156

CICS update

November 1998

In this issue

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 £170.00 in the UK; $260.00 in the USA and Canada; £176.00 in Europe; £182.00 in Australasia and Japan; and £180.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1994 issue, are available separately to subscribers for £14.50 ($22.00) each including postage.

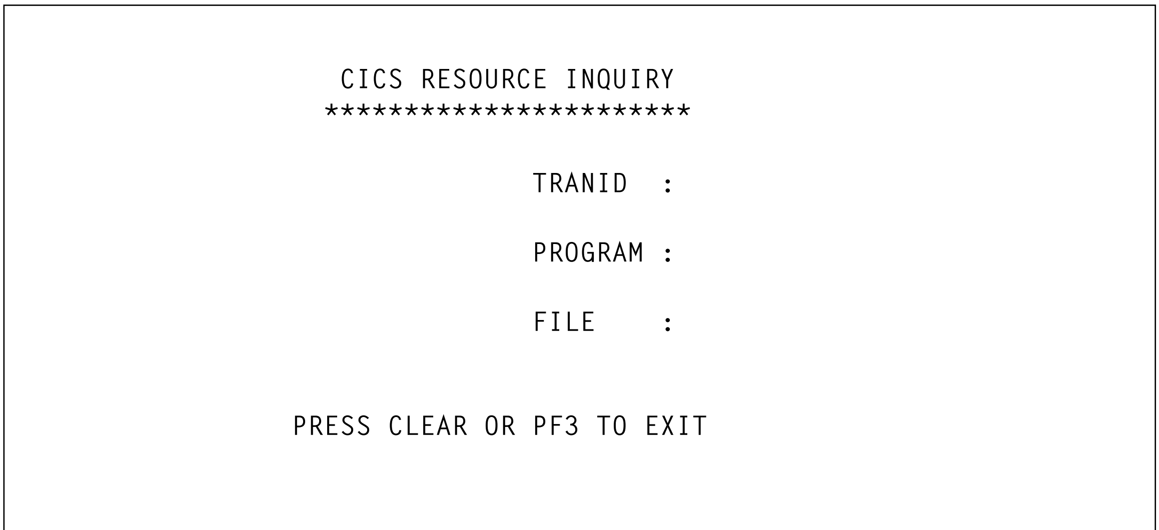
Printed in England.

Cross memory resource inquiry program

The following COBOL II program was developed to help CICS programmers locate CICS system resources within a group of CICS regions that are connected through MVS cross memory services. Since CICS INQUIRY commands are not shippable to other CICS regions, this program was developed to do a remote link from itself into the other connected CICS regions to collect system resource information requested by the user. The program makes use of the SYSID parameter on the LINK command, available in Release 4.1 of CICS, to communicate with the other CICS regions.

The program code determines whether it needs to play the role of a ‘client’ or a ‘server’. The ‘client’ program links to the available cross memory (XM) CICS regions that are acquired by the CICS region that is executing the original transaction. The originating CICS region also handles the terminal interaction between the user and the program. When a ‘server’ program in another CICS is linked to by the ‘client’ program, the ‘server’ program collects information about the requested resource and returns the information to the ‘client’ program. The ‘client’ program sends the collected information back to the terminal.

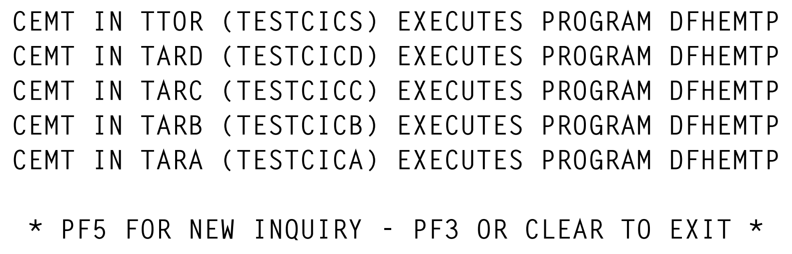
The program makes use of a BMS map that is sent to the invoking terminal to allow the user to specify the resource type (transaction,



```
                     CICS RESOURCE INQUIRY
                    ***********************

                                 TRANID  :

                                 PROGRAM :

                                 FILE    :


                  PRESS CLEAR OR PF3 TO EXIT
```

Figure 1: Inquiry screen

program, or file) and the name to use in the inquiry (see Figure 1). After all the ‘server’ programs have returned their information to the ‘client’ program, another BMS screen is used to present the information to the user (Figure 2).



```
CEMT IN TTOR (TESTCICS) EXECUTES PROGRAM DFHEMTP
CEMT IN TARD (TESTCICD) EXECUTES PROGRAM DFHEMTP
CEMT IN TARC (TESTCICC) EXECUTES PROGRAM DFHEMTP
CEMT IN TARB (TESTCICB) EXECUTES PROGRAM DFHEMTP
CEMT IN TARA (TESTCICA) EXECUTES PROGRAM DFHEMTP

 * PF5 FOR NEW INQUIRY - PF3 OR CLEAR TO EXIT *
```

Figure 2: Information screen

The design of this program limits the configuration of the CICS complex to a simple two-tier design with up to twenty AOR regions attached to a TOR. A more complex configuration of CICS regions presents the interesting challenge of modifying the program code to allow a ‘server’ program to temporarily become a ‘client’ in order to complete the search for the requested information. (Watch out for recursive program links!)

While this program was developed with the idea of exploiting some of the newer INQUIRY functions available in CICS Version 4.1, a ‘server only’ version was also created to execute in some CICS regions that are at an earlier CICS release (2.1.2). (A remote program LINK to earlier releases of CICS is supported when the earlier release is the target of the program LINK.)

The program source code was copied and then modified to remove all INQUIRY functions not supported at the CICS release. (Running the program code through the 2.1.2 translator flagged all the non-supported code.) The modified source was then linked with the original program name into a library that is available only to the earlier release CICS regions.

