November 2001

In this issue

3 CICS hot-pooling for Java applications – hints and tips
8 CICS load module information
11 Displaying Temporary Storage queues
25 Support for the COBOL SORT verb in CICS
46 December 1999 – November 2001 index
48 CICS news

© Xephon plc 2001
Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

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

Subscriptions and back-issues
A year’s subscription to CICS Update, comprising twelve monthly issues, costs £175.00 in the UK; $270.00 in the USA and Canada; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the December 1998 issue, are available separately to subscribers for £16.00 ($24.00) each including postage.

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

Editor
Trevor Eddolls

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

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

© Xephon plc 2001. 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.
CICS hot-pooling for Java applications – hints and tips

INTRODUCTION

CICS Transaction Server for OS/390 Release 1.3 (CICS TS 1.3) provides major performance improvements for Java application programs compiled using the Enterprise Toolkit for OS/390 Compiler and Binder (also known as the High Performance Java compiler or HPJ) via a technique known as ‘hot-pooling’. Support for hot-pooling within CICS TS 1.3 was shipped and enabled via PTF UQ44003.

This article provides guidance on various aspects of hot-pooling utilization.

STACK AND HEAP STORAGE CONSIDERATIONS

For non hot-pooled Java programs, the -lerunopts option on the HPJ command (used to compile and bind the program) can specify values such as the Language Environment (LE) stack size. For hot-pooled programs, CICS provides User Replaceable Module (URM) DFHAPH8O, where values such as stack and heap are specified. DFHAPH8O has default initial settings of 128KB for stack storage and 4MB for heap storage.

Since garbage collection is disabled for hot-pooled programs, heap storage usage will grow with each reuse of an LE enclave until CICS determines that the enclave should be terminated and a new one created. This process is handled automatically, and is one of the factors that determines the lifetime of a hot-pooled enclave environment within CICS. You should therefore review the heap storage requirement for your Java programs (see below). 4MB of heap storage may be insufficient for your object requirements, and you will therefore need to increase the heap value specified within DFHAPH8O to provide a larger object storage capacity for the hot-pooled programs.
DFHAPH8O is an optional URM. If it is not available for use, CICS provides default settings for the lerunopt parameters used for the hot-pooled enclaves. These defaults are hard-coded within CICS itself. It should be noted that CICS TS 1.3 PTF UQ46069 has modified the hard-coded default value for stack storage from 4KB to 128KB. This makes the hard-coded default stack size the same as that given as the example stack size specification within DFHAPH8O. A 4KB stack size can lead to GETMAINs for additional stack storage by LE when a hot-pooled program is being executed, if this initial stack amount is insufficient for the program’s needs. Such additional GETMAINs represent an increase in CPU utilization. For this reason, you should ensure PTF UQ46069 is applied to avoid additional GETMAIN calls/CPU overhead, if DFHAPH8O is not being used.

LE OPTIONS AND STORAGE REPORTING

You can obtain reports on LE options and storage usage for hot-pooled Java programs by respectively specifying RPTO(ON) and RPTS(ON) within DFHAPH8O. Note that the supplied version of DFHAPH8O has these settings commented-out within the source. This means that use of the default DFHAPH8O will not generate option and storage reports. If you wish to produce such reports whilst setting up and testing programs in a hot-pooled environment, uncomment the two lines in the DFHAPH8O source code.

Specifying RPTO(ON) and RPTS(ON) results in additional processing overhead, since the report information is written to transient data queue CESE by LE when the reports are generated at each enclave termination. Requesting both options can result in approximately 200 writes to the CESE queue when one pair of reports is generated.

The supplied definition for CESE is as an extrapartition transient data queue that relates to destination CEEMSG.

DETERMINING HEAP STORAGE USAGE

To determine the heap usage for an enclave, one approach is to set the initial heap size in DFHAPH8O to a small value (eg 4KB). Set
**RPTS(ON)** as well, by uncommenting it within DFHAPH8O. The first run of a hot-pooled program will use more than this amount of heap, and so force an enclave termination. Analysis of the report generated from **RPTS(ON)** shows how much heap storage was actually required by this run of the program. Some example figures are given below:

**HEAP statistics:**

Initial size: 4096

Total heap storage used (sugg initial size): 3174072

The storage report shows that in this example 3,174,072 bytes were required for the test run. The amount includes both the initial set-up of the function references for the DLLs used by the program, along with the object storage used by the execution of the program itself. Now set the initial heap size to its intended value (e.g., 10MB). Re-run the program until another report is generated (as the result of a subsequent enclave termination after a number of reuses of the environment). This time, the example report is as follows:

**HEAP statistics:**

Initial size: 10485760

Total heap storage used (sugg initial size): 10514808

In this example, it had been observed from analysis of the CICS trace that 57 runs of the hot-pooled program managed to use the enclave before it was terminated. So, 10514808−3174072 = 7340736 bytes were used by the last 56 runs, therefore each run used 131KB of heap storage on average for (for example) object storage.

**DSA LIMITS AND MAXOPENTCBS VALUES**

A CICS task running a hot-pooled Java program uses an LE enclave and an H8 Open TCB, managed by the Open Transaction Environment (OTE). Once a task has linked to a hot-pooled program, it maintains use of this TCB/enclave until the end of task. Each hot-pooled enclave within CICS is associated with a particular H8 TCB. The maximum throughput of hot-pooled transactions in a CICS region can therefore
be determined by dividing the number of available H8 TCBs by the average transaction response time. The number of concurrent H8 TCBs that may coexist within a single CICS region depends on the available storage for that system. The main restriction for this is below-the-line (< 16MB) storage, from both the CICS DSAs and the CICS Private Area storage within the address space.

Using the default LE options as provided within URM DFHAPLH80, an H8 TCB and its associated LE enclave require 6KB of below-the-line CICS Private Area storage and 20KB of below-the-line CICS DSA storage. This storage is in addition to any existing storage requirements that the CICS system may have.

The CICS-supplied statistics transaction STAT can be used when calculating current storage availability in both the CICS DSA and CICS Private Area storage. The following example was taken from such a STAT transaction for a CICS system that was to be used as a hot-pooled application-owning region:

Current DSA Limit...... 6,144K (set by DSALIM in the SIT)
Current DSA Used....... 5,196K
(Current DSA Limit) - (Current DSA Used) = (DSA available)
6,144K - 5,196K = 948K

Not all the available storage should be used when calculating how many H8 TCBs could exist in a CICS system. To provide sufficient relief from unexpectedly high demands on storage, a DSA buffer of approximately 500KB is recommended to be left. Therefore, assuming that 400KB of CICS DSA may be used by hot-pooled transactions, the example shows that (400K / 20K), or 20 H8 TCBs, could coexist with this CICS region’s available DSA.

In the same example, using STAT to return CICS Private Area storage usage shows:

Private Area storage available below 16MB....: 1,990K

1,990KB would support the 120KB requirement for 20 H8 TCBs. Again, it is recommended that a 500KB buffer of free CICS Private
Area storage be left when determining Private Area availability for hot-pooling use.

