT
SPGXMEOD AMODE 31
SPGXMEOD RMODE ANY
MEXIT
.CSECT
SPGXMEOD CSECT
SPGXMEOD AMODE 31
SPGXMEOD RMODE ANY
MEXIT
.CSECT
XMEOMESS

MACRO
XMEOMESS &TYPE=,&DOMAIN=,&MESSAGE=,&DEST=,&ROUTE=
AIF ('&TYPE' EQ 'INITIAL').INITIAL
AIF ('&TYPE' EQ 'ENTRY').ENTRY
AIF ('&TYPE' EQ 'FINAL').FINAL
MNOTE 8,'TYPE PARAMETER MUST BE INITIAL, ENTRY OR FINAL'
MEXIT

.INITIAL ANOP
SPGXMEOM CSECT
SPGXMEOM AMODE 31
SPGXMEOM RMODE ANY
MEXIT

.ENTRY   ANOP
AIF ('&DEST' EQ '').NODEST
AIF ('&ROUTE' EQ '').NOROUTE
MNOTE 8,'DEST AND ROUTE ARE MUTUALLY EXCLUSIVE'
MEXIT

.NODEST ANOP
AIF ('&ROUTE' NE '').ROUTE
&R SETC 'Ø'
&D SETC '     '
AGO .TABENT

.ROUTE ANOP
&R SETC '&ROUTE'
&D SETC '     '
AGO .TABENT

.NOROUTE ANOP
&R SETC 'Ø'
&D SETC '&DEST'

.TABENT ANOP
DC AL4(&MESSAGE)          MESSAGE NUMBER
DC CL2 '&DOMAIN'          MESSAGE DOMAIN
DC AL1(Ø)                RESERVED
DC AL1(&R)               MESSAGE ROUTE CODE
DC CL4 '&D'              MESSAGE DESTINATION
MEXIT

.FINAL ANOP
DC AL4(-1)
MEND

---

Kevin Wailes
J Sainsbury (UK)
Information Builders has announced Parlay JavaBean for CICS, enabling transparent communications between applications managed by CICS and Java application server programs. It provides automatic network protocol translation between HTTP and/or IIOP and APPC and/or TCP/IP and SNA when communicating cross-platform.

The CICS bean provides automatic formatting for data output from CICS and provides encrypted communications and either synchronous or asynchronous interaction between platform environments.

For further information contact:
Information Builders, 1250 Broadway, 30th Floor, New York, NY 10001, USA.
Tel: (212) 736 4433.
Information Builders (UK), Wembley Point, Harrow Road, Wembley, Middlesex, HA9 6DE, UK.
Tel: (0181) 982 4700.

***

CICS users can benefit from a joint package from Blue Lobster and SevenMountains Software. The Web-to-legacy package integrates SevenMountains’ TaskForce enterprise desktop and productivity suite with Blue Lobster’s Mako and Stingray Java software for 3270, 5250, and transactional CICS Web connectivity. This provides Java connectivity to mainframe applications and data via the corporate intranet or Internet.

TaskForce provides an alternative to fat client groupware and desktop environments, as an easy-to-administer IP solution that can be accessed by any Java-enabled client.

Mako and Stingray are for Web-to-mainframe connectivity. The latter has a record/code generation model that allows users to build Java-based applets and applications that communicate with any terminal-based 3270/5250 application. Mako’s LBO Builder allows users to create client or server-based Web applications that directly access any CICS application.

For further information contact:
SevenMountains Software, 1450 Fashion Island Boulevard, Suite 680, San Mateo, CA 94404, USA.
Tel: (650) 574 5023.
Blue Lobster, 2005 Hamilton Avenue, Suite 270, San Jose, CA 95125, USA.
Tel: (408) 371 5300.

***

IBM has announced its CICS Web enablement planning service offering on-site planning assistance. An IBM specialist reviews the user’s internet e-business strategy, analyses the existing environment (including the technical infrastructure, CICS configuration, and CICS applications), and evaluates business requirements for Web enablement.

For further information contact your local IBM representative.

***