INQT100

```
CBL    XOPTS(SP)
       IDENTIFICATION DIVISION.
       PROGRAM-ID. INQT1ØØ.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       77  WS-LENGTH           PIC S9(4) COMP.
       77  WS-SUB1             PIC S9(4) COMP.
       Ø1  WS-WORK-VALUES.
           Ø2  WS-WORK-ID      PIC X(Ø6).
           Ø2  WS-WORK-CONN    PIC X(Ø4).
           Ø2  WS-WORK-NET     PIC X(Ø8).
           Ø2  WS-WORK-ACC     PIC S9(Ø8) COMP.
           Ø2  WS-WORK-SRV     PIC S9(Ø8) COMP.
           Ø2  WS-WORK-LANG    PIC S9(Ø8) COMP.
           Ø2  WS-WORK-PTYPE   PIC S9(Ø8) COMP.
           Ø2  WS-WORK-CTYPE   PIC S9(Ø8) COMP.
           Ø2  WS-WORK-OPEN    PIC S9(Ø8) COMP.
           Ø2  WS-WORK-READ    PIC S9(Ø8) COMP.
           Ø2  WS-WORK-BROWSE  PIC S9(Ø8) COMP.
           Ø2  WS-WORK-ADD     PIC S9(Ø8) COMP.
           Ø2  WS-WORK-UPDATE  PIC S9(Ø8) COMP.
           Ø2  WS-WORK-DELETE  PIC S9(Ø8) COMP.
           Ø2  WS-WORK-TRAN    PIC X(Ø4).
           Ø2  WS-WORK-PROG    PIC X(Ø8).
           Ø2  WS-WORK-FILE    PIC X(Ø8).
           Ø2  WS-WORK-RSYS    PIC X(Ø4).
           Ø2  WS-WORK-RTRAN   PIC X(Ø4).
       Ø1  WS-SEARCH-TYPE      PIC X(Ø4).
       Ø1  WS-NATIVE-ID        PIC X(Ø4).
       Ø1  WS-NATIVE-NET       PIC X(Ø8).
       Ø1  WS-CONN-TABLE.
           Ø2  WS-CONN-ENTRY  OCCURS 2Ø TIMES.
             Ø4 WS-CONN-ID     PIC X(Ø4).
             Ø4 WS-CONN-NET    PIC X(Ø8).

       Ø1  WS-DETAIL-TRAN.
           Ø2 WS-DT-TRAN       PIC X(Ø4).
           Ø2 FILLER           PIC X(Ø4)      VALUE ' IN '.
           Ø2 WS-DT-SYSID      PIC X(Ø4).
           Ø2 FILLER           PIC X(Ø2)      VALUE ' ('.
           Ø2 WS-DT-NETNM      PIC X(Ø8).
           Ø2 FILLER           PIC X(19)    VALUE ') EXECUTES PROGRAM '.
           Ø2 WS-DT-PROG       PIC X(Ø8).

       Ø1  WS-RDETAIL-TRAN.
           Ø2 WS-RDT-TRAN      PIC X(Ø4).
           Ø2 FILLER           PIC X(Ø4)      VALUE ' IN '.
           Ø2 WS-RDT-SYSID     PIC X(Ø4).
           Ø2 FILLER           PIC X(Ø2)      VALUE ' ('.
           Ø2 WS-RDT-NETNM     PIC X(Ø8).
           Ø2 FILLER           PIC X(Ø8)      VALUE ') SHIPS '.
           Ø2 FILLER           PIC X(Ø8)      VALUE ' TO ==> '.
           Ø2 WS-RDT-TARG      PIC X(Ø4).
           Ø2 FILLER           PIC X(Ø2)      VALUE SPACE.
           Ø2 WS-RDT-MSG       PIC X(17)      VALUE SPACE.

       Ø1  WS-REMOTETRAN-MSG.
           Ø2 FILLER           PIC X(12)      VALUE '(REMOTENAME:'.
           Ø2 WS-RTRAN-NAME    PIC X(Ø4)      VALUE SPACE.
           Ø2 FILLER           PIC X(Ø1)      VALUE ')'.

       Ø1  WS-DETAIL-PROG.
           Ø2 WS-DP-PROG       PIC X(Ø8).
           Ø2 FILLER           PIC X(15)      VALUE ' IS DEFINED IN '.
           Ø2 WS-DP-SYSID      PIC X(Ø4).
           Ø2 FILLER           PIC X(Ø2)      VALUE ' ('.
           Ø2 WS-DP-NETNM      PIC X(Ø8).
           Ø2 FILLER           PIC X(Ø5)      VALUE ') AS '.
           Ø2 WS-DP-LANG       PIC X(Ø3).
           Ø2 FILLER           PIC X(Ø9)      VALUE ' PROGRAM '.
           Ø2 WS-DP-LTYPE      PIC X(14)      VALUE SPACE.

       Ø1  WS-RDETAIL-PROG.
           Ø2 WS-RDP-PROG      PIC X(Ø8).
           Ø2 FILLER           PIC X(Ø4)      VALUE ' IN '.
           Ø2 WS-RDP-SYSID     PIC X(Ø4).
           Ø2 FILLER           PIC X(Ø2)      VALUE ' ('.
           Ø2 WS-RDP-NETNM     PIC X(Ø8).
           Ø2 FILLER           PIC X(Ø8)      VALUE ') SHIPS '.
           Ø2 FILLER           PIC X(Ø8)      VALUE ' TO ==> '.
           Ø2 WS-RDP-TARG      PIC X(Ø4).

       Ø1  WS-DETAIL-FILE.
           Ø2 WS-DF-FILE       PIC X(Ø8).
           Ø2 FILLER           PIC X(15)      VALUE ' IS DEFINED IN '.
           Ø2 WS-DF-SYSID      PIC X(Ø4).
           Ø2 FILLER           PIC X(Ø2)      VALUE ' ('.
           Ø2 WS-DF-NETNM      PIC X(Ø8).
           Ø2 FILLER           PIC X(Ø5)      VALUE ') AS '.
           Ø2 WS-DF-FUNC       PIC X(13).
           Ø2 FILLER           PIC X(Ø7)      VALUE ' FILE  '.
           Ø2 WS-DF-MSG        PIC X(14)      VALUE SPACE.

       Ø1  WS-RDETAIL-FILE.
           Ø2 WS-RDF-FILE      PIC X(Ø8).
           Ø2 FILLER           PIC X(Ø4)      VALUE ' IN '.
           Ø2 WS-RDF-SYSID     PIC X(Ø4).
           Ø2 FILLER           PIC X(Ø2)      VALUE ' ('.
           Ø2 WS-RDF-NETNM     PIC X(Ø8).
           Ø2 FILLER           PIC X(Ø8)      VALUE ') SHIPS '.
           Ø2 FILLER           PIC X(Ø8)      VALUE ' TO ==> '.
```

```
     Ø2 WS-RDF-TARG      PIC X(Ø4).

 Ø1  WS-COMM.
     Ø2 WS-COMM-ID       PIC X(Ø6).
     Ø2 WS-COMM-TRAN     PIC X(Ø4)      VALUE SPACE.
     Ø2 WS-COMM-PROG     PIC X(Ø8)      VALUE SPACE.
     Ø2 WS-COMM-FILE     PIC X(Ø8)      VALUE SPACE.
     Ø2 WS-COMM-DLINE    PIC X(77)      VALUE SPACE.

 COPY INQTMØ1.
 COPY INQTMØ2.
 COPY DFHAID.
 COPY DFHBMSCA.

 LINKAGE SECTION.
 Ø1  DFHCOMMAREA         PIC X(1Ø3).
 Ø1  COMM-AREA  REDEFINES DFHCOMMAREA.
     Ø2 COMM-ID          PIC X(Ø6).
     Ø2 COMM-TRAN        PIC X(Ø4).
     Ø2 COMM-PROG        PIC X(Ø8).
     Ø2 COMM-FILE        PIC X(Ø8).
     Ø2 COMM-DETAIL-LINE PIC X(77).
 PROCEDURE DIVISION.
 ØØØØ-MAIN.
*****************************************************************
**  DETERMINE IF THIS IS THE FIRST TIME INTO THE PROGRAM BY    **
**  CHECKING THE COMMAREA LENGTH. SEND THE INPUT MAP ON FIRST  **
**  ENTRY.                                                     **
*****************************************************************
     IF EIBCALEN > Ø
       GO TO 1ØØØ-PROCESS-COMMAREA.

 Ø1ØØ-SEND-MAP.
     EXEC CICS SEND MAP('INQTMØ1')
               ERASE
               END-EXEC.

 Ø2ØØ-RETURN.
*****************************************************************
**  SEND A 'CLIENT' ID TO THE NEXT ITERATION OF THIS PROGRAM   **
**  TO HELP IT DETERMINE WHAT TASKS WILL NEED TO BE DONE.      **
*****************************************************************
     MOVE 'CLIENT'     TO WS-COMM-ID.
     MOVE 1Ø3          TO WS-LENGTH.
     EXEC CICS RETURN TRANSID('INQT')
                   COMMAREA(WS-COMM)
                   LENGTH(WS-LENGTH)
                   END-EXEC.

 1ØØØ-PROCESS-COMMAREA.
*****************************************************************
```

```
**  IF THE USER HIT THE CLEAR KEY OR PF3 THEN CLEAR THE SCREEN  *
**  AND END THE TRANSACTION. RE-SEND THE INITAL SCREEN IF THE   *
**  USER HIT PF5.                                               *
*****************************************************************
     IF EIBAID = DFHCLEAR
       GO TO 9999-END.

     IF EIBAID = DFHPF3
       GO TO 9999-END.

     IF EIBAID = DFHPF5
       GO TO Ø1ØØ-SEND-MAP.

*****************************************************************
**  IF THIS PROGRAM IS A 'SERVER' THEN DROP DOWN TO THE SERVER  *
**  CODE.                                                       *
*****************************************************************
     IF COMM-ID = 'SERVER'
       GO TO 5ØØØ-PROCESS-SERVER.

*****************************************************************
**  THE 'CLIENT' EXECUTION OF THIS PROGRAM WILL INTERFACE WITH  *
**  THE ATTACHED TERMINAL BY PULLING IN THE RESOURCE REQUEST    *
**  FROM THE TERMINAL.                                          *
*****************************************************************
     EXEC CICS RECEIVE MAP('INQTMØ1')
         NOHANDLE
         END-EXEC.

*****************************************************************
*   SELECT THE RESOURCE NAME AND TYPE TO BE USED IN THE SEARCH. *
*   RE-SEND THE INPUT MAP IF ALL THE INPUT FIELDS ARE EMPTY.    *
*****************************************************************
     IF TRANAMEL > Ø
       MOVE TRANAMEI TO WS-COMM-TRAN
       MOVE 'TRAN'   TO WS-SEARCH-TYPE
     ELSE
     IF PRGNAMEL > Ø
       MOVE PRGNAMEI TO WS-COMM-PROG
       MOVE 'PROG'   TO WS-SEARCH-TYPE
     ELSE
     IF FILNAMEL > Ø
       MOVE FILNAMEI TO WS-COMM-FILE
       MOVE 'FILE'   TO WS-SEARCH-TYPE
     ELSE
       MOVE '* NO INPUT DETECTED - PLEASE RE-ENTER *' TO MSGO
       MOVE DFHPROTI TO MSGA
       GO TO Ø1ØØ-SEND-MAP.

*****************************************************************
* DETERMINE WHAT OTHER CICS REGIONS ARE CONNECTED TO THIS CICS  *
* REGION. LOOK FOR ALL CROSS MEMORY (XM) CONNECTIONS THAT ARE   *
* ACQUIRED.                                                     *
*****************************************************************
     EXEC CICS INQUIRE CONNECTION
               START
               NOHANDLE
               END-EXEC.
     MOVE Ø TO WS-SUB1.

 11ØØ-INQ-CONNECTIONS.

     EXEC CICS INQUIRE NEXT
               CONNECTION(WS-WORK-CONN)
               NETNAME(WS-WORK-NET)
               ACCESSMETHOD(WS-WORK-ACC)
               CONNSTATUS(WS-WORK-SRV)
               NOHANDLE
               END-EXEC.

     IF EIBRESP > Ø
         GO TO 111Ø-CONN-END.

*****************************************************************
* TEST FOR CONNECTIONS THAT ARE CROSS MEMORY (XM) AND ACQUIRED. *
* SAVE THE SYSIDS OF ALL CONNECTIONS THAT QUALIFY AS TARGETS    *
* FOR A REMOTE PROGRAM LINK.                                    *
*****************************************************************
     IF WS-WORK-ACC NOT = 123
       GO TO 11ØØ-INQ-CONNECTIONS.

     IF WS-WORK-SRV NOT = 69
       GO TO 11ØØ-INQ-CONNECTIONS.

     ADD 1 TO WS-SUB1.
     MOVE WS-WORK-CONN TO WS-CONN-ID(WS-SUB1).
     MOVE WS-WORK-NET  TO WS-CONN-NET(WS-SUB1).
     GO TO 11ØØ-INQ-CONNECTIONS.

 111Ø-CONN-END.

     EXEC CICS INQUIRE CONNECTION
               END
               NOHANDLE
               END-EXEC.

 15ØØ-LOCAL-PROCESS.
*****************************************************************
*   DO LOCAL INQUIRIES ABOUT THE REQUESTED RESOURCE.            *
*****************************************************************
     EXEC CICS ASSIGN
           APPLID(WS-NATIVE-NET)
           SYSID(WS-NATIVE-ID)
           END-EXEC.

     IF WS-SEARCH-TYPE = 'TRAN'
       PERFORM 16ØØ-TRAN-INQ THRU 16ØØ-EXIT
     ELSE
     IF WS-SEARCH-TYPE = 'PROG'
       PERFORM 17ØØ-PROG-INQ THRU 17ØØ-EXIT
     ELSE
       PERFORM 18ØØ-FILE-INQ THRU 18ØØ-EXIT.

     GO TO 2ØØØ-SERVER-LINK.

 16ØØ-TRAN-INQ.
*****************************************************************
*  THE TRANSACTION IS NOT DEFINED TO THE LOCAL REGION IF THE    *
*  FOLLOWING COMMAND RETURNS WITH A TRANSIDERR.                 *
*****************************************************************
     EXEC CICS INQUIRE TRANSACTION(TRANAMEO)
               PROGRAM(WS-WORK-PROG)
               REMOTESYSTEM(WS-WORK-RSYS)
               REMOTENAME(WS-WORK-RTRAN)
               NOHANDLE
               END-EXEC.

     IF EIBRESP = DFHRESP(TRANSIDERR)
       GO TO 16ØØ-EXIT.

     IF WS-WORK-PROG NOT = SPACE
       MOVE WS-NATIVE-ID   TO WS-DT-SYSID
       MOVE WS-NATIVE-NET  TO WS-DT-NETNM
       MOVE TRANAMEO       TO WS-DT-TRAN
       MOVE WS-WORK-PROG   TO WS-DT-PROG
       MOVE WS-DETAIL-TRAN TO INQTML1O
     ELSE
       MOVE WS-NATIVE-ID   TO WS-RDT-SYSID
       MOVE WS-NATIVE-NET  TO WS-RDT-NETNM
       MOVE TRANAMEO       TO WS-RDT-TRAN
       MOVE WS-WORK-RSYS   TO WS-RDT-TARG
       IF WS-WORK-RTRAN NOT = TRANAMEO
         MOVE WS-WORK-RTRAN TO WS-RTRAN-NAME
         MOVE WS-REMOTETRAN-MSG TO WS-RDT-MSG
         MOVE WS-RDETAIL-TRAN TO INQTML1O
       ELSE
         MOVE WS-RDETAIL-TRAN TO INQTML1O.

     IF COMM-ID = 'CLIENT'
       EXEC CICS SEND MAP('INQTML') MAPSET('INQTMØ2')
                 FROM(INQTMLO)
                 ACCUM
                 END-EXEC
     ELSE
       MOVE INQTML1O TO COMM-DETAIL-LINE.

 16ØØ-EXIT.
     EXIT.

 17ØØ-PROG-INQ.
******************************************************************
*  THE PROGRAM IS NOT DEFINED TO THE LOCAL REGION IF THE         *
*  FOLLOWING COMMAND RETURNS WITH A PGMIDERR.                    *
******************************************************************
     EXEC CICS INQUIRE PROGRAM(PRGNAMEO)
               REMOTESYSTEM(WS-WORK-RSYS)
               REMOTENAME(WS-WORK-PROG)
               LANGUAGE(WS-WORK-LANG)
               PROGTYPE(WS-WORK-PTYPE)
               COBOLTYPE(WS-WORK-CTYPE)
               NOHANDLE
               END-EXEC.

     IF EIBRESP = DFHRESP(PGMIDERR)
       GO TO 17ØØ-EXIT.

******************************************************************
*  SET UP LOCAL PROGRAM INFORMATION.                             *
******************************************************************

     IF WS-WORK-PTYPE = 155
       MOVE 'ASM' TO WS-DP-LANG
       MOVE '(MAP)' TO WS-DP-LTYPE
     ELSE
     IF WS-WORK-PTYPE = 156
       MOVE 'ASM' TO WS-DP-LANG
       MOVE '(PARTITIONSET)' TO WS-DP-LTYPE
     ELSE
     IF WS-WORK-PTYPE = 154
       IF WS-WORK-LANG = 149
         MOVE ' C ' TO WS-DP-LANG
       ELSE
       IF WS-WORK-LANG = 152 OR 153
         MOVE 'PL1' TO WS-DP-LANG
       ELSE
       IF WS-WORK-LANG = 151
         MOVE 'COB' TO WS-DP-LANG
         IF WS-WORK-CTYPE = 375
           MOVE '(COBOLII)' TO WS-DP-LTYPE
         ELSE
         IF WS-WORK-CTYPE = 377
           MOVE '(LE37Ø)' TO WS-DP-LTYPE.

******************************************************************
```

```
*   DECIDE WHICH DETAIL LINE TO USE, LOCAL OR REMOTE.           *
*****************************************************************
     IF WS-WORK-RSYS  = SPACE
       MOVE PRGNAMEO        TO WS-DP-PROG
       MOVE WS-NATIVE-ID    TO WS-DP-SYSID
       MOVE WS-NATIVE-NET   TO WS-DP-NETNM
       MOVE WS-DETAIL-PROG  TO INQTML1O
     ELSE
       MOVE WS-NATIVE-ID    TO WS-RDP-SYSID
       MOVE WS-NATIVE-NET   TO WS-RDP-NETNM
       MOVE PRGNAMEO        TO WS-RDP-PROG
       MOVE WS-WORK-RSYS    TO WS-RDP-TARG
       MOVE WS-RDETAIL-PROG TO INQTML1O.

     IF COMM-ID = 'CLIENT'
       EXEC CICS SEND MAP('INQTML') MAPSET('INQTMØ2')
                 FROM(INQTMLO)
                 ACCUM
                 END-EXEC
     ELSE
       MOVE INQTML1O TO COMM-DETAIL-LINE.

 17ØØ-EXIT.
     EXIT.

 18ØØ-FILE-INQ.
*****************************************************************
*  THE FILE IS NOT DEFINED TO THE LOCAL REGION IF THE FOLLOWING *
*  COMMAND RETURNS WITH A FILENOTFOUND CONDITION                *
*****************************************************************
     EXEC CICS INQUIRE FILE(FILNAMEO)
               REMOTESYSTEM(WS-WORK-RSYS)
               REMOTENAME(WS-WORK-FILE)
               OPENSTATUS(WS-WORK-OPEN)
               ADD(WS-WORK-ADD)
               UPDATE(WS-WORK-UPDATE)
               DELETE(WS-WORK-DELETE)
               READ(WS-WORK-READ)
               BROWSE(WS-WORK-BROWSE)
               NOHANDLE
               END-EXEC.

     IF EIBRESP = DFHRESP(FILENOTFOUND)
       GO TO 18ØØ-EXIT.
*****************************************************************
*  SET UP LOCAL FILE INFORMATION.                               *
*****************************************************************
     IF WS-WORK-RSYS  = SPACE
       MOVE FILNAMEO       TO WS-DF-FILE
       MOVE WS-NATIVE-ID   TO WS-DF-SYSID
       MOVE WS-NATIVE-NET  TO WS-DF-NETNM.
```

```
****************************************************************
*  DECIDE IF THE FILE HAS ANY MODIFICATION ATTRIBUTES, OR IF   *
*  THE FILE HAS ONLY READ-ONLY ATTRIBUTES.                     *
****************************************************************
       IF WS-WORK-ADD = 41
         MOVE 'A MODIFIABLE' TO WS-DF-FUNC
       ELSE
       IF WS-WORK-UPDATE = 37
         MOVE 'A MODIFIABLE' TO WS-DF-FUNC
       ELSE
       IF WS-WORK-DELETE = 43
         MOVE 'A MODIFIABLE' TO WS-DF-FUNC
       ELSE
       IF WS-WORK-READ = 35
         MOVE 'A READ ONLY' TO WS-DF-FUNC
       ELSE
       IF WS-WORK-BROWSE = 39
         MOVE 'A READ ONLY' TO WS-DF-FUNC.

****************************************************************
*  IF THE FILE IS OPEN, ADD A TRAILER MESSAGE.                 *
****************************************************************
     IF WS-WORK-OPEN = 18
       MOVE ' (OPEN)' TO WS-DF-MSG.

****************************************************************
*   DECIDE WHICH DETAIL LINE TO USE, LOCAL OR REMOTE.          *
****************************************************************
     IF WS-WORK-RSYS  = SPACE
       MOVE WS-DETAIL-FILE TO INQTML1O
     ELSE
       MOVE WS-NATIVE-ID    TO WS-RDF-SYSID
       MOVE WS-NATIVE-NET   TO WS-RDF-NETNM
       MOVE FILNAMEO        TO WS-RDF-FILE
       MOVE WS-WORK-RSYS    TO WS-RDF-TARG
       MOVE WS-RDETAIL-FILE TO INQTML1O.

     IF COMM-ID = 'CLIENT'
     EXEC CICS SEND MAP('INQTML') MAPSET('INQTMØ2')
               FROM(INQTMLO)
               ACCUM
               END-EXEC
     ELSE
       MOVE INQTML1O TO COMM-DETAIL-LINE.

 18ØØ-EXIT.
     EXIT.

 2ØØØ-SERVER-LINK.
****************************************************************
```

```
*  IF THERE ARE NO ACTIVE CONNECTIONS, GO TO THE DISPLAY SECTION*
*****************************************************************
     IF WS-SUB1 < 1
       GO TO 6ØØØ-DISPLAY-INFO.

     MOVE WS-CONN-ID(WS-SUB1)  TO WS-WORK-ID.
     MOVE 'SERVER' TO WS-COMM-ID.
     MOVE SPACE TO WS-COMM-DLINE.
     MOVE 1Ø3 TO WS-LENGTH.

     EXEC CICS LINK PROGRAM('INQT1ØØ')
               SYSID(WS-WORK-ID)
               COMMAREA(WS-COMM)
               LENGTH(WS-LENGTH)
               NOHANDLE
               END-EXEC.

     IF WS-COMM-DLINE NOT = SPACE
       MOVE WS-COMM-DLINE TO INQTML1O
       EXEC CICS SEND MAP('INQTML') MAPSET('INQTMØ2')
                 FROM(INQTMLO)
                 ACCUM
                 END-EXEC.

     SUBTRACT 1 FROM WS-SUB1.

     GO TO 2ØØØ-SERVER-LINK.

 5ØØØ-PROCESS-SERVER.

     EXEC CICS ASSIGN
           APPLID(WS-NATIVE-NET)
           SYSID(WS-NATIVE-ID)
           END-EXEC.

     IF COMM-TRAN NOT = SPACE
       MOVE COMM-TRAN TO TRANAMEO
       PERFORM 16ØØ-TRAN-INQ THRU 16ØØ-EXIT
     ELSE
     IF COMM-PROG NOT = SPACE
       MOVE COMM-PROG TO PRGNAMEO
       PERFORM 17ØØ-PROG-INQ THRU 17ØØ-EXIT
     ELSE
     IF COMM-FILE NOT = SPACE
       MOVE COMM-FILE TO FILNAMEO
       PERFORM 18ØØ-FILE-INQ THRU 18ØØ-EXIT.

     GO TO 9999-END.

 6ØØØ-DISPLAY-INFO.
```

```
     EXEC CICS SEND MAP('INQTMF') MAPSET('INQTM02')
               MAPONLY
               ACCUM
               ERASE
               END-EXEC.

     EXEC CICS SEND PAGE
               END-EXEC.

     GO TO 0200-RETURN.

 9999-END.
*****************************************************************
**  IF THIS PROGRAM IS SERVICING THE USER'S TERMINAL, CLEAR THE *
**  SCREEN                                                      *
*****************************************************************
     IF COMM-ID = 'CLIENT'
       EXEC CICS SEND CONTROL
                 ERASE
                 FREEKB
                END-EXEC.

     EXEC CICS RETURN
               END-EXEC.
```