OTE system tuning parameter MAXOPENTCBS limits the total number of concurrent Open TCBs that may exist in a given CICS TS 1.3 system. Assuming JVMs are not being used (ie no J8 TCBs exist), MAXOPENTCBS therefore reflects the total number of H8 TCBs that can coexist within a single CICS region. Therefore, attention needs to be paid to ensure that MAXOPENTCBS is not set to too high a value, and that the optimum balance between CICS DSA and Private Area storage needs to be found.

Note that you should use statistics taken at times of peak system activity when making this determination, since you need to know the peak storage requirements for the current system set-up, when determining what additional storage availability may be exploited by hot-pooling transactions. Note also that other subsystems (eg DB2) will utilize CICS Private Area storage too.

The CICS Performance Guide gives further detailed information on Java hot-pooling, and on storage tuning within a CICS system in general.

SUMMARY
The DSA limit should be adjusted to its optimum value, and then MAXOPENTCBS be set to the lesser of (DSA available - 500KB)/20KB or (Private Area available - 500KB)/6KB. If this calculation yields too low a number of H8 TCBs to satisfy transaction throughput requirements, one possibility is to spread the applications across multiple CICS Java Application Owning Regions (JORs).

Readers wishing to discuss the material in this article further are welcome to contact me via e-mail, at andy_wright@uk.ibm.com.

_________

Andy Wright  
CICS Change Team  
IBM (UK)  

© IBM 2001
CICS load module information

Ken Rogers, one of our CICS Update readers, e-mailed the following query:

*I am trying to remember the CICS transaction utility that will tell me:

1 What CICS load libraries a given CICS load module is located in.
2 Which one is currently being executed.
3 What the concatenation of load libraries is for a given CICS region.

I think the easiest way to answer this question is to lead you through a set of CICS CPSM screen shots. CPSM (CICSplex System Manager) supplies data from all attached CICS address spaces running in a TSO address space on the same plex as the set of CICS address spaces.

So to specifically answer Ken’s question above, go into a TSO session, fire up the CSPM TSO End User Interface, and issue the PROGRAM command, which, after Enter is pressed, will provide the screen shown below:

```
12JUL2001 16:55:25 ----- INFORMATION DISPLAY -----------.
COMMAND ===> SCROLL ===> PAGE
CURR WIN ===> 1 ALT WIN ===> 
>W1 =PROGRAM========PJPLX====PJPLX======12JUL2001==16:54:56===PSM======76
CMD Program CICS Enabled Use Current Program Shared CEDF Copy
  Name— System— Status— Count— Use— Language— Status— Option— Require
  EYUTVOMD IYCWXCGF ENABLED 0 0 ASSEMBLER PRIVATE NOCEDF NOTREQU
  EYUTVOSE IYCWXCGF ENABLED 0 0 ASSEMBLER PRIVATE NOCEDF NOTREQU
  EYUTVOSK IYCWXCGF ENABLED 0 0 ASSEMBLER PRIVATE NOCEDF NOTREQU
  EYUTVOSS IYCWXCGF ENABLED 0 0 ASSEMBLER PRIVATE NOCEDF NOTREQU
  EYUTVTGE IYCWXCGF ENABLED 0 0 ASSEMBLER PRIVATE NOCEDF NOTREQU
  EYUTVGK IYCWXCGF ENABLED 0 0 ASSEMBLER PRIVATE NOCEDF NOTREQU
  EYUTVTSI IYCWXCGF ENABLED 0 0 ASSEMBLER PRIVATE NOCEDF NOTREQU
  EYUTVTHK IYCWXCGF ENABLED 0 0 ASSEMBLER PRIVATE NOCEDF NOTREQU
  EYUTVTHS IYCWXCGF ENABLED 0 0 ASSEMBLER PRIVATE NOCEDF NOTREQU
  EYUTVTHS IYCWXCGF ENABLED 0 0 ASSEMBLER PRIVATE NOCEDF NOTREQU
  EYUTVTHS IYCWXCGF ENABLED 0 0 ASSEMBLER PRIVATE NOCEDF NOTREQU
  EYUTVTHS IYCWXCGF ENABLED 0 0 ASSEMBLER PRIVATE NOCEDF NOTREQU
```
If we wanted information on program EYUTVOMD, tab down the display till the cursor is under its name and hit Enter. The following detail screen results:

12JUL2001 16:54:56 ——— INFORMATION DISPLAY ————————
COMMAND ===> SCROLL ===> PAGE
CURR WIN ===> 1 ALT WIN ===> 
<W1 =PROGRAM=PROGMOD=PJPLX==PJPLX==12JUL2001==16:54:56==CPM==1
CICS System... IYCZCGF Curr Use Cnt. 10
Exec Key...... CICSEXECKEY Tot Use Cnt.. 11
Execution Set. FULLAPI Use In Intvl. 11
Mirror Tranid. Newcopy Cnt.. 0
Shared Status. PRIVATE Removed Cnt.. 0
LPA/SVA Stat.. NOTLPA Fetch Cnt.... 1
Current Loc... RDAA RPL Number... 0
Held Status... NOHOLD Remote Name...
Fetch Time.... 00:00:00.01 Remote Sysid.
Avg Fetch Time 00:00:00.01 Copy Required NOTREQUA
Concurrency... QUASIRENT Runtime...... NONLE370
JVM Debug..... NODEBUG
JVM Profile... DFHVJMPR

The answer to question 2 lies in the RPL number – 8 in this case. Tab down to this field and hit Enter. The following screen results, detailing the RPL concatenation; so 8 maps to BLDBSF.PLUXA.SEYULO. So answering question 3:

12JUL2001 16:53:50 ——— INFORMATION DISPLAY ————————
COMMAND ===> SCROLL ===> PAGE
CURR WIN ===> 1 ALT WIN ===> 
W1 =RPLLISTD=====PJPLX====PJPLX==12JUL2001==16:53:50==CPM==20
RPL CICS Dataset
Num System- Name——————-
0 IYCZCGF CPSMDEV.PJOHNSO.LOAD
1 IYCZCGF CPSMDEV.TEST.LOAD
2 IYCZCGF CPSMDEV.BSF.LOAD
3 IYCZCGF CPSMDEV.DUMMY.LOAD
4 IYCZCGF UTL.PJOHNSO.LOAD
5 IYCZCGF CPSMDEV.TABLE620.LOAD
6 IYCZCGF PUBPLU.CPSM.LOAD
7 IYCZCGF PUBPLU.CPSM.TABLES
8 IYCZCGF BLDBSF.PLUXA.SEYULO
9 IYCZCGF PP.ADLE370.OS390210.SCECICS

To find which libraries a given program is located in, you could use the above list of libraries and search them via the library list utility under TSO (via cut and paste into a split screen). An alternative programming solution could use the EXEC CICSPlex SM application Programming Interface – see CICS Update, Utilizing the power of the CICSPlex SM Web User Interface, Issue 191, October 2001.

Andy Krasun
IBM (UK) © IBM 2001

Contributing to CICS Update

If you have ever experienced any difficulties with CICS, or made an interesting discovery, you could receive a cash payment, a free subscription to any of our Updates, or a credit against any of Xephon’s wide range of products and services, simply by telling us all about it.

If you’re interested in writing an article, but not sure what on, then visit the CICS Update Web site, http://www.xephon.com/cics, and follow the link to Opportunities for CICS specialists. Here you’ll find a list of topics that readers have asked us for more articles about.