INQTM01

```
***********************************************************************
         TITLE 'INQT - MAP FOR RESOURCE INFO INQUIRY'
INQTMS   DFHMSD MODE=INOUT,                                            X
               CTRL=(FREEKB,FRSET),                                    X
               LANG=COBOL,TIOAPFX=YES
INQTM01  DFHMDI SIZE=(24,80)
         DFHMDF POS=(2,26),LENGTH=28,                                  X
               INITIAL='CICS RESOURCE INQUIRY',                        X
               ATTRB=(PROT,BRT)
         DFHMDF POS=(3,25),LENGTH=30,                                  X
               INITIAL='***********************',                      X
               ATTRB=(PROT,BRT)
         DFHMDF POS=(8,27),LENGTH=10,                                  X
               INITIAL='TRANID  : '
TRANAME  DFHMDF POS=(8,41),LENGTH=4,                                   X
               ATTRB=(UNPROT,IC)
         DFHMDF POS=(8,46),LENGTH=1,                                   X
               ATTRB=(PROT,DRK)
         DFHMDF POS=(10,27),LENGTH=10,                                 X
               INITIAL='PROGRAM : '
PRGNAME  DFHMDF POS=(10,41),LENGTH=8,                                  X
               ATTRB=(UNPROT)
         DFHMDF POS=(10,50),LENGTH=1,                                  X
```

```
               ATTRB=(PROT,DRK)
         DFHMDF POS=(12,27),LENGTH=1Ø,                                 X
               INITIAL='FILE    : '
FILNAME  DFHMDF POS=(12,41),LENGTH=8,                                  X
               ATTRB=(UNPROT)
         DFHMDF POS=(12,5Ø),LENGTH=1,                                  X
               ATTRB=(PROT,DRK)
MSG      DFHMDF POS=(2Ø,2Ø),LENGTH=46,                                 X
               INITIAL='       PRESS CLEAR OR PF3 TO EXIT'
         DFHMSD TYPE=FINAL
         END
```

INQTM02

```
***********************************************************************
         TITLE 'INQD - MAP FOR RESOURCE INFO DISPLAY'
INQTMØ2  DFHMSD MODE=OUT,                                              X
               CTRL=(FREEKB,FRSET),                                    X
               LANG=COBOL,TIOAPFX=YES
INQTML   DFHMDI SIZE=(1,8Ø)
INQTML1  DFHMDF POS=(1,2),LENGTH=77
INQTMF   DFHMDI SIZE=(2,8Ø)
         DFHMDF POS=(1,2),LENGTH=1
INQTMF1  DFHMDF POS=(2,2),LENGTH=48,                                   X
               INITIAL='* PF5 FOR NEW INQUIRY - PF3 OR CLEAR TO EXIT *'
         DFHMSD TYPE=FINAL
         END
```

INQTRDO

```
DEFINE MAPSET(INQTMØ1) GROUP(INQTGRP)
       RESIDENT(NO) USAGE(NORMAL) USELPACOPY(NO) STATUS(ENABLED)
DEFINE MAPSET(INQTMØ2) GROUP(INQTGRP)
       RESIDENT(NO) USAGE(NORMAL) USELPACOPY(NO) STATUS(ENABLED)
DEFINE PROGRAM(INQT1ØØ) GROUP(INQTGRP)
       LANGUAGE(COBOL) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
       USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
       EXECKEY(USER) EXECUTIONSET(FULLAPI)
DEFINE TRANSACTION(INQT) GROUP(INQTGRP)
       PROGRAM(INQT1ØØ) TWASIZE(Ø) PROFILE(DFHCICST) STATUS(ENABLED)
       TASKDATALOC(ANY) TASKDATAKEY(USER) STORAGECLEAR(NO)
       RUNAWAY(SYSTEM) SHUTDOWN(DISABLED) ISOLATE(YES) DYNAMIC(NO)
       PRIORITY(1) TRANCLASS(DFHTCLØØ) DTIMOUT(NO) INDOUBT(BACKOUT)
       RESTART(NO) SPURGE(YES) TPURGE(YES) DUMP(YES) TRACE(YES)
       CONFDATA(NO) RESSEC(NO) CMDSEC(NO)
```

Kyle Keenan
Centura (USA)