Articles can be e-mailed to Trevor Eddolls at trevore@xephon.com. A copy of our Notes for Contributors is available from www.xephon.com/nfc.
Displaying Temporary Storage queues

CICS provides CEBR as the standard transaction for the visualization of temporary storage queues. However, you must know beforehand the name of the queue you want to browse. If you are not sure what queue you are looking for, then it can be difficult to find what you want, specially if the number of queues in the system is large.

To solve this problem, I created a program that shows all the TS queues in a CICS region, along with each one’s characteristics, as you can see below. To browse a queue, just put the cursor in the respective line and press Enter. The queue browsing is limited to the first 78 characters of each line, but that should be sufficient in most cases. If you need more complete browsing, then call CEBR for that queue. When you are in the visualization screen, you can jump directly to a specific line item by placing the desired item number in the Item Nr field that appears on top of it.

This application consists of a program (TSVIEW) and two BMS maps – one to show the queue names (TSVIEW1) and the other to display them (TSVIEW2). You can assign TSVIEW any transaction you want, because the program always returns to whatever transaction it was called by.

An example of the queue list screen is shown below:

<table>
<thead>
<tr>
<th>Queue</th>
<th>Items</th>
<th>Loc</th>
<th>Totlen</th>
<th>Max</th>
<th>Min</th>
<th>ACICS7B</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAIL3544</td>
<td>00002</td>
<td>AUX</td>
<td>000128</td>
<td>0064</td>
<td>0064</td>
<td></td>
</tr>
<tr>
<td>ABIL3550</td>
<td>00022</td>
<td>AUX</td>
<td>006550</td>
<td>0128</td>
<td>0064</td>
<td></td>
</tr>
<tr>
<td>ABTS1110</td>
<td>00501</td>
<td>AUX</td>
<td>005364</td>
<td>0064</td>
<td>0064</td>
<td></td>
</tr>
<tr>
<td>ABTS1111</td>
<td>00322</td>
<td>AUX</td>
<td>0013128</td>
<td>0128</td>
<td>0064</td>
<td></td>
</tr>
<tr>
<td>ABTS1115</td>
<td>00004</td>
<td>AUX</td>
<td>0056128</td>
<td>0064</td>
<td>0064</td>
<td></td>
</tr>
<tr>
<td>ABTS1116</td>
<td>00112</td>
<td>AUX</td>
<td>0055528</td>
<td>0128</td>
<td>0064</td>
<td></td>
</tr>
<tr>
<td>ABTS1117</td>
<td>05242</td>
<td>AUX</td>
<td>0023128</td>
<td>0128</td>
<td>0064</td>
<td></td>
</tr>
<tr>
<td>ABTS1118</td>
<td>00006</td>
<td>AUX</td>
<td>000028</td>
<td>0064</td>
<td>0064</td>
<td></td>
</tr>
<tr>
<td>ABTS1119</td>
<td>00402</td>
<td>AUX</td>
<td>0050128</td>
<td>0128</td>
<td>0064</td>
<td></td>
</tr>
<tr>
<td>ABTS1120</td>
<td>00566</td>
<td>AUX</td>
<td>004064</td>
<td>0064</td>
<td>0064</td>
<td></td>
</tr>
<tr>
<td>ABTS1121</td>
<td>00202</td>
<td>AUX</td>
<td>0055128</td>
<td>0064</td>
<td>0064</td>
<td></td>
</tr>
<tr>
<td>ABTS1122</td>
<td>01201</td>
<td>AUX</td>
<td>0023064</td>
<td>0064</td>
<td>0064</td>
<td></td>
</tr>
<tr>
<td>ABTS1123</td>
<td>00004</td>
<td>AUX</td>
<td>000464</td>
<td>0064</td>
<td>0064</td>
<td></td>
</tr>
</tbody>
</table>
TSVIEW SOURCE CODE

IDENTIFICATION DIVISION.
PROGRAM-ID. TSVIEW.

*=============================================================================*
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 LIXO PIC X VALUE SPACE.
77 X PIC 9(4) COMP VALUE 0.
77 Y PIC 9(4) COMP VALUE 0.
77 Z PIC 9(4) COMP VALUE 78.
77 Z1 PIC 9(4) COMP VALUE 78.
77 QPLIMIT PIC 9(4) COMP VALUE 100.
77 CURPOS PIC S9(2) VALUE +0.
77 COMMAREAL PIC 9(4) COMP VALUE 3900.
77 SCREEN1-LINES PIC 9(4) COMP VALUE 20.
77 SCREEN2-LINES PIC 9(4) COMP VALUE 20.
COPY DFHAID.
Ø1 WORK-FIELDS.
  Ø2 TXT.
    Ø3 FILLER PIC X(3).
    Ø3 TXT-ITEMS PIC 9(5).
    Ø3 FILLER PIC X(3).
    Ø3 TXT-LOCATION PIC X(4).
    Ø3 FILLER PIC X(3).
    Ø3 TXT-FIELD PIC 9(7).
    Ø3 FILLER PIC X(3).
    Ø3 TXT-MAXL PIC 9(4).
    Ø3 FILLER PIC X(3).
    Ø3 TXT-MINL PIC 9(4).
    Ø3 FILLER PIC X(7).

Ø2 TS-FIELDS.
    Ø3 TS-QUEUE PIC X(8).
    Ø3 TS-ITEMS PIC S9(4) COMP.
    Ø3 TS-FIELD PIC S9(8) COMP.
03 TS-LOCATION PIC S9(8) COMP.
03 TS-MAXITEMLEN PIC S9(4) COMP.
03 TS-MINITEMLEN PIC S9(4) COMP.
03 FILLER PIC X(54).

02 W-RESP PIC S9(8) COMP.
02 W-RESP2 PIC S9(8) COMP.
02 W-NUMERIC8 PIC 9(8).
02 FILLER REDEFINES W-NUMERIC8.
   03 FILLER PIC X.
   03 W-NUMERIC7 PIC 9(7).
02 FILLER REDEFINES W-NUMERIC8.
   03 FILLER PIC X(3).
   03 W-NUMERIC5 PIC 9(5).
02 FILLER REDEFINES W-NUMERIC8.
   03 FILLER PIC X(4).
   03 W-NUMERIC4 PIC 9(4).

01 COMMAREA.
   02 SELFFLAG PIC 9.
   02 NEXT-QUEUE PIC X(8).
   02 QUEUE-DISPLAY PIC X(8).
   02 Q-NRECX PIC X(5).
   02 Q-NREC9 REDEFINES Q-NRECX PIC 9(5).
   02 QUEUE-NRECS PIC S9(4) COMP.
   02 QUEUE-ITEM PIC S9(4) COMP.
   02 QUEUE-EXTRA PIC S9(4) COMP.
   02 QP PIC S9(4) COMP.
   02 QUEUE-FILLER PIC X(800).
   02 QUEUE-TABLE REDEFINES QUEUE-FILLER
      PIC X(8) OCCURS 100.

02 TSVIEW1I.
   03 FILLER PIC X(12).
   03 SISIDL PIC S9(4) COMP.
   03 SISIDA PIC X.
   03 SISIDI PIC X(8).
   03 SCREEN1-LINE OCCURS 20.
      04 CMDL PIC S9(4) COMP.
      04 CMDA PIC X.
      04 CMDI PIC X(3).
      04 QUEL PIC S9(4) COMP.
      04 QUEA PIC X.
      04 QUEI PIC X(8).
      04 TXTL PIC S9(4) COMP.
      04 TXTA PIC X.
      04 TXTI PIC X(46).
02 TSVIEWI0 REDEFINES TSVIEW1I PIC X(1343).

02 TSVIEW2I.
   03 FILLER PIC X(12).
`03 QNAME2L PIC S9(4) COMP.
03 QNAME2A PIC X.
03 QNAME2I PIC X(8).
03 SITEM2L PIC S9(4) COMP.
03 SITEM2A PIC X.
03 SITEM2I PIC 9(5).
03 SCREEM2-LINE OCCURS 20.
  04 QLINE PIC S9(4) COMP.
  04 QLINEA PIC X.
  04 QLINEI PIC X(70).
02 TSVIEW2O REDEFINES TSVIEW2I PIC X(1651).
02 FILLER PIC X(500).
*
** LINKAGE SECTION.**
*====================================================================*
01 DFHCOMMAREA.
  02 FILLER PIC X(4000).
*====================================================================*
** PROCEDURE DIVISION.**
*====================================================================*
* MOVE LOW-VALUES TO WORK-FIELDS COMMAREA.
*
** FIRST-TIME-ONLY.**
*====================================================================*
  IF EIBCALLEN = Ø
    EXEC CICS ASSIGN APPLID (SISIDI)
    END-EXEC
    MOVE 1 TO QP
    MOVE SPACES TO QUEUE-FILLER
    PERFORM NEXT-PAGE1
    PERFORM GET-FIRST-FORWARD
    PERFORM SEND-MAP1
    MOVE 1 TO SELFLAG
    GO TO RETURN-TRANSID
  END-IF.
*
** OTHER-TIMES.**
*====================================================================*
  MOVE DFHCOMMAREA TO COMMAREA
  IF SELFLAG = 1
    PERFORM RECEIVE-MAP1
    IF EIBAID = DFHENTER
      PERFORM CHECK-COMMAND
    END-IF
    IF EIBAID = DFHPF7 OR EIBAID = DFHPF19
      IF QP > 1
        SUBTRACT 1 FROM QP
      END-IF
    MOVE QUEUE-TABLE(QP) TO NEXT-QUEUE
PERFORM NEXT-PAGE1
END-IF
IF EIBAID = DFHPF8 OR EIBAID = DFHPF20
   IF QP < QPLIMIT
      ADD 1 TO QP
   END-IF
   MOVE QUEUE-TABLE(QP) TO NEXT-QUEUE
   PERFORM NEXT-PAGE1
   PERFORM GET-FIRST-FORWARD
END-IF
PERFORM SEND-MAP1
GO TO RETURN-TRANSID
END-IF.

IF SELFLAG = 2
   PERFORM RECEIVE-MAP2
   IF EIBAID = DFHPF7 OR EIBAID = DFHPF19
      PERFORM PREV-PAGE2
   ELSE
      PERFORM NEXT-PAGE2
   END-IF
   PERFORM SEND-MAP2
   GO TO RETURN-TRANSID
END-IF.

*========================================================================*
* Subroutines
*========================================================================*
* NEXT-PAGE1.
*========================================================================*
PERFORM CLEAN-SCREEN1 VARYING X FROM 1 BY 1
   UNTIL X > SCREEN1-LINES
EXEC CICS INQUIRE TSQUEUE START
END-EXEC
MOVE 0 TO X
PERFORM LOAD-SCREEN1 THRU LOAD-SCREEN1-EXIT
   UNTIL X > SCREEN1-LINES.

LOAD-SCREEN1.
*========================================================================*
EXEC CICS INQUIRE TSQUEUE (TS-QUEUE)
   NUMITEMS (TS-NUMITEMS)
   FLENGTH (TS-FLENGTH)
   LOCATION (TS-LOCATION)
   MAXITEMLEN (TS-MAXITEMLEN)
   MINITEMLEN (TS-MINITEMLEN)
   RESP (W-RESP)
   RESP2 (W-RESP2)
   NEXT
END-EXEC
IF W-RESP2 > \0
  MOVE 21 TO X
  GO TO LOAD-SCREEN1-EXIT
END-IF.
IF TS-QUEUE < NEXT-QUEUE
  GO TO LOAD-SCREEN1-EXIT
END-IF.
ADD 1 TO X
IF X > SCREEN1-LINES
  GO TO LOAD-SCREEN1-EXIT
END-IF.
IF TS-LOCATION = DFHVALUE(MAIN)
  MOVE 'MAIN' TO TXT-LOCATION
END-IF.
IF TS-LOCATION = DFHVALUE(AUXILIARY)
  MOVE 'AUX ' TO TXT-LOCATION
END-IF
MOVE TS-NUMITEMS TO W-NUMERICA
MOVE W-NUMERIC5 TO TXT-NUMITEMS
MOVE TS-FLENGTH TO W-NUMERICA
MOVE W-NUMERIC7 TO TXT-FLENGTH
MOVE TS-MAXITEMLEN TO W-NUMERICA
MOVE W-NUMERIC4 TO TXT-MAXL
MOVE TS-MINITEMLEN TO W-NUMERICA
MOVE W-NUMERIC4 TO TXT-MINL
MOVE TXT TO TXTI(X)
MOVE TS-QUEUE TO QUEI(X).