DL/I database display and control facility – part 2

This month we complete the code for the easy-to-use facility that enables developers to display the status of their databases and to start and stop them in a similar manner to that with CEMT.

```
         CLC   SUFFIX_LENGTH,ZERO      ANY SUFFIX PROVIDED
         BE    WE_WANT_THIS            NO - ACCEPT THIS ONE
*
* REVERSE THE DBNAME IN ORDER TO CHECK THE SUFFIX
*
         LA    R2,DBNAME+7             GET ADDRESS OF END OF DBNAME
         LA    R3,8                    GET LENGTH OF DNAME
         LA    R4,DBNAME_REVERSED      GET ADDRESS OF REVERSE DBNAME
         MVC   DBNAME_REVERSED,SPACES  SPACE OUT REVERSE DBNAME
MOVE_CHAR DS ØH
         CLI   Ø(R2),C' '              IS THIS CHAR A SPACE
         BE    DONT_MOVE               YES - BOUNCE ROUND IT
         MVC   Ø(1,R4),Ø(R2)           MOVE THE CURRENT CHAR
         LA    R4,1(R4)                POINT TO NEXT BYTE
DONT_MOVE DS    ØH
         BCTR  R2,Ø                    POINT TO NEXT DBNAME BYTE
         BCT   R3,MOVE_CHAR            ANY MORE - GO ROUND AGAIN
         LH    R1,SUFFIX_LENGTH        GET SUFFIX LENGTH
         BCTR  R1,Ø                    -1 FOR EX
         EX    R1,EXAMINE_ENDING       DO WE WANT THIS ONE
         BNE   GET_NEXT_MESSAGE        NO - GO TO GET NEXT MESSAGE
WE_WANT_THIS DS ØH
         LH    R1,IOLEN                GET RETURNED LENGTH
         LA    R7,4                    SET LENGTH OF RDW
         SR    R1,R7                   SUBTRACT FROM LENGTH
         STH   R1,LENGTH               SAVE RESULT
         EXEC CICS WRITEQ TS QUEUE(TSQNAME) FROM(IOTEXT) MAIN          X
               NUMITEMS(ITEMS) LENGTH(LENGTH)
         B     GET_NEXT_MESSAGE        GO AND LEAVE
MESSAGE_LINE DS ØH
*        PROCESS MESSAGE LINE
         B     GET_NEXT_MESSAGE
*
GET_NEXT_MESSAGE DS ØH
         MVC   AIB_CMD,RCMD            SET AIB COMMAND TO RCMD
         CLI   LAST_SEGMENT,C'N'       HAVE WE HAD LAST SEGMENT
         BE    CALL_AIB_FOR_DISPLAY_COMMAND
*
TERM_PSB DS    ØH
         CALL  ASMTDLI,                                                X
               (TERM),                                                 X
               VL,                                                     X
               MF=(E,CALLLIST)
DO_THE_DISPLAY_END DS ØH
         BR    R8
*-------------------------------------------------------------------- *
*                                                                     *
*        ISSUE AN AIB COMMAND AND DECODE THE RETURN/REASON CODE       *
*                                                                     *
*-------------------------------------------------------------------- *
ISSUE_AIB_COMMAND DS ØH
         CALL  AIBTDLI,                                                X
               (AIB_CMD,AIBAREA,IOAREA),                               X
               VL,                                                     X
               MF=(E,CALLLIST)
*

         LA    R3,AIBAREA
         EXEC CICS ENTER TRACENUM(2) FROM(DFSAIB) FROMLENGTH(AIB_LEN)  X
               RESOURCE('SPGDBDSP') RESP(RESPONSE) RESP2(REASON)
         EXEC CICS ENTER TRACENUM(3) FROM(IOAREA) FROMLENGTH(IOA_LEN)  X
               RESOURCE('SPGDBDSP') RESP(RESPONSE) RESP2(REASON)
*
         LA    R1,GMSG_RRT         GMSG RR TABLE ADDRESS
         LA    R15,GMSG_RRT_LEN    TABLE ENTRY LENGTH
         LA    RØ,GMSG_RRT_CNT     NUMBER OF ENTRIES
GMSG_RRT_LOOP DS ØH
         CLC   AIBRETRN(8),Ø(R1)   RETURN/REASON MATCH
         BE    GOT_RET_REAS        YES, CONTINUE
         BL    UNKNOWN_RET_REAS    UNEXPECTED RETURN CODES
         AR    R1,R15              NEXT ENTRY ADDRESS
         BCT   RØ,GMSG_RRT_LOOP    CHECK NEXT ENTRY
         B     UNKNOWN_RET_REAS    UNEXPECTED RETURN CODES
         SPACE
GOT_RET_REAS DS ØH
         L     R15,8(R1)           GET BRANCH ADDRESS
         B     ISSUE_AIB_COMMAND_END
UNKNOWN_RET_REAS DS ØH
         LA    R15,12
ISSUE_AIB_COMMAND_END DS ØH
         BR    R9
*-------------------------------------------------------------------- *
*                                                                     *
*        CLEAR MAP AREA AND GET DATE, TIME ETC                        *
*                                                                     *
*-------------------------------------------------------------------- *
CLEAR_MAP DS ØH
         LA    R2,DDDCMØ1O      POINT AT RECEIVING AREA
         LA    R3,DDDCMØ1L      SET ITS LENGTH
         XR    R4,R4            SET DUMMY FROM ADDRESS
         XR    R5,R5            SET DUMMY FROM ADDRESS
         MVCL  R2,R4            BLANK OUT THE AREA
         EXEC CICS ASKTIME ABSTIME(ABSTIME)
         EXEC CICS FORMATTIME ABSTIME(ABSTIME)                         X
               DDMMYYYY(DATEO) DATESEP('/')                            X
               TIME(TIMEO) TIMESEP(':')
         EXEC CICS ASSIGN APPLID(CICSO)
         MVC   TERMIDO,EIBTRMID
         BR    R1Ø
*-------------------------------------------------------------------- *
*                                                                     *
*        RETURN CODE / REASON CODE TABLE                              *
*                                                                     *
*        SEE IMS/ESA V5 APPLICATION PROGRAMMING : DATABASE MANAGER    *
*        FOR MORE INFORMATION                                         *
*                                                                     *
*-------------------------------------------------------------------- *
GMSG_RRT DS ØF
         DC    XL4'ØØØØ',XL4'ØØØØ',A(Ø)        CALL COMPLETED OK
GMSG_RRT_LEN EQU *-GMSG_RRT
         DC    XL4'ØØØ4',XL4'ØØØ4',A(4)        LAST SEGMENT RETURNED
         DC    XL4'ØØØ4',XL4'ØØ14',A(8)        NO MORE MESSAGES
         DC    XL4'ØØØ4',XL4'ØØ18',A(8)        NO MORE SEGMENTS
GMSG_RRT_CNT EQU ((*-GMSG_RRT)/GMSG_RRT_LEN)
*-------------------------------------------------------------------- *
*                                                                     *
*        LINE TYPE TABLE                                              *
*                                                                     *
*-------------------------------------------------------------------- *
LINE_TYPE_TABLE DS ØF
         DC    C'Ø',A(Ø)
LINE_TYPE_LEN EQU *-LINE_TYPE_TABLE
         DC    C'1',A(Ø)
         DC    C'2',A(Ø)
         DC    C'3',A(Ø)
         DC    C'4',A(Ø)
         DC    C'5',A(4)
         DC    C'6',A(4)
         DC    C'7',A(8)
         DC    C'8',A(8)
         DC    C'9',A(8)
LINE_TYPE_CNT EQU ((*-LINE_TYPE_TABLE)/LINE_TYPE_LEN)
*-------------------------------------------------------------------- *
*                                                                     *
*        LITERALS                                                     *
*                                                                     *
*-------------------------------------------------------------------- *
ZERO     DC    H'Ø'
ALL_DATABASES DC H'-1'
UIB_LEN  DC   AL2(UIBLEN)
AIB_LEN  DC   AL2(AIBLL)
IOA_LEN  DC   AL2(LIOAREA)
*
PCB      DC   CL4'PCB'
```

```
ICMD     DC   CL4'ICMD'
RCMD     DC   CL4'RCMD'
TERM     DC   CL4'TERM'
PSBNAME  DC   CL8'DFHDBMP'
SYSSERVE DC   CL8'IOPCB'
SPACES   DC   CL11' '
STAR     DC   C'*       '
CMDDIS   DC   CL11'/DIS DB ALL'
NOTSTOP  DC   CL11'NOT STOPPED'
NOTSTART DC   CL11'NOT STARTED'
STOP     DC   CL11'STOPPED'
START    DC   CL11'STARTED'
WRONG_KEY       DC CL8Ø'THE KEY YOU PRESSED HAS NO FUNCTION'
PSB_SCHED_ERROR DC CL8Ø'PROBLEM WITH DBCTL, TRY AGAIN LATER'
CANT_PAGE_FWD   DC CL8Ø'NO MORE TO SHOW'
CANT_PAGE_BACK  DC CL8Ø'YOU ARE ON THE FIRST PAGE'
*---------------------------------------------------------------------*
*                                                                     *
*        THAT'S ALL FOLKS                                             *
*                                                                     *
*---------------------------------------------------------------------*
         END
```