* LOAD-SCREEN1-EXIT.
*==================================
  EXIT.
*
CLEAN-SCREEN1.
*============
  MOVE SPACES TO TXT TXTI(X) QUEI(X) CMDI(X).
*
GET-FIRST-FORWARD.
*=================
  EXEC CICS INQUIRE TSQUEUE (TS-QUEUE)
END-EXEC
  IF QP < QPLIMIT
    ADD 1 TO QP
  END-IF
  MOVE TS-QUEUE TO QUEUE-TABLE(QP)
  SUBTRACT 1 FROM QP.
*
CHECK-COMMAND.
*===============
  COMPUTE CURPOS = EIBCPOSN / 80 - 1.
IF CURPOS GREATER 0 AND
  CURPOS NOT GREATER SCREEN1-LINES
  MOVE QUEI(CURPOS) TO QUEUE-DISPLAY QNAME2I
  MOVE '*' TO CMDI(CURPOS)
  MOVE TXTI(CURPOS)(4:5) TO Q-NRECX
  MOVE Q-NRECS TO QUEUE-NRECS
  MOVE 2 TO SELFLAG
  MOVE 1 TO QUEUE-ITEM
  MOVE 0 TO QUEUE-EXTRA
  PERFORM NEXT-PAGE2
  PERFORM SEND-MAP2
  GO TO RETURN-TRANSID
END-IF.
*
NEXT-PAGE2.
*============*
IF SITEM2I > 0 AND SITEM2I NUMERIC
  IF SITEM2I > 0 AND SITEM2I < QUEUE-NRECS
    MOVE SITEM2I TO QUEUE-ITEM
  END-IF
  IF SITEM2I > QUEUE-NRECS
    SUBTRACT 19 FROM QUEUE-NRECS GIVING QUEUE-ITEM
  END-IF
  IF SITEM2I < 1
    MOVE 1 TO QUEUE-ITEM
  END-IF
END-IF
IF QUEUE-ITEM LESS 1
  MOVE 1 TO QUEUE-ITEM
END-IF
EXEC CICS IGNORE CONDITION LENGERR
END-EXEC
PERFORM CLEAN-SCREEN2 VARYING Y FROM 1 BY 1
  UNTIL Y > SCREEN2-LINES
PERFORM LOAD-SCREEN2 THRU LOAD-SCREEN2-EXIT
  VARYING Y FROM 1 BY 1 UNTIL Y > SCREEN2-LINES
IF QUEUE-ITEM > QUEUE-NRECS
  MOVE SITEM2I TO QUEUE-ITEM
  MOVE 20 TO QUEUE-EXTRA
END-IF.
*
PREV-PAGE2.
*============*
SUBTRACT 40 FROM QUEUE-ITEM
ADD QUEUE-EXTRA TO QUEUE-ITEM
MOVE 0 TO QUEUE-EXTRA
IF QUEUE-ITEM LESS 1
  MOVE 1 TO QUEUE-ITEM
END-IF
EXEC CICS IGNORE CONDITION LENGERR
END-EXEC
PERFORM CLEAN-SCREEN2 VARYING Y FROM 1 BY 1
UNTIL Y > SCREEN2-LINES
PERFORM LOAD-SCREEN2 THRU LOAD-SCREEN2-EXIT
   VARYING Y FROM 1 BY 1 UNTIL Y > SCREEN2-LINES.
*
LOAD-SCREEN2.
*==========*
   IF QUEUE-ITEM > QUEUE-NRECS
      GO TO LOAD-SCREEN2-EXIT
   END-IF
   IF Y = 1
      MOVE QUEUE-ITEM TO SITEM2I
   END-IF
   MOVE Z TO Z1
   EXEC CICS READQ TS QUEUE (QUEUE-DISPLAY)
      ITEM (QUEUE-ITEM)
      INTO (QLINEI(Y))
      LENGTH(Z1)
   END-EXEC
   ADD 1 TO QUEUE-ITEM.
*
LOAD-SCREEN2-EXIT.
*------------------*
   EXIT.
*
CLEAN-SCREEN2.
*==========*
   MOVE SPACES TO QLINEI(Y).
*
RECEIVE-MAP1.
*================*
   EXEC CICS RECEIVE MAP('TSVIEW1')
   END-EXEC.
   IF EIBAID = DFHPF3 OR EIBAID = DFHPF15
      GO TO EXIT-PROGRAM
   END-IF.
*
SEND-MAP1.
*========*
   EXEC CICS SEND MAP('TSVIEW1') ERASE
   END-EXEC.
*
RECEIVE-MAP2.
*================*
   EXEC CICS RECEIVE MAP('TSVIEW2')
   END-EXEC.
   IF EIBAID = DFHPF3 OR EIBAID = DFHPF15
      MOVE 1 TO SELFLAG
      PERFORM SEND-MAP1
GO TO RETURN-TRANSID
END-IF.
*
SEND-MAP2.
*=================* EXEC CICS SEND MAP('TSVIEW2') ERASE END-EXEC.
*
RETURN-TRANSID.
*=================* EXEC CICS RETURN TRANSID (EIBTRNID)
COMMArea (COMMArea)
LENGTH (COMMArea)
END-EXEC.
*
EXIT-PROGRAM.
*=================* EXEC CICS SEND FROM(LIX0) LENGTH(1) ERASE END-EXEC EXEC CICS RETURN END-EXEC.
GOBACK.

TSVIEW1 SOURCE CODE

MAPSET DFHMSD TYPE=&SYS Parm, MODE=INOUT, CTRL=(FREEKB),
* LANG=COBOL, TIOAPFX=YES, EXTATT=MAPONLY
*
T SVIEW1 DFHMDI SIZE=(24,80)
*
   DFHMDF POS=(01,06), LENGTH=46, ATTRB=(ASKIP, PROT),
   COLOR=TURQUOISE,
   INITIAL='Queue Items Loc Totlen Max Min'
SISID DFHMDF POS=(01,08), LENGTH=08, ATTRB=(PROT, FSET),
   COLOR=DEFAULT
*
   DFHMDF POS=(02,01), LENGTH=76, ATTRB=(ASKIP, PROT),
   COLOR=RED,
   INITIAL='__________________________'
   ____________________'
*
   DFHMDF POS=(03,01), LENGTH=03, ATTRB=(UNPROT, FSET),
   COLOR=RED
CMD01 DFHMDF POS=(03,01), LENGTH=03, ATTRB=(UNPROT, FSET),
   COLOR=RED
*
CMD02 DFHMDF POS=(04,01), LENGTH=03, ATTRB=(UNPROT, FSET),
*
COLOR=RED
QUE02 DFHMDF POS=(04,05),LENGTH=08,ATTRB=(ASKIP, PROT, FSET),
COLOR=RED

COLOR=RED
TEX03 DFHMDF POS=(06,01),LENGTH=03,ATTRB=(UNPROT, FSET),
COLOR=RED

COLOR=RED
QUE04 DFHMDF POS=(06,05),LENGTH=08,ATTRB=(ASKIP, PROT, FSET),
COLOR=RED

COLOR=RED
TEX04 DFHMDF POS=(06,14),LENGTH=46,ATTRB=(ASKIP, PROT, FSET),
COLOR=RED

COLOR=RED
QUE05 DFHMDF POS=(07,01),LENGTH=03,ATTRB=(UNPROT, FSET),
COLOR=RED

COLOR=RED
TEX05 DFHMDF POS=(07,05),LENGTH=08,ATTRB=(ASKIP, PROT, FSET),
COLOR=RED

COLOR=RED
QUE06 DFHMDF POS=(07,14),LENGTH=46,ATTRB=(ASKIP, PROT, FSET),
COLOR=RED

COLOR=RED
TEX06 DFHMDF POS=(08,01),LENGTH=03,ATTRB=(UNPROT, FSET),
COLOR=RED

COLOR=RED
QUE07 DFHMDF POS=(08,05),LENGTH=08,ATTRB=(ASKIP, PROT, FSET),
COLOR=RED

COLOR=RED
TEX07 DFHMDF POS=(08,14),LENGTH=46,ATTRB=(ASKIP, PROT, FSET),
COLOR=RED

COLOR=RED
QUE08 DFHMDF POS=(09,01),LENGTH=03,ATTRB=(UNPROT, FSET),
COLOR=RED

COLOR=RED
TEX08 DFHMDF POS=(09,05),LENGTH=08,ATTRB=(ASKIP, PROT, FSET),
COLOR=RED

COLOR=RED
QUE09 DFHMDF POS=(09,14),LENGTH=46,ATTRB=(ASKIP, PROT, FSET),
COLOR=RED

COLOR=RED
TEX09 DFHMDF POS=(10,01),LENGTH=03,ATTRB=(UNPROT, FSET),
COLOR=RED

COLOR=RED
TEX0A DFHMDF POS=(10,05),LENGTH=08,ATTRB=(ASKIP, PROT, FSET),
COLOR=RED

COLOR=RED
TEX0B DFHMDF POS=(10,14),LENGTH=46,ATTRB=(ASKIP, PROT, FSET),
COLOR=RED

COLOR=RED
TEX0C DFHMDF POS=(11,01),LENGTH=03,ATTRB=(UNPROT, FSET),
COLOR=RED
DFHMDH POS=(11,05), LENGTH=8, ATTRB=(ASKIP, PROT, FSET), COLOR=RED
DFHMDH POS=(11,14), LENGTH=46, ATTRB=(ASKIP, PROT, FSET), COLOR=TURQUOISE
* CMD10 DFHMDH POS=(12,01), LENGTH=3, ATTRB=(UNPROT, FSET), COLOR=RED
* QUE10 DFHMDH POS=(12,05), LENGTH=8, ATTRB=(ASKIP, PROT, FSET), COLOR=RED
* TXT10 DFHMDH POS=(12,14), LENGTH=46, ATTRB=(ASKIP, PROT, FSET), COLOR=TURQUOISE
* CMD11 DFHMDH POS=(13,01), LENGTH=3, ATTRB=(UNPROT, FSET), COLOR=RED
* QUE11 DFHMDH POS=(13,05), LENGTH=8, ATTRB=(ASKIP, PROT, FSET), COLOR=RED
* TXT11 DFHMDH POS=(13,14), LENGTH=46, ATTRB=(ASKIP, PROT, FSET), COLOR=TURQUOISE
* CMD12 DFHMDH POS=(14,01), LENGTH=3, ATTRB=(UNPROT, FSET), COLOR=RED
* QUE12 DFHMDH POS=(14,05), LENGTH=8, ATTRB=(ASKIP, PROT, FSET), COLOR=RED
* TXT12 DFHMDH POS=(14,14), LENGTH=46, ATTRB=(ASKIP, PROT, FSET), COLOR=TURQUOISE
* CMD13 DFHMDH POS=(15,01), LENGTH=3, ATTRB=(UNPROT, FSET), COLOR=RED
* QUE13 DFHMDH POS=(15,05), LENGTH=8, ATTRB=(ASKIP, PROT, FSET), COLOR=RED
* TXT13 DFHMDH POS=(15,14), LENGTH=46, ATTRB=(ASKIP, PROT, FSET), COLOR=TURQUOISE
* CMD14 DFHMDH POS=(16,01), LENGTH=3, ATTRB=(UNPROT, FSET), COLOR=RED
* QUE14 DFHMDH POS=(16,05), LENGTH=8, ATTRB=(ASKIP, PROT, FSET), COLOR=RED
* TXT14 DFHMDH POS=(16,14), LENGTH=46, ATTRB=(ASKIP, PROT, FSET), COLOR=TURQUOISE
* CMD15 DFHMDH POS=(17,01), LENGTH=3, ATTRB=(UNPROT, FSET), COLOR=RED
* QUE15 DFHMDH POS=(17,05), LENGTH=8, ATTRB=(ASKIP, PROT, FSET), COLOR=RED
* TXT15 DFHMDH POS=(17,14), LENGTH=46, ATTRB=(ASKIP, PROT, FSET), COLOR=TURQUOISE
* CMD16 DFHMDH POS=(18,01), LENGTH=3, ATTRB=(UNPROT, FSET), COLOR=RED
QUE16  DFHMDF POS=(18,05), LENGTH=08, ATTRB=(ASKIP, PROT, FSET), * 
COLOR= YELLOW
TXT16  DFHMDF POS=(18, 14), LENGTH=46, ATTRB=(ASKIP, PROT, FSET), * 
COLOR=TURQUOISE
CMD17  DFHMDF POS=(19, 01), LENGTH=03, ATTRB=(UNPROT, FSET), * 
COLOR=RED
QUE17  DFHMDF POS=(19, 05), LENGTH=08, ATTRB=(ASKIP, PROT, FSET), * 
COLOR= YELLOW
TXT17  DFHMDF POS=(19, 14), LENGTH=46, ATTRB=(ASKIP, PROT, FSET), * 
COLOR=TURQUOISE
CMD18  DFHMDF POS=(20, 01), LENGTH=03, ATTRB=(UNPROT, FSET), * 
COLOR=RED
QUE18  DFHMDF POS=(20, 05), LENGTH=08, ATTRB=(ASKIP, PROT, FSET), * 
COLOR= YELLOW
TXT18  DFHMDF POS=(20, 14), LENGTH=46, ATTRB=(ASKIP, PROT, FSET), * 
COLOR=TURQUOISE
DFHMDF POS=(20, 63), LENGTH=13, ATTRB=(ASKIP, PROT), * 
COLOR=NEUTRAL, * 
INITIAL='PF7 Prev page'
CMD19  DFHMDF POS=(21, 01), LENGTH=03, ATTRB=(UNPROT, FSET), * 
COLOR=RED
QUE19  DFHMDF POS=(21, 05), LENGTH=08, ATTRB=(ASKIP, PROT, FSET), * 
COLOR= YELLOW
TXT19  DFHMDF POS=(21, 14), LENGTH=46, ATTRB=(ASKIP, PROT, FSET), * 
COLOR=TURQUOISE
DFHMDF POS=(21, 63), LENGTH=13, ATTRB=(ASKIP, PROT), * 
COLOR=NEUTRAL, * 
INITIAL='PF8 Next page'
CMD20  DFHMDF POS=(22, 01), LENGTH=03, ATTRB=(UNPROT, FSET), * 
COLOR=RED
QUE20  DFHMDF POS=(22, 05), LENGTH=08, ATTRB=(ASKIP, PROT, FSET), * 
COLOR= YELLOW
TXT20  DFHMDF POS=(22, 14), LENGTH=46, ATTRB=(ASKIP, PROT, FSET), * 
COLOR=TURQUOISE
DFHMDF POS=(22, 63), LENGTH=08, ATTRB=(ASKIP, PROT), * 
COLOR=NEUTRAL, * 
INITIAL='PF3 Exit'

DFHMDF POS=(23, 01), LENGTH=75, ATTRB=(ASKIP, PROT), * 
COLOR=RED, * 
INITIAL='______________________________'

DFHMDF POS=(24, 08), LENGTH=64, ATTRB=(ASKIP, PROT), * 
COLOR=NEUTRAL, * 
INITIAL='Place the cursor in front of a queue and press * ENTER to display it'
*  
  DFHMSD TYPE=FINAL  
  END  