SPGDBSP LISTING

```
*---------------------------------------------------------------------*
*                          S P G D B S P                              *
*                          = = = = = = =                              *
*                                                                     *
* THIS ROUTINE IMPLEMENTS A REPLACEMENT FOR THE FOLLOWING CEMT         *
* COMMANDS WHICH ARE NOT AVAILABLE FOR DATABASES ACCESSED VIA DBCTL   *
*                                                                     *
*       CEMT SET DLIDATABASE(.......) START                           *
*       CEMT SET DLIDATABASE(.......) STOP                            *
*                                                                     *
*---------------------------------------------------------------------*
         DFHREGS
*---------------------------------------------------------------------*
*                                                                     *
* COMMAREA                                                            *
*                                                                     *
*---------------------------------------------------------------------*
         USING COMMAREA,R2
COMMAREA DSECT
FUNCTION DS    CL1       S -> START DATABASE, P -> STOP DATABASE
DATABASE DS    CL8       NAME OF DATABASE TO BE STARTED OR STOPPED
RESULT   DS    CL1       OUTCOME - SPACE => ACTION SUCCESSFUL
*                                      F => FUNCTION INVALID
*                                      D => NO DATABASE NAME
*                                      P => PSB SCHEDULE FAILED
*                                      M => IMS MESSAGE ISSUED
*                                      N => UNEXPECTED RETURN CODE
*
*-------------------------------------------------------------------- *
*                                                                     *
*        IMS AIB LAYOUT                                               *
*                                                                     *
*-------------------------------------------------------------------- *
         USING DFSAIB,R3
         DFSAIB
*-------------------------------------------------------------------- *
*                                                                     *
*        DL/I UIB LAYOUT                                              *
*                                                                     *
*-------------------------------------------------------------------- *
         USING UIB,R4
         DLIUIB
*-------------------------------------------------------------------- *
*                                                                     *
*        HERE WE GO                                                   *
*                                                                     *
*-------------------------------------------------------------------- *
SPGDBSP  DFHEIENT EIBREG=11,CODEREG=12,DATAREG=13
*-------------------------------------------------------------------- *
*                                                                     *
*        VALIDATE THE COMMAREA PARAMETERS                             *
*                                                                     *
*-------------------------------------------------------------------- *
         CLC   EIBCALEN,TEN            HAVE WE GOT 1Ø BYTES OF COMMAREA
         BNE   THE_END                 NO - LEAVE NOW
         L     R2,DFHEICAP             GET THE COMMAREA ADDRESS
         MVI   RESULT,C' '             SET RESULT TO OK
         CLI   FUNCTION,C'S'           VALIDATE
         BE    FUNCTION_OK             THE
         CLI   FUNCTION,C'P'           FUNCTION
         BE    FUNCTION_OK             PARAMETER
         MVI   RESULT,C'F'             SET RESULT
         B     THE_END                 AND LEAVE
*
FUNCTION_OK DS ØH
         CLC   DATABASE,SPACES         VALIDATE DATABASE
         BNE   DATABASE_OK             NAME PARAMETER
         MVI   RESULT,C'D'             SET RESULT
         B     THE_END                 AND LEAVE
*-------------------------------------------------------------------- *
*                                                                     *
*        INITIALIZE THE AIB                                           *
*                                                                     *
*-------------------------------------------------------------------- *
```

```
DATABASE_OK DS ØH
         LA    R3,AIBAREA
         MVC   AIBID,=CL8'DFSAIB'      INITIALIZE ...
         MVC   AIBLEN,=A(AIBLL)        .. DFSAIB ...
         MVC   AIBOALEN,=A(LIOAREA)    .. CONTROL BLOCK
*---------------------------------------------------------------------*
*                                                                     *
*        DO THE PCB CALL                                              *
*                                                                     *
*---------------------------------------------------------------------*
         CALL  ASMTDLI,                                                X
               (PCB,PSBNAME,UIBPTR,SYSSERVE),                          X
               VL,                                                     X
               MF=(E,CALLLIST)
         L     R4,UIBPTR               GET UIB ADDRESS
*
         EXEC CICS ENTER TRACENUM(1) FROM(UIB) FROMLENGTH(UIB_LEN)     X
               RESOURCE('SPGDBSP') RESP(RESPONSE) RESP2(REASON)
*
         CLI   UIBFCTR,X'ØØ'           CHECK RETURN CODE
         BE    PSB_SCHEDULED           ZERO - WE'RE OK
         MVI   RESULT,C'P'             SET RESULT
         B     THE_END                 THEN LEAVE NOW
*---------------------------------------------------------------------*
*                                                                     *
*        SET UP THE IO AREA FOR THE AIB CALL                          *
*                                                                     *
*---------------------------------------------------------------------*
PSB_SCHEDULED DS ØH
         MVC   IOLEN,=Y(L'IOTEXT,Ø)    SET COMMAND LENGTH
         MVI   IOTEXT,C' '
         MVC   IOTEXT+1(L'IOTEXT-1),IOTEXT
         CLI   FUNCTION,C'P'           IF IT'S NOT STOP
         BNE   NOT_STOP                THEN GO TO SET UP START
         MVC   IOCMD,CMDDBR            MOVE IN /DBR COMMAND
         MVC   IONOFEOV,NOFEOV         AND NOFEOV OPTION
         B     SET_DATABASE_NAME       GO AND SET DB NAME
*
NOT_STOP DS    ØH
         MVC   IOCMD,CMDSTA            SET /STA COMMAND
*---------------------------------------------------------------------*
*                                                                     *
*        ISSUE THE AIB CALL FOR EITHER /STA OR /DBR                   *
*                                                                     *
*---------------------------------------------------------------------*
SET_DATABASE_NAME DS ØH
         MVC   IODBNAME,DATABASE       MOVE IN DATABASE NAME
         MVC   IOLIT,CMDLIT            AND LITERAL
         MVC   AIB_CMD,ICMD            SET AIB COMMAND TO ICMD
         BAL   R1Ø,ISSUE_AIB_COMMAND
```

```
*-------------------------------------------------------------------- *
*                                                                     *
*        WAIT A BIT FOR IT TO COMPLETE                                *
*                                                                     *
*-------------------------------------------------------------------- *
         EXEC CICS DELAY FOR SECONDS(2)
*-------------------------------------------------------------------- *
*                                                                     *
*        ISSUE THE AIB CALL FOR /DIS TO SEE IF OUR PREVIOUS CALL      *
*        WORKED                                                       *
*                                                                     *
*        FOR A START REQUEST WE ISSUE /DIS DB ALLOCS AND FOR A        *
*        STOP WE ISSUE /DIS DB STOPPED                                *
*                                                                     *
*        WE THEN SCAN THE RESULTING MESSAGES LOOKING FOR OUR D/B      *
*                                                                     *
*-------------------------------------------------------------------- *
         MVI   IOTEXT,C' '             CLEAR THE IO AREA
         MVC   IOTEXT+1(L'IOTEXT-1),IOTEXT
         MVI   LAST_SEGMENT,C'N'       SET LAST SEGMENT FLAG
         MVI   RESULT,C'N'             SET RESULT
         CLI   FUNCTION,C'S'           START ?
         BE    SET_DIS_ALLOCS          GO AND SET UP COMMAND
         MVC   IOTEXT(L'DISSTOP),DISSTOP
         B     CALL_AIB_FOR_DISPLAY_COMMAND
SET_DIS_ALLOCS DS ØH
         MVC   IOTEXT(L'DISALLOC),DISALLOC
CALL_AIB_FOR_DISPLAY_COMMAND DS ØH
         BAL   R1Ø,ISSUE_AIB_COMMAND
*-------------------------------------------------------------------- *
*                                                                     *
*        ACT UPON THE RETURN CODE FROM THE AIB CALL                   *
*                                                                     *
*-------------------------------------------------------------------- *
         B     CHECK_DISPLAY_RETURN_CODE(R15)
CHECK_DISPLAY_RETURN_CODE DS ØH
         B     CHECK_MESSAGE_FROM_DISPLAY
         B     LAST_SEGMENT_RETURNED
         B     TERM_PSB
         B     TERM_PSB
LAST_SEGMENT_RETURNED DS ØH
         MVI   LAST_SEGMENT,C'Y'       SET LAST SEGMENT FLAG
CHECK_MESSAGE_FROM_DISPLAY DS ØH
         CLI   IOTEXT,C'D'             IS THIS A DISPLAY SEGMENT?
         BNE   GET_NEXT_MESSAGE        NO - NOT INTERESTED
         LA    R1,LINE_TYPE_TABLE
         LA    R15,LINE_TYPE_LEN       TABLE ENTRY LENGTH
         LA    RØ,LINE_TYPE_CNT        NUMBER OF ENTRIES
LINE_TYPE_LOOP DS ØH
         CLC   IOTEXT+1(1),Ø(R1)       MATCH
         BE    GOT_LINE_TYPE           YES, CONTINUE
         AR    R1,R15                  NEXT ENTRY ADDRESS
         BCT   RØ,LINE_TYPE_LOOP       CHECK NEXT ENTRY
         LA    R15,8                   UNEXPECTED LINE TYPE
         B     ACT_ON_LINE_TYPE
GOT_LINE_TYPE DS ØH
         L     R15,4(R1)               GET BRANCH ADDRESS
ACT_ON_LINE_TYPE DS ØH
         B     PROCESS_LINE_TYPE(R15)  GO TO APPROPRIATE PLACE
PROCESS_LINE_TYPE DS ØH
         B     DATA_LINE               LINE TYPES ØØ - 49
         B     MESSAGE_LINE            LINE TYPES 5Ø - 69
         B     GET_NEXT_MESSAGE        LINE TYPES 7Ø - 99
DATA_LINE DS   ØH
         CLC   DATABASE,IOTEXT+4
         BNE   GET_NEXT_MESSAGE
         MVI   RESULT,C' '
         B     TERM_PSB
MESSAGE_LINE DS ØH
         MVI   RESULT,C'M'
         B     TERM_PSB
*
GET_NEXT_MESSAGE DS ØH
         MVC   AIB_CMD,RCMD            SET AIB COMMAND TO RCMD
         CLI   LAST_SEGMENT,C'N'       HAVE WE HAD LAST SEGMENT
         BE    CALL_AIB_FOR_DISPLAY_COMMAND
*
TERM_PSB DS    ØH
         CALL  ASMTDLI,                                                X
               (TERM),                                                 X
               VL,                                                     X
               MF=(E,CALLLIST)
*
THE_END  DS    ØH
         EXEC CICS RETURN
*-------------------------------------------------------------------- *
*                                                                     *
*        ISSUE AN AIB COMMAND AND DECODE THE RETURN/REASON CODE       *
*                                                                     *
*-------------------------------------------------------------------- *
ISSUE_AIB_COMMAND DS ØH
         CALL  AIBTDLI,                                                X
               (AIB_CMD,AIBAREA,IOAREA),                               X
               VL,                                                     X
               MF=(E,CALLLIST)
*
         EXEC CICS ENTER TRACENUM(2) FROM(DFSAIB) FROMLENGTH(AIB_LEN)  X

               RESOURCE('SPGDBSP') RESP(RESPONSE) RESP2(REASON)
         EXEC CICS ENTER TRACENUM(3) FROM(IOAREA) FROMLENGTH(IOA_LEN)  X
```

```
               RESOURCE('SPGDBSP') RESP(RESPONSE) RESP2(REASON)
*
         LA    R1,GMSG_RRT         GMSG RR TABLE ADDRESS
         LA    R15,GMSG_RRT_LEN    TABLE ENTRY LENGTH
         LA    RØ,GMSG_RRT_CNT     NUMBER OF ENTRIES
GMSG_RRT_LOOP DS ØH
         CLC   AIBRETRN(8),Ø(R1)   RETURN/REASON MATCH
         BE    GOT_RET_REAS        YES, CONTINUE
         BL    UNKNOWN_RET_REAS    UNEXPECTED RETURN CODES
         AR    R1,R15              NEXT ENTRY ADDRESS
         BCT   RØ,GMSG_RRT_LOOP    CHECK NEXT ENTRY
         B     UNKNOWN_RET_REAS    UNEXPECTED RETURN CODES
         SPACE
GOT_RET_REAS DS ØH
         L     R15,8(R1)           GET BRANCH ADDRESS
         B     ISSUE_AIB_COMMAND_END
UNKNOWN_RET_REAS DS ØH
         LA    R15,12
ISSUE_AIB_COMMAND_END DS ØH
         BR    R1Ø
*
*--------------------------------------------------------------------- *
*                                                                      *
*        RETURN CODE / REASON CODE TABLE                               *
*                                                                      *
*--------------------------------------------------------------------- *
GMSG_RRT DS ØF
         DC    XL4'ØØØØ',XL4'ØØØØ',A(Ø)
GMSG_RRT_LEN EQU *-GMSG_RRT
         DC    XL4'ØØØ4',XL4'ØØØ4',A(4)
         DC    XL4'ØØØ4',XL4'ØØ14',A(8)
         DC    XL4'ØØØ4',XL4'ØØ18',A(8)
GMSG_RRT_CNT EQU ((*-GMSG_RRT)/GMSG_RRT_LEN)
*--------------------------------------------------------------------- *
*                                                                      *
*        LINE TYPE TABLE                                               *
*                                                                      *
*--------------------------------------------------------------------- *
LINE_TYPE_TABLE DS ØF
         DC    C'Ø',A(Ø)
LINE_TYPE_LEN EQU *-LINE_TYPE_TABLE
         DC    C'1',A(Ø)
         DC    C'2',A(Ø)
         DC    C'3',A(Ø)
         DC    C'4',A(Ø)
         DC    C'5',A(4)
         DC    C'6',A(4)
         DC    C'7',A(8)
         DC    C'8',A(8)
         DC    C'9',A(8)
LINE_TYPE_CNT EQU ((*-LINE_TYPE_TABLE)/LINE_TYPE_LEN)
```

```
*-------------------------------------------------------------------- *
*                                                                     *
*        LITERALS                                                     *
*                                                                     *
*-------------------------------------------------------------------- *
TEN      DC   AL2(1Ø)
ZERO     DC   A(Ø),A(Ø)
UIB_LEN  DC   AL2(UIBLEN)
AIB_LEN  DC   AL2(AIBLL)
IOA_LEN  DC   AL2(LIOAREA)
*
PCB      DC   CL4'PCB'
ICMD     DC   CL4'ICMD'
RCMD     DC   CL4'RCMD'
TERM     DC   CL4'TERM'
PSBNAME  DC   CL8'DFHDBMP'
SYSSERVE DC   CL8'IOPCB'
SPACES   DC   CL8'        '
CMDSTA   DC   CL4'/STA'
CMDDBR   DC   CL4'/DBR'
CMDLIT   DC   CL8'DATABASE '
NOFEOV   DC   CL6'NOFEOV'
D5       DC   C'D5'
*
DISSTOP  DC   C'/DIS DB STOPPED'
DISALLOC DC   C'/DIS DB ALLOCS'
*
*-------------------------------------------------------------------- *
*                                                                     *
*        WORKING STORAGE                                              *
*                                                                     *
*-------------------------------------------------------------------- *
         DFHEISTG
*
CALLLIST CALL  ,(,,,,,),MF=L
*
UIBPTR   DS    F                        UIB POINTER
*
RESPONSE DS    F                        RESP
REASON   DS    F                        RESP2
*
AIB_CMD  DS    CL4                      COMMAND FOR AIB CALL
*
IOAREA   DS    CL136                    IO AREA FOR AIB CALL
LIOAREA  EQU   *-IOAREA
         ORG   IOAREA
IOLEN    DS    CL4
IOTEXT   DS    CL132
         ORG   IOTEXT
IOCMD    DS    CL4
```

```
         DS    CL1
IOLIT    DS    CL8
         DS    CL1
IODBNAME DS    CL8
         DS    CL1
IONOFEOV DS    CL6
         ORG
*
AIBAREA  DC    (AIBLL)X'ØØ'            RESERVE SPACE FOR AIB
*
LAST_SEGMENT DS CL1
*--------------------------------------------------------------------- *
*                                                                      *
*        THAT'S ALL FOLKS                                              *
*                                                                      *
*--------------------------------------------------------------------- *
         END
```