TSVIEW2 SOURCE CODE

MAPSET  DFHMSD TYPE=&SYSParm,MODE=INOUT,CTRL=(FREEKB),  *
  LANG=COBOL,TIOAPFX=YES,EXTATT=MAPONLY  
  *  
  TSVIEW2  DFHMDI SIZE=(24,80)  
  *  
  DFHMD DFHMDI POS=(01,07),LENGTH=06,ATTRB=(ASKIP,PROT),  *
  COLOR=TURQUOISE,  *  
  INITIAL='Queue:'  
  QName2  DFHMD DFHMDI POS=(01,14),LENGTH=08,ATTRB=(ASKIP,PROT,FSET),  *
  COLOR=NEUTRAL  
  DFHMD DFHMDI POS=(01,30),LENGTH=12,ATTRB=(ASKIP,PROT),  *
  COLOR=YELLOW,  *  
  INITIAL='Item Number:'  
  SITEM2  DFHMD DFHMDI POS=(01,43),LENGTH=05,ATTRB=(UNPROT,NUM,IC),  *
  COLOR=NEUTRAL  
  DFHMD DFHMDI POS=(01,49),LENGTH=01  
  *  
  DFHMD DFHMDI POS=(02,01),LENGTH=76,ATTRB=(ASKIP,PROT),  *
  COLOR=RED,  *  
  INITIAL='__________________________  
                  '  
  *  
  Qlin01  DFHMD DFHMDI POS=(03,01),LENGTH=78,ATTRB=(ASKIP,PROT,FSET),  *
  COLOR=DEFAULT  
  QLIN02  DFHMD DFHMDI POS=(04,01),LENGTH=78,ATTRB=(ASKIP,PROT,FSET),  *
  COLOR=DEFAULT  
  QLIN03  DFHMD DFHMDI POS=(05,01),LENGTH=78,ATTRB=(ASKIP,PROT,FSET),  *
  COLOR=DEFAULT  
  QLIN04  DFHMD DFHMDI POS=(06,01),LENGTH=78,ATTRB=(ASKIP,PROT,FSET),  *
  COLOR=DEFAULT  
  QLIN05  DFHMD DFHMDI POS=(07,01),LENGTH=78,ATTRB=(ASKIP,PROT,FSET),  *
  COLOR=DEFAULT  
  QLIN06  DFHMD DFHMDI POS=(08,01),LENGTH=78,ATTRB=(ASKIP,PROT,FSET),  *
  COLOR=DEFAULT  
  QLIN07  DFHMD DFHMDI POS=(09,01),LENGTH=78,ATTRB=(ASKIP,PROT,FSET),  *
  COLOR=DEFAULT  
  QLIN08  DFHMD DFHMDI POS=(10,01),LENGTH=78,ATTRB=(ASKIP,PROT,FSET),  *
  COLOR=DEFAULT  
  QLIN09  DFHMD DFHMDI POS=(11,01),LENGTH=78,ATTRB=(ASKIP,PROT,FSET),  *
  COLOR=DEFAULT  
  QLIN10  DFHMD DFHMDI POS=(12,01),LENGTH=78,ATTRB=(ASKIP,PROT,FSET),  *
  COLOR=DEFAULT  
  QLIN11  DFHMD DFHMDI POS=(13,01),LENGTH=78,ATTRB=(ASKIP,PROT,FSET),  *

Circle CICS TS 1.3 control blocks wall chart

Circle, famous for its control blocks wall chart, has just completed the CICS TS 1.3 version. This is the most comprehensive yet, covering all the new Web and Sockets blocks and their connectivity, plus much more. To see a thumb-nail of the chart, go to www.circle-group.com To order your copy please contact: USA: Ezriel@circle-us.com Tel: (973) 890-9331; rest of the world: elaine@circle-group.com Tel: +44 (0)1273 721123.
Support for the COBOL SORT verb in CICS

This article describes the process which resulted in the creation of a CICS-compatible SORT routine, callable from general purpose COBOL programs via the SORT verb.

BACKGROUND

Some time ago I wrote a program that lists the terminals acquired by a CICS region, including information about running (or next) transaction code, user-id, and user names. The program returned the information in the order returned by CICS from the INQUIRE TERMINAL NEXT commands.

It became obvious that the program could be more useful, especially for fairly large sets of data (up to several hundred or more terminals), if the data could be sorted into other sequences, such as user name, terminal-id, network-id, or user-id.

I investigated alternatives to determine what was the easiest way to sort small amounts of data in CICS, and found from the COBOL manuals that the SORT verb is supported under CICS when using COBOL for OS/390 and VM.

Having found this, and knowing that I really didn’t want to build a sort program from scratch, I proceeded to write the COBOL code to implement an in-store sort using input and output procedures, as required for use within CICS.

The first test failed dismally, which is when I discovered that COBOL doesn’t actually implement a sort. It just builds an interface to the DFSORT (or compatible) module ‘SORT’. My program received an abend because the SORT program could not be loaded.

I took a typically male approach at this point, trying to fix the problem without reading the manuals (a bad mistake). I ended up adding the system library containing DFSORT to my DFHRPL concatenation, and then to STEPLIB as well. At first look, this approach seemed to work. The abend went away, the data came back to the program
sorted, and I had the output I wanted. I was a bit concerned though that there was a long delay whenever I called the sort. Further investigation found even more problems. During the several seconds which even a small sort (<10 records) was taking, all work within the CICS region was suspended. It was clear that I had a problem.

Finally doing what I should have earlier and returning to the manuals, I found that DFSORT does not have a CICS-compatible routine, and I was doing an MVS sort in my CICS region, hogging the QR TCB for the life of the sort. This was when I discovered that IBM does not provide a CICS-compatible sort module.

I found that third parties do have sorts for CICS available (I believe that CA-CICSort and possibly Syncsort provide a routine) but my site was not a user, and I could not justify the expense.

I was left with two alternatives. I could implement a callable sort, and throw out the COBOL SORT verb, or I could implement a DFSORT and CICS-compatible sort routine myself.

I suspect that the former option would have been more sensible.

I built the sort routine.

**DESIGN**

Once I had decided to write the SORT module, I needed to find out what its interface should look like so that I could interface successfully with COBOL. My first attempt at this failed dismally. I asked IBM to provide the details I would need, pointing out to them that as their COBOL manual said that it was compatible with CICS, but they didn’t provide a routine, they should at least give me enough information so that I could write the routine. After a wait of some weeks, I received a note from the change team saying that the information was proprietary and was not released to customers.

Back to the manuals again, after realizing that the DFSORT manuals document how to write the E15 and E35 exits routines so that DFSORT can call them. This gave me what I needed. All I needed to do was build a program that would correctly interface with routines
built to the E15 and E35 specifications in the manual.

After several test routines, which I used to determine which interface (24 or 31-bit) COBOL was using, and exactly how the sort parameter text was formatted, I got down to writing the interface code. I ended up with a program that could be called by COBOL, would call the E15 exit to obtain all the records, and would then call the E35 exit to return the records back to the program. Now I just had to solve my original problem – how was I going to sort the data?

SORT ALGORITHM

Realizing that sorting algorithms are well documented, I went to my university texts, and found samples of various sorts. I looked at implementing a shell sort and a heap sort, for which I had Pascal code available, but then realized that someone else may already have answered my need. I didn’t find anything on the CBT tape, but I did find a CICS sort routine in the November 1990 edition of *CICS Update*, Issue 60. That program implemented a tag-key quick sort, which suited my purposes, so I downloaded the source and adapted it for my program.

The original TSQ sort program on which the sort component here is based was credited to Safran Menachem, Systems Programmer, Mivtachim Computers (Israel), © Xephon 1990. It is downloadable from Xephon’s Web site at www.xephon.com/cics.

The routine has not been changed in any significant way except to add EXEC CICS SUSPEND calls after every hundred or so iterations. This is to ensure that a call to sort does not monopolize the QR TCB, or become vulnerable to an AICA abend.

Debug output is written to a TD queue. The name of the queue is coded in the program (at my site we have a standard debugging destination called NBUG).