DDDCM01 LISTING

```
         PRINT ON,NOGEN
DDDCMØ1  DFHMSD TYPE=MAP,LANG=ASM,MODE=INOUT,STORAGE=AUTO,SUFFIX=
DDDCMØ1  DFHMDI SIZE=(24,8Ø),CTRL=(FREEKB,FRSET),MAPATTS=(COLOR),      X
               DSATTS=(COLOR),COLUMN=1,LINE=1,DATA=FIELD,TIOAPFX=YES,  X
               CURSLOC=YES,OBFMT=NO
         DFHMDF POS=(1,1),LENGTH=6,INITIAL='Date :',ATTRB=(PROT,NORM),
               COLOR=TURQUOISE
* DATE                            DATE
DATE     DFHMDF POS=(1,8),LENGTH=1Ø,ATTRB=(PROT,NORM),COLOR=GREEN
         DFHMDF POS=(1,19),LENGTH=1,ATTRB=(PROT,NORM)
         DFHMDF POS=(1,24),LENGTH=32,                                  X
               INITIAL='DLI Database Display and
Control',ATTRB=(PROT,B*ØØØØØ12Ø
               RT),COLOR=YELLOW
         DFHMDF POS=(1,65),LENGTH=6,INITIAL='CICS
:',ATTRB=(PROT,NORM),*ØØØØØ14Ø
               COLOR=TURQUOISE
* CICS                            CICS
CICS     DFHMDF POS=(1,72),LENGTH=8,ATTRB=(PROT,NORM),COLOR=GREEN
         DFHMDF POS=(2,1),LENGTH=6,INITIAL='Time :',ATTRB=(PROT,NORM), X
               COLOR=TURQUOISE
* TIME                            TIME
TIME     DFHMDF POS=(2,8),LENGTH=8,ATTRB=(PROT,NORM),COLOR=GREEN
         DFHMDF POS=(2,17),LENGTH=1,ATTRB=(PROT,NORM)
         DFHMDF POS=(2,63),LENGTH=8,INITIAL='Termid
:',ATTRB=(PROT,NORM*ØØØØØ23Ø
               ),COLOR=TURQUOISE
* TERMID                          TERMID
TERMID   DFHMDF POS=(2,72),LENGTH=4,ATTRB=(PROT,NORM),COLOR=GREEN
```

```
         DFHMDF POS=(2,77),LENGTH=1,ATTRB=(PROT,NORM)
         DFHMDF POS=(3,80),LENGTH=19,INITIAL='Enter Database Name',    X
               ATTRB=(PROT,NORM),COLOR=TURQUOISE
         DFHMDF POS=(4,21),LENGTH=5,INITIAL='====>',ATTRB=(PROT,NORM), X
               COLOR=NEUTRAL
* DBNAME                          DBNAME
DBNAME   DFHMDF POS=(4,28),LENGTH=8,ATTRB=(UNPROT,NORM,IC,FSET),       X
               COLOR=GREEN
         DFHMDF POS=(4,37),LENGTH=33,                                  X
               INITIAL='(Can be generic : eg BK*N or
CM*)',ATTRB=(PROT,*00000360
               NORM),COLOR=TURQUOISE
         DFHMDF POS=(5,80),LENGTH=6,INITIAL='Cmds
:',ATTRB=(PROT,NORM),*00000380
               COLOR=TURQUOISE
         DFHMDF POS=(6,7),LENGTH=1,INITIAL='S',ATTRB=(PROT,BRT),       X
               COLOR=YELLOW
         DFHMDF POS=(6,9),LENGTH=5,INITIAL='Start',ATTRB=(PROT,NORM),  X
               COLOR=TURQUOISE
         DFHMDF POS=(6,15),LENGTH=1,INITIAL='P',ATTRB=(PROT,BRT),      X
               COLOR=YELLOW
         DFHMDF POS=(6,17),LENGTH=4,INITIAL='Stop',ATTRB=(PROT,NORM),  X
               COLOR=TURQUOISE
         DFHMDF POS=(7,80),LENGTH=22,INITIAL='Cmd  Database   Status', X
               ATTRB=(PROT,NORM),COLOR=TURQUOISE
         DFHMDF POS=(8,23),LENGTH=1,ATTRB=(PROT,NORM)
         DFHMDF POS=(8,25),LENGTH=1,ATTRB=(ASKIP,NORM)
* CMD1                            CMD1
CMD1     DFHMDF POS=(9,1),LENGTH=1,ATTRB=(UNPROT,NORM),COLOR=RED
         DFHMDF POS=(9,3),LENGTH=1,ATTRB=(PROT,NORM)
* NAME1                           NAME1
NAME1    DFHMDF POS=(9,5),LENGTH=8,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(9,14),LENGTH=1,ATTRB=(PROT,NORM)
* STATUS1                         STATUS1
STATUS1  DFHMDF POS=(9,16),LENGTH=49,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(9,66),LENGTH=0,ATTRB=(PROT,NORM)
* RESULT1                         RESULT1
RESULT1  DFHMDF POS=(9,67),LENGTH=11,ATTRB=(PROT,NORM),COLOR=GREEN
         DFHMDF POS=(9,79),LENGTH=1,ATTRB=(PROT,NORM)
* CMD2                            CMD2
CMD2     DFHMDF POS=(10,1),LENGTH=1,ATTRB=(UNPROT,NORM),COLOR=RED
         DFHMDF POS=(10,3),LENGTH=1,ATTRB=(PROT,NORM)
* NAME2                           NAME2
NAME2    DFHMDF POS=(10,5),LENGTH=8,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(10,14),LENGTH=1,ATTRB=(PROT,NORM)
* STATUS2                         STATUS2
STATUS2  DFHMDF POS=(10,16),LENGTH=49,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(10,66),LENGTH=0,ATTRB=(PROT,NORM)
* RESULT2                         RESULT2
RESULT2  DFHMDF POS=(10,67),LENGTH=11,ATTRB=(PROT,NORM),COLOR=GREEN
         DFHMDF POS=(10,79),LENGTH=1,ATTRB=(PROT,NORM)
```

```
* CMD3                            CMD3
CMD3     DFHMDF POS=(11,1),LENGTH=1,ATTRB=(UNPROT,NORM),COLOR=RED
         DFHMDF POS=(11,3),LENGTH=1,ATTRB=(PROT,NORM)
* NAME3                           NAME3
NAME3    DFHMDF POS=(11,5),LENGTH=8,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(11,14),LENGTH=1,ATTRB=(PROT,NORM)
* STATUS3                         STATUS3
STATUS3  DFHMDF POS=(11,16),LENGTH=49,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(11,66),LENGTH=Ø,ATTRB=(PROT,NORM)
* RESULT3                         RESULT3
RESULT3  DFHMDF POS=(11,67),LENGTH=11,ATTRB=(PROT,NORM),COLOR=GREEN
         DFHMDF POS=(11,79),LENGTH=1,ATTRB=(PROT,NORM)
* CMD4                            CMD4
CMD4     DFHMDF POS=(12,1),LENGTH=1,ATTRB=(UNPROT,NORM),COLOR=RED
         DFHMDF POS=(12,3),LENGTH=1,ATTRB=(PROT,NORM)
* NAME4                           NAME4
NAME4    DFHMDF POS=(12,5),LENGTH=8,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(12,14),LENGTH=1,ATTRB=(PROT,NORM)
* STATUS4                         STATUS4
STATUS4  DFHMDF POS=(12,16),LENGTH=49,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(12,66),LENGTH=Ø,ATTRB=(PROT,NORM)
* RESULT4                         RESULT4
RESULT4  DFHMDF POS=(12,67),LENGTH=11,ATTRB=(PROT,NORM),COLOR=GREEN
         DFHMDF POS=(12,79),LENGTH=1,ATTRB=(PROT,NORM)
* CMD5                            CMD5
CMD5     DFHMDF POS=(13,1),LENGTH=1,ATTRB=(UNPROT,NORM),COLOR=RED
         DFHMDF POS=(13,3),LENGTH=1,ATTRB=(PROT,NORM)
* NAME5                           NAME5
NAME5    DFHMDF POS=(13,5),LENGTH=8,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(13,14),LENGTH=1,ATTRB=(PROT,NORM)
* STATUS5                         STATUS5
STATUS5  DFHMDF POS=(13,16),LENGTH=49,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(13,66),LENGTH=Ø,ATTRB=(PROT,NORM)
* RESULT5                         RESULT5
RESULT5  DFHMDF POS=(13,67),LENGTH=11,ATTRB=(PROT,NORM),COLOR=GREEN
         DFHMDF POS=(13,79),LENGTH=1,ATTRB=(PROT,NORM)
* CMD6                            CMD6
CMD6     DFHMDF POS=(14,1),LENGTH=1,ATTRB=(UNPROT,NORM),COLOR=RED
         DFHMDF POS=(14,3),LENGTH=1,ATTRB=(PROT,NORM)
* NAME6                           NAME6
NAME6    DFHMDF POS=(14,5),LENGTH=8,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(14,14),LENGTH=1,ATTRB=(PROT,NORM)
* STATUS6                         STATUS6
STATUS6  DFHMDF POS=(14,16),LENGTH=49,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(14,66),LENGTH=Ø,ATTRB=(PROT,NORM)
* RESULT6                         RESULT6
RESULT6  DFHMDF POS=(14,67),LENGTH=11,ATTRB=(PROT,NORM),COLOR=GREEN
         DFHMDF POS=(14,79),LENGTH=1,ATTRB=(PROT,NORM)
* CMD7                            CMD7
CMD7     DFHMDF POS=(15,1),LENGTH=1,ATTRB=(UNPROT,NORM),COLOR=RED
```

```
         DFHMDF POS=(15,3),LENGTH=1,ATTRB=(PROT,NORM)
* NAME7                           NAME7
NAME7    DFHMDF POS=(15,5),LENGTH=8,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(15,14),LENGTH=1,ATTRB=(PROT,NORM)
* STATUS7                         STATUS7
STATUS7  DFHMDF POS=(15,16),LENGTH=49,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(15,66),LENGTH=Ø,ATTRB=(PROT,NORM)
* RESULT7                         RESULT7
RESULT7  DFHMDF POS=(15,67),LENGTH=11,ATTRB=(PROT,NORM),COLOR=GREEN
         DFHMDF POS=(15,79),LENGTH=1,ATTRB=(PROT,NORM)
* CMD8                            CMD8
CMD8     DFHMDF POS=(16,1),LENGTH=1,ATTRB=(UNPROT,NORM),COLOR=RED
         DFHMDF POS=(16,3),LENGTH=1,ATTRB=(PROT,NORM)
* NAME8                           NAME8
NAME8    DFHMDF POS=(16,5),LENGTH=8,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(16,14),LENGTH=1,ATTRB=(PROT,NORM)
* STATUS8                         STATUS8
STATUS8  DFHMDF POS=(16,16),LENGTH=49,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(16,66),LENGTH=Ø,ATTRB=(PROT,NORM)
* RESULT8                         RESULT8
RESULT8  DFHMDF POS=(16,67),LENGTH=11,ATTRB=(PROT,NORM),COLOR=GREEN
         DFHMDF POS=(16,79),LENGTH=1,ATTRB=(PROT,NORM)
* CMD9                            CMD9
CMD9     DFHMDF POS=(17,1),LENGTH=1,ATTRB=(UNPROT,NORM),COLOR=RED
         DFHMDF POS=(17,3),LENGTH=1,ATTRB=(PROT,NORM)
* NAME9                           NAME9
NAME9    DFHMDF POS=(17,5),LENGTH=8,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(17,14),LENGTH=1,ATTRB=(PROT,NORM)
* STATUS9                         STATUS9
STATUS9  DFHMDF POS=(17,16),LENGTH=49,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(17,66),LENGTH=Ø,ATTRB=(PROT,NORM)
* RESULT9                         RESULT9
RESULT9  DFHMDF POS=(17,67),LENGTH=11,ATTRB=(PROT,NORM),COLOR=GREEN
         DFHMDF POS=(17,79),LENGTH=1,ATTRB=(PROT,NORM)
* CMDA                            CMDA
CMDA     DFHMDF POS=(18,1),LENGTH=1,ATTRB=(UNPROT,NORM),COLOR=RED
         DFHMDF POS=(18,3),LENGTH=1,ATTRB=(PROT,NORM)
* NAMEA                           NAMEA
NAMEA    DFHMDF POS=(18,5),LENGTH=8,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(18,14),LENGTH=1,ATTRB=(PROT,NORM)
* STATUSA                         STATUSA
STATUSA  DFHMDF POS=(18,16),LENGTH=49,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(18,66),LENGTH=Ø,ATTRB=(PROT,NORM)
* RESULTA                         RESULTA
RESULTA  DFHMDF POS=(18,67),LENGTH=11,ATTRB=(PROT,NORM),COLOR=GREEN
         DFHMDF POS=(18,79),LENGTH=1,ATTRB=(PROT,NORM)
* CMDB                            CMDB
CMDB     DFHMDF POS=(19,1),LENGTH=1,ATTRB=(UNPROT,NORM),COLOR=RED
         DFHMDF POS=(19,3),LENGTH=1,ATTRB=(PROT,NORM)
* NAMEB                           NAMEB
```

```
NAMEB    DFHMDF POS=(19,5),LENGTH=8,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(19,14),LENGTH=1,ATTRB=(PROT,NORM)
* STATUSB                         STATUSB
STATUSB  DFHMDF POS=(19,16),LENGTH=49,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(19,66),LENGTH=0,ATTRB=(PROT,NORM)
* RESULTB                         RESULTB
RESULTB  DFHMDF POS=(19,67),LENGTH=11,ATTRB=(PROT,NORM),COLOR=GREEN
         DFHMDF POS=(19,79),LENGTH=1,ATTRB=(PROT,NORM)
* CMDC                            CMDC
CMDC     DFHMDF POS=(20,1),LENGTH=1,ATTRB=(UNPROT,NORM),COLOR=RED
         DFHMDF POS=(20,3),LENGTH=1,ATTRB=(PROT,NORM)
* NAMEC                           NAMEC
NAMEC    DFHMDF POS=(20,5),LENGTH=8,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(20,14),LENGTH=1,ATTRB=(PROT,NORM)
* STATUSC                         STATUSC
STATUSC  DFHMDF POS=(20,16),LENGTH=49,ATTRB=(ASKIP,NORM),COLOR=GREEN
         DFHMDF POS=(20,66),LENGTH=0,ATTRB=(PROT,NORM)
* RESULTC                         RESULTC
RESULTC  DFHMDF POS=(20,67),LENGTH=11,ATTRB=(PROT,NORM),COLOR=GREEN
         DFHMDF POS=(20,79),LENGTH=1,ATTRB=(PROT,NORM)
         DFHMDF POS=(21,1),LENGTH=1,ATTRB=(ASKIP,NORM)
* MESSAGE                         MESSAGE
MESSAGE  DFHMDF POS=(21,80),LENGTH=80,ATTRB=(PROT,BRT),COLOR=RED
         DFHMDF POS=(23,1),LENGTH=1,ATTRB=(PROT,NORM)
         DFHMDF POS=(23,80),LENGTH=4,INITIAL='Keys',ATTRB=(PROT,NORM), X
               COLOR=TURQUOISE
         DFHMDF POS=(24,5),LENGTH=1,INITIAL=':',ATTRB=(PROT,NORM),     X
               COLOR=NEUTRAL
         DFHMDF POS=(24,7),LENGTH=1,INITIAL='3',ATTRB=(PROT,BRT),      X
               COLOR=YELLOW
         DFHMDF POS=(24,9),LENGTH=3,INITIAL='End',ATTRB=(PROT,NORM),   X
               COLOR=TURQUOISE
         DFHMDF POS=(24,15),LENGTH=1,INITIAL='7',ATTRB=(PROT,BRT),     X
               COLOR=YELLOW
         DFHMDF POS=(24,17),LENGTH=4,INITIAL='Back',ATTRB=(PROT,NORM), X
               COLOR=TURQUOISE
         DFHMDF POS=(24,24),LENGTH=1,INITIAL='8',ATTRB=(PROT,BRT),     X
               COLOR=YELLOW
         DFHMDF
POS=(24,26),LENGTH=7,INITIAL='Forward',ATTRB=(PROT,NORM*00002140
               ),COLOR=TURQUOISE
         DFHMSD TYPE=FINAL
         END
```

Kevin Wailes
J Sainsbury (UK)

Relating response time to labour cost

On some days when our daily-average response-time numbers look quite good, users have been reporting that the CICS response time is quite slow. Because of this, I have been trying to analyse response time to evaluate whether response is ‘good’, ‘bad’, or ‘medium’. This is a challenging task for three reasons:

- Response time is highly variable.
- Both objective numerical issues and subjective psychological issues need to be considered.
- It’s hard to assign a good/bad value judgment to a plain number.

As I puzzled over these questions, a new way to analyse response time occurred to me.

This article addresses the issue of CICS response time degradation caused by competition for resources – from within the same CICS and from other tasks in the MVS system. It does not address the issue of degradation caused by factors within the transaction itself, such as inefficient program logic or inadequate file buffering.

EMPIRICAL OBSERVATIONS

Response time during any one minute, hour, or day can be vastly different from that during another similar time period, and the difference may be with or without apparent meaning.

I have analysed the range of response times during one day of CICSPDSS. When a graph relating time of day and response time was plotted, the result looked like noise. Although the highest points in each column form something of a trend across the page, the fact that each column (representing 15 minutes of the day) was filled with points going right down to the X-axis (ie zero response time) reflects the great variability in transaction response time, even within a 15-minute interval.

For the day in question, the mean response time was 0.469 seconds and the standard deviation was 0.891 seconds. With the standard deviation being larger than the mean, this shows how poorly the mean alone can represent the day’s events.

Because of this variability, any analytical presentation of the day’s response must involve some type of averaging. The simplest approach, as mentioned above, is to report the mean for the day. However, saying that ‘the response time for this day was 0.469 seconds’ fails to convey the customer frustration and labour cost during the peaks of up to 17.6 seconds per transaction.

Further analysis of the same day was performed, with each point on the graph representing the response time averaged over ‘n’ consecutive transactions, with ‘n’ increasing from 10, to 20, to 40. Comparing the graphs obtained revealed two trends:

- They conceal more noise, thus revealing more of the underlying pattern across the page.
- The peak response-time numbers get smaller (from 4.54 to 2.75 to 1.71 seconds), thus concealing more of the trouble represented by sharp peaks.

The first trend helps our understanding of the day, but the second tends to hinder it. An ideal measure of response time would combine the benefits of each.

Instead of averaging over a fixed *number* of consecutive transactions, it’s possible to average over fixed-length time *intervals* during a day. With further analysis of this effect, again using the same day as before, each point gives the average over a time interval, with the intervals increased from approximately 7.5, to 15, to 30 minutes.

This technique reduces much of the noise of the raw data, but a clear trend during the day is still elusive. Also, the day’s peak decreased from 1.27 seconds, to 0.773, to 0.663, obscuring the trouble at the peaks that reach 17.6 seconds.

THEORETICAL CONCERNS

Averaging over longer intervals yields smaller numbers, because the great majority of transactions are very fast. In experiential terms, averaging takes the 1,000 transactions during a day that ran for 30 seconds each and combines them with the 99,000 transactions that ran for 0.4 seconds each, giving an ‘average’ response time of 0.696 seconds. This average number totally hides the fact that, 1,000 times during the day, a person sat staring at a screen for 30 seconds, unable to do any productive work and growing frustrated at an accelerating rate.

Because of the effect of the rule of large numbers, quoting an average response time of ‘x’ seconds is meaningless without also quoting the length of time over which that average was taken.

Another major inadequacy of average response time is that it ignores the fact that some transactions do far more computing than others. If some transactions do 5 seconds of CPU work and 2,000 file accesses, while others do 0.1 seconds of CPU and 4 file accesses, it makes no sense to average their elapsed times together.

When deciding what is a ‘good’ response time, many people say that anything under half a second is great, or that people don’t notice a change in response time of less than a factor of two, either longer or shorter. Certainly few humans would notice, much less complain, about the difference between 0.2 and 0.5 seconds. On the other hand, if contention for resources were to cause 10,000 transactions in a day to run for 0.5 seconds instead of 0.2 seconds, that contention would cause 50 person-minutes of wasted labour time that day. An ideal measure of response time would recognize this labour time, even if the human users are never aware of this delay.

THE PROPOSED APPROACH

Instead of reporting a transaction’s elapsed time, I propose reporting the amount of labour time during that transaction caused by resource contention. This equals ‘the elapsed time of the transaction’ minus ‘what its elapsed time would have been in an unloaded system’. This recognizes the distinction between transactions that do little computation and those that do much.

I use the term ‘response latency’ for ‘the amount of time a transaction takes beyond what it would have taken in an unloaded system'. Because CPU usage and file I/O are the two kinds of work a transaction does, I estimate the time in an unloaded system by:

```
unloaded = (CPU time) + factor * (number of file accesses)
```

where 'factor' is the relative cost of one file access. Therefore I estimate the response latency by:

```
latency = (elapsed time) - ((CPU time) + factor * (file accesses))
```

In practical terms, I estimate the file 'factor' by examining response times on a weekend, when I assume the system is very lightly loaded. Adding up the elapsed times, CPU times, and file accesses of all transactions during a weekend, the file-factor can be estimated by:

$$\text{factor} = \frac{\text{(total elapsed time) - (total CPU time)}}{\text{total file accesses}}$$

In my measurements, I've found this file-factor to be about 0.001 for CICSPLAW (CICSPDSS has too few transactions on weekends to be statistically significant). Whether or not this finding can be generalized to other CICSs remains a topic for future research, as is the possible refinement of using a separate file-factor for each file.

In some installations, it might be difficult to obtain both CPU usage and the number of file accesses per transaction. At our installation, we use Omegamon II for CICS, which gives both measurements easily.

After computing the response latencies of each transaction, instead of averaging them over an interval of time, I propose summing them over the same interval of time. This gives the total amount of human labour that was wasted during that interval because of resource contention. Adding 5,000 numbers and dividing by 5,000 has the effect of shrinking the contributions of the peaks, whereas summing 5,000 numbers tends to give each number equal representation to the whole.

It is useful, however, to divide the sum of transactions' response latencies during an interval by the length of that interval. This yields the 'percent inflation' of labour time caused by resource contention, as compared to an unloaded system. For example, if the transactions running during a 5-minute interval sum to an aggregate response latency of 1 minute, the percent inflation would be (1/5)*100% = 20% during that interval. This means that the users' aggregate labour time during that interval was inflated by 20% when compared to that on an unloaded system.

With this method I have used interval lengths of approximately 7.5, 15, and 45 minutes. These give graphs with less noise, which show the traditional measure of response time averaged over the same length intervals. Incidentally, this method shows the lunch break very clearly – values near zero mean that users can work almost as fast as they could on an unloaded system.

CONCLUSION AND FUTURE RESEARCH

The percent inflation of labour time measures the system's response to its group of users during a chosen time interval. This measure has the advantages sought at the beginning of this paper:

- It filters out much random noise while retaining the impact of instantaneous spikes.
- It takes into account that some transactions do more computing than others.
- It represents the real cost of response delays regardless of whether human users notice those delays.

This method also yields a number that has an intuitive meaning that response time lacks – a percent inflation of zero means that users can work as fast as they could on an unloaded system. Inflation of 50% during an interval means that, effectively, 50% of a person (or 25% of two people, etc) was unproductive during that interval, because of contention for resources. In this way, it's easier to assign a good/bad value judgment to percent labour inflation than to simple response time.

One disadvantage of this approach can arise if one transaction gets stuck in the system, for example waiting for I/O from a broken terminal. Such a transaction might show an 'elapsed time' of several hours, but probably does not lead to much labour loss, because the user probably abandoned waiting for its completion within a few minutes.

The average response time approach would reflect this singularity accurately, by averaging its response time together with thousands of other transactions. However, the response latency approach of this article would be fooled into thinking that the user stared at the blank screen for the entire elapsed time of the anomalous transaction, and would therefore report several person-hours of labour cost. Perhaps a solution to this drawback can be found.

Future research could seek to correlate an interval's percent inflation to the end-users' subjective perception of productivity/frustration during that interval. Designing such a psychological experiment would be challenging, because people's expectations tend to influence their perceptions, and because of the difficulty people have assigning a numeric value to their subjective perceptions.

Another future topic would be to sum the response latencies (in person-minutes) over an entire day, to yield the number of person-hours of labour wasted because of resource contention during that day. This number summarizes the CICS's responsiveness over the day – with the advantages described above. It also has the potential to assign a monetary figure to the cost of resource contention, which could then be compared against the monetary cost of upgrading the computer hardware to reduce that contention. This would take much more research, however, because labour costs include emotional factors such as frustration, and frustration increases non-linearly with increasing response time.

Mark Krilanovich
Systems Programmer III
County of Santa Barbara (USA)

CICS task storage usage

The transaction TMAP creates a list of CICS tasks and their allocated storage elements. To do this it uses the program LCIMAPST to gather a list of active tasks, then individually interrogates their storage allocations.

Here's a brief outline of what LCIMAPST does:

- Inquire on all active tasks.
- Save task list to program's GETMAINed area.
- Inquire on each task to retrieve associated transaction, task key, and task location.
- Inquire on each transaction to retrieve a list of storage elements.
- Write output to transient data queue (CSML), which is associated with ddname MSGUSR.
- FREEMAIN storage.
- EXEC CICS return.

I've defined TMAP to run in CICS key and above the line.

The program has been tested using storage protection but I haven't tried it out under transaction isolation – although it should still work because it runs in CICS key.

A sample output from the program is shown in Figure 1.

LCIMAPST

```
LCIMAPST DFHEIENT CODEREG=(12),DATAREG=(13),EIBREG=(11)
*
GET_TASKS    EQU     *
     EXEC CICS INQUIRE TASK LIST LISTSIZE(NUMBER_ENTRIES)            X
             SET(2)
     ICM     5,15,NUMBER_ENTRIES
     BZ      EXIT_POINT
     LA      9,4
     MR      8,5
```

```
TRAN      TASK#     KEY  LOC  WHERE     STORADDR  STORLEN
AAON      0000030   CICS ANY  CICS24    0005E008  00000130
          CICS31         1386E448       00000350
          CICS31         138A7008       00001000
          CICS31         1386E008       00000430
          USER24         00140008       00000010
TOTAL     CICS31                        00001780
          CICS24                        00000130
          USER31                        00000000
          USER24                        00000010
DSNC      0000031   CICS ANY  CICS31    1388A7F8  00000160
          CICS31         388A588        00000260
          CICS31         138A9008       00001000
          CICS31         1388A008       00000430
          USER24         00141008       00000010
TOTAL     CICS31                        000017F0
          CICS24                        00000000
          USER31                        00000000
          USER24                        00000010
JNL2      0000034   CICS BELOW CICS24   0005B448  000002D0
          CICS24         0005B008       00000430
          CICS31         138A3008       00001000
TOTAL     CICS31                        00001000
          CICS24                        00000700
          USER31                        00000000
          USER24                        00000000
TMAP      0000577   CICS ANY  CICS31    138D2818  000000D0
          CICS31         138D27F8       00000010
          CICS31         138D2698       000000B0
          CICS31         138D2448       00000240
          CICS31         138DA008       00001000
          CICS31         138D2008       00000430
TOTAL     CICS31                        00001800
          CICS24                        00000000
          USER31                        00000000
          USER24                        00000000
```

Figure 1: Sample output

```
ST      9,GETMAIN_LENGTH
LR      3,9
EXEC CICS GETMAIN FLENGTH(GETMAIN_LENGTH) SET(8)
ST      8,GETMAIN_ADDRESS
MVCL    8,2
L       8,GETMAIN_ADDRESS
L       1,=A(L'RETURN_MESSAGE)
```

```
     STH     1,RETURN_LENGTH
*
     EXEC CICS ASSIGN USERID(CALL_USER)
     MVC     RETURN_MESSAGE,HEADER
     MVC     RETURN_MESSAGE+7Ø,CALL_USER
     EXEC    CICS WRITEQ TD FROM(RETURN_MESSAGE)                       X
                LENGTH(RETURN_LENGTH) QUEUE('CSML') NOHANDLE
*
INQUIRE_TASK_START      EQU     *
     MVC     TASK_ID,Ø(8)
     MVI     RETURN_MESSAGE,C' '
     MVC     RETURN_MESSAGE+1(L'RETURN_MESSAGE-1),RETURN_MESSAGE
*
     EXEC CICS INQUIRE TASK(TASK_ID)                                   X
             TRANSACTION(ELEMENT_TRANSID)                              X
             TASKDATAKEY(TASK_KEY) TASKDATALOC(TASK_LOC) NOHANDLE
*
     CLC     EIBRESP,DFHRESP(TASKIDERR)
     BE      EXIT_POINT
*
     MVC     WORKS,TASK_ID
     LA      15,CONVERT1
     BALR    14,15
     MVC     ELEMENT_ID,WORK_VAR
*
CHECK_KEY       EQU     *
     MVC     ELEMENT_KEY,=CL4'CICS'
     CLC     TASK_KEY,DFHVALUE(CICSDATAKEY)
     BE      CHECK_LOC
     MVC     ELEMENT_KEY,=CL4'USER'
*
CHECK_LOC EQU *
     MVC     ELEMENT_LOC,=CL5'BELOW'
     CLC     TASK_LOC,DFHVALUE(BELOW)
     BE      INQUIRE_TASK_STORAGE
     MVC     ELEMENT_LOC,=CL5'ANY'
*
INQUIRE_TASK_STORAGE EQU *
     EXEC CICS INQUIRE STORAGE TASK(TASK_ID)                           X
             ELEMENTLIST(4)                                            X
             NUMELEMENTS(NUMBER_STORAGE_ELEMENTS)                      X
             LENGTHLIST(6)
*
     CLC     EIBRESP,DFHRESP(TASKIDERR)
     BE      EXIT_POINT
*
     L       3,NUMBER_STORAGE_ELEMENTS
*
CHECK_WHERE_INIT        EQU     *
     XC      CICS31,CICS31
     XC      CICS24,CICS24
     XC      USER31,USER31
     XC      USER24,USER24
*
WRITE_RESULTS   EQU     *
     ICM     1,15,Ø(4)
     S       1,=F'8'
*
CHECK_C24    EQU     *
     CLI     Ø(1),C'M'
     BNE     CHECK_C31
     MVC     ELEMENT_NAME,=CL6'CICS24'
     L       1,Ø(,6)
     L       2,CICS24
     AR      2,1
     ST      2,CICS24
     B       CHECK_WHERE_END
*
CHECK_C31    EQU     *
     CLI     Ø(1),C'C'
     BNE     CHECK_U24
     MVC     ELEMENT_NAME,=CL6'CICS31'
     L       1,Ø(,6)
     L       2,CICS31
     AR      2,1
     ST      2,CICS31
     B       CHECK_WHERE_END
*
CHECK_U24    EQU     *
     CLI     Ø(1),C'B'
     BNE     CHECK_U31
     MVC     ELEMENT_NAME,=CL6'USER24'
     L       1,Ø(,6)
     L       2,USER24
     AR      2,1
     ST      2,USER24
     B       CHECK_WHERE_END
*
CHECK_U31    EQU     *
     CLI     Ø(1),C'U'
     BNE     CHECK_WHERE_END
     MVC     ELEMENT_NAME,=CL6'USER31'
     L       1,Ø(,6)
     L       2,USER31
     AR      2,1
     ST      2,USER31
     B       CHECK_WHERE_END
*
CHECK_WHERE_END EQU     *
*
```

```
     MVC     WORKS,Ø(4)
     LA      15,CONVERT1
     BALR    14,15
     MVC     ELEMENT_START,WORK_VAR
     MVC     WORKS,Ø(6)
     LA      15,CONVERT1
     BALR    14,15
     MVC     ELEMENT_SIZE,WORK_VAR
     EXEC CICS WRITEQ TD FROM(RETURN_MESSAGE)                          X
             LENGTH(RETURN_LENGTH) QUEUE('CSML') NOHANDLE
     LA      4,4(,4)
     LA      6,4(,6)
     MVC     ELEMENT_TRANSID,BLANKS
     MVC     ELEMENT_ID,BLANKS
     MVC     ELEMENT_KEY,BLANKS
     MVC     ELEMENT_LOC,BLANKS
     BCT     3,WRITE_RESULTS
*
     MVI     RETURN_MESSAGE,C' '
     MVC     RETURN_MESSAGE+1(L'RETURN_MESSAGE-1),RETURN_MESSAGE
     MVC     ELEMENT_LOC,=CL6'TOTALS'
     LA      4,4
*
WRITE_TOTALS EQU     *
*
TOTAL_C31    EQU     *
     C       4,=F'4'
     BL      TOTAL_C24
     MVC     ELEMENT_NAME,=CL6'CICS31'
     MVC     WORKS,CICS31
     LA      15,CONVERT1
     BALR    14,15
     MVC     ELEMENT_SIZE,WORK_VAR
     B       TOTAL_CHECK_END
*
TOTAL_C24    EQU     *
     C       4,=F'3'
     BL      TOTAL_U31
     MVC     ELEMENT_NAME,=CL6'CICS24'
     MVC     WORKS,CICS24
     LA      15,CONVERT1
     BALR    14,15
     MVC     ELEMENT_SIZE,WORK_VAR
     B       TOTAL_CHECK_END
*
TOTAL_U31    EQU     *
     C       4,=F'2'
     BL      TOTAL_U24
     MVC     ELEMENT_NAME,=CL6'USER31'
     MVC     WORKS,USER31
     LA      15,CONVERT1
     BALR    14,15
     MVC     ELEMENT_SIZE,WORK_VAR
     B       TOTAL_CHECK_END
*
TOTAL_U24    EQU     *
     MVC     ELEMENT_NAME,=CL6'USER24'
     MVC     WORKS,USER24
     LA      15,CONVERT1
     BALR    14,15
     MVC     ELEMENT_SIZE,WORK_VAR
*
TOTAL_CHECK_END EQU     *
     EXEC CICS WRITEQ TD FROM(RETURN_MESSAGE)                          X
             LENGTH(RETURN_LENGTH) QUEUE('CSML') NOHANDLE
     MVC     ELEMENT_LOC,BLANKS
     BCT     4,WRITE_TOTALS
*
     LA      8,4(,8)
     BCT     5,INQUIRE_TASK_START
*
     L       8,GETMAIN_ADDRESS
     EXEC CICS FREEMAIN DATAPOINTER(8) NOHANDLE
*
     MVI     RETURN_MESSAGE,C' '
     MVC     RETURN_MESSAGE+1(L'RETURN_MESSAGE-1),RETURN_MESSAGE
     MVC     RETURN_MESSAGE(L'MESSAGE),MESSAGE
     EXEC CICS SEND CONTROL ERASE
     EXEC CICS SEND FROM(RETURN_MESSAGE)
*
EXIT_POINT DS ØH
     EXEC CICS RETURN
*
*
*
CONVERT1     EQU     *
     UNPK    WORK_VAR(9),WORKS(5)
     MVZ     WORK_VAR,=XL8'ØØ'
     TR      WORK_VAR,TABLE
     XC      WORKS,WORKS
     BR      14
*
TABLE DC     C'Ø123456789ABCDEF'
BLANKS DC    CL2Ø' '
MESSAGE DC   C'TMAP COMPLETED - CHECK MSGUSR FILE'
HEADER DC    CL(L'RETURN_MESSAGE)' '
     ORG     HEADER+1
     DC      CL4'TRAN',C' '
     DC      CL7'TASK#',C' '
     DC      CL4'KEY',C' '
     DC      CL5'LOC',C' '
     DC      CL6'WHERE',C' '
     DC      CL8'STORADDR',C' '
     DC      CL8'STORLEN',C' '
     ORG     HEADER+6Ø
     DC      CL8'USERID -'
     ORG     ,
*
     LTORG   ,
*
     DFHEISTG
GETMAIN_LENGTH DS F
GETMAIN_ADDRESS DS F
RETURN_LENGTH  DS H
RETURN_MESSAGE DS CL8Ø
     ORG     RETURN_MESSAGE+1
ELEMENT_TRANSID DS CL4,C
ELEMENT_ID  DS CL7,C
ELEMENT_KEY DS CL4,C
ELEMENT_LOC DS CL5,C
ELEMENT_NAME DS CL6,C
ELEMENT_START DS CL8,C
ELEMENT_SIZE  DS CL8,C
     ORG     ,
RESULT   DS     F
WORKS    DS     CL4,C
WORK_VAR DS      CL8,C
NUMBER_ENTRIES DS F
ADDR_TASK_LIST_PTR DS F
TASK_ID  DS     F
ELEMENTLIST_PTR DS F
LENGTHLIST_PTR DS F
TASK_LOC DS     F
TASK_KEY DS     F
CALL_USER DS CL8
NUMBER_STORAGE_ELEMENTS DS F
CICS31   DS     F
CICS24   DS     F
USER31   DS     F
USER24   DS     F
*
     END
```

Calum Reid
Senior Systems Technician (UK)

January 1994 – November 1998 index

Items below are references to articles that have appeared in *CICS Update* since Issue 98, January 1994. References show the issue number followed by the page number(s). Back-issues of *CICS Update* are available back to issue 98 (January 1994). See page 2 for details.

CICS news

Trax Softworks has announced MailServer/390, a System/390 CICS-based SMTP Internet e-mail gateway and POP3 server that allows mainframes to use TCP/IP e-mail systems. Inbound and outbound messages reside in VSAM files delivered to mainframe users via 3270 interfaces.

The integrated system supports PC and 3270 clients and links with LAN-based systems and PC-based POP3 compliant clients. Remote users can send and receive mail by connecting to their home mailbox through any ISP. This provides a single unified method of sharing mail between diverse computing platforms, including company intranets and the Internet.

For further information contact:
Trax Softworks, 5840 Uplander Way, Culver City, CA 90230-6620, USA.
Tel: (310) 649 5800.
URL: http://www.traxsoft.com.

* * *

IBM has announced Release 3 of CICS Transaction Server for OS/390, incorporating CICS server, client, Transaction Gateway, and management function in the one package. The new Transaction Gateway (Version 3.0) supports OS/2, NT, AIX, and Solaris, and provides access to CICS servers from Web browsers and network computers. It also takes advantage of System/390 parallel sysplex.

New functions include Java application support, an object interface to CICS services for C++, CICS business transaction services, and long temporary storage queue names. For e-business, there's new CORBA client support, CICS Web interface enhancements, EXCI enhancements for resource recovery, a better 3270 bridge interface, and CICS Universal Clients.

Scalability features include dynamic routing and load balancing of Distributed Program Link (DPL) and EXEC CICS START requests, plus support for Coupling Facility data tables, Sysplex Wide Enqueue (ENQ) and Dequeue (DEQ), and named counter server.

Management improvements include CICSPlex System Manager enhancements, Resource Definition On-line (RDO) for CICS temporary storage, Auto-install for MVS consoles, and enhancements to CICS monitoring and statistics.

For further information contact your local IBM representative.

* * *

CICS users can now benefit from the joining of Insession and Destiny Software to deliver financial services applications. Insession's TransFuse legacy integration products and Destiny's various on-line products will be integrated to enable the applications to link to TP monitors and messaging systems such as CICS, IMS, and MQSeries.

For further information contact:
Insession, 100 Arapahoe Avenue, Boulder, CO 80302, USA.
Tel: (303) 440 3300
URL: http://www.insession.com.

* * *