USAGE

This routine is designed to be invoked from a COBOL program using
the SORT verb, and the example below shows that usage. It should work correctly when called by any program meeting its interface requirements for invocation and E15/E35 exits. It is, however, very sensitive to the format of the parameter text and cannot handle free format input.

The following extract from a COBOL program shows how the sort can be invoked:

```
......
INPUT-OUTPUT SECTION.
FILE-CONTROL.
  * -----------------------------------------------
SELECT SD01-SORT-TERMINAL-FILE
ASSIGN TO SORTFILE.
DATA DIVISION.
  * -----------------------------------------------
FILE SECTION.
  * -----------------------------------------------
SD SD01-SORT-TERMINAL-FILE.
 01 SD01-SORT-TERMINAL-RECORD.
 05 SD01-LONG-KEY.
 10 SD01-SHORT-KEY.
 15 SD01-SHORT-KEY-9 PIC 9(8).
 10 SD01-KEY-FILLER PIC X(242).
 05 SD01-DATA-AREA.
 10 SD01-DATA-BYTE PIC X
OCCURS 1 TO 9999
DEPENDING ON RECLLEN.
......
PROCEDURE DIVISION.
......
COMPUTE SORT-FILE-SIZE = NUMRECS
END-COMPUTE
COMPUTE SORT-CORE-SIZE =
  ( SORT-FILE-SIZE * RECLLEN)
END-COMPUTE
......
SORT SD01-SORT-TERMINAL-FILE
ON Descending KEY SD01-LONG-KEY
INPUT PROCEDURE IS X1-SORT-INPUT
OUTPUT PROCEDURE IS Y1-SORT-OUTPUT
......
X1-SORT-INPUT.
  * -----------------------------------------------
MOVE NUMRECS TO HIGHLIMIT
PERFORM XI1-RELEASE-RECORD
```
VARYING NUMRECS FROM 1 BY 1
UNTIL NUMRECS > HIGHLIMIT
.
*
X11-RELEASE-RECORD.
*  --------------------------------------------------------
MOVE NUMRECS TO SD01-SHORT-KEY-9
RELEASE SD01-SORT-TERMINAL-RECORD
.
*
X-SORT-EXIT.
*  --------------------------------------------------------
EXIT
.
Y1-SORT-OUTPUT.
*  --------------------------------------------------------
PERFORM Y11-RETURN-RECORD
UNTIL NO-MORE-TERMINALS
*  GO TO Y-SORT-EXIT
.
*
Y11-RETURN-RECORD.
*  --------------------------------------------------------
RETURN SD01-SORT-TERMINAL-FILE
AT END
SET NO-MORE-TERMINALS TO TRUE
END-RETURN
.
*
Y-SORT-EXIT.
*  --------------------------------------------------------
EXIT
.

PERFORMANCE

Once I had completed and proved the routine, I wrote a test program so that I could analyse the performance of the sort. The following table shows approximate CPU times for a range of sorts. As with all CPU time statements, the numbers have to be taken with a pinch of salt, but the relationship between the tests should hold true, no matter what sort of CPU you are running on.

Figure 1 shows total transaction times for a test transaction with tests of small (260 byte) and large (10,000 byte) records. Figures were produced for a range of sort file sizes, from 5 records to 5000 records.
Because of performance evaluation during this testing, the routine will not sort sets of more than 5000 records. Sorts of large amounts of data (greater than the size of ECDSA) can make CICS go SOS, and so were eliminated from the test.

<table>
<thead>
<tr>
<th>#Records</th>
<th>Key length</th>
<th>Record length</th>
<th>Initial order</th>
<th>CPU time (milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>8</td>
<td>260</td>
<td>In order</td>
<td>6.5</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>260</td>
<td>Reversed</td>
<td>4.6</td>
</tr>
<tr>
<td>50</td>
<td>8</td>
<td>260</td>
<td>In order</td>
<td>19</td>
</tr>
<tr>
<td>50</td>
<td>8</td>
<td>260</td>
<td>Reversed</td>
<td>20</td>
</tr>
<tr>
<td>500</td>
<td>8</td>
<td>260</td>
<td>In order</td>
<td>180</td>
</tr>
<tr>
<td>500</td>
<td>8</td>
<td>260</td>
<td>Reversed</td>
<td>179</td>
</tr>
<tr>
<td>5000</td>
<td>8</td>
<td>260</td>
<td>In order</td>
<td>2,600</td>
</tr>
<tr>
<td>5000</td>
<td>8</td>
<td>260</td>
<td>Reversed</td>
<td>2,700</td>
</tr>
<tr>
<td>5</td>
<td>250</td>
<td>260</td>
<td>In order</td>
<td>4.7</td>
</tr>
<tr>
<td>5</td>
<td>250</td>
<td>260</td>
<td>Reversed</td>
<td>4.7</td>
</tr>
<tr>
<td>50</td>
<td>250</td>
<td>260</td>
<td>In order</td>
<td>19</td>
</tr>
<tr>
<td>50</td>
<td>250</td>
<td>260</td>
<td>Reversed</td>
<td>20</td>
</tr>
<tr>
<td>500</td>
<td>250</td>
<td>260</td>
<td>In order</td>
<td>184</td>
</tr>
<tr>
<td>500</td>
<td>250</td>
<td>260</td>
<td>Reversed</td>
<td>185</td>
</tr>
<tr>
<td>5000</td>
<td>250</td>
<td>260</td>
<td>In order</td>
<td>3,600</td>
</tr>
<tr>
<td>5000</td>
<td>250</td>
<td>260</td>
<td>Reversed</td>
<td>3,400</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>10000</td>
<td>In order</td>
<td>4.8</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>10000</td>
<td>Reversed</td>
<td>4.9</td>
</tr>
<tr>
<td>50</td>
<td>8</td>
<td>10000</td>
<td>In order</td>
<td>22</td>
</tr>
<tr>
<td>50</td>
<td>8</td>
<td>10000</td>
<td>Reversed</td>
<td>22</td>
</tr>
<tr>
<td>500</td>
<td>8</td>
<td>10000</td>
<td>In order</td>
<td>209</td>
</tr>
<tr>
<td>500</td>
<td>8</td>
<td>10000</td>
<td>Reversed</td>
<td>211</td>
</tr>
<tr>
<td>5</td>
<td>250</td>
<td>10000</td>
<td>In order</td>
<td>4.8</td>
</tr>
<tr>
<td>5</td>
<td>250</td>
<td>10000</td>
<td>Reversed</td>
<td>5.1</td>
</tr>
<tr>
<td>50</td>
<td>250</td>
<td>10000</td>
<td>In order</td>
<td>23</td>
</tr>
<tr>
<td>500</td>
<td>250</td>
<td>10000</td>
<td>Reversed</td>
<td>23</td>
</tr>
<tr>
<td>500</td>
<td>250</td>
<td>10000</td>
<td>In order</td>
<td>213</td>
</tr>
<tr>
<td>500</td>
<td>250</td>
<td>10000</td>
<td>Reversed</td>
<td>214</td>
</tr>
</tbody>
</table>

*Figure 1: Transaction times*

**DOCUMENTATION REFERENCE**

In writing this routine, I referred to the IBM COBOL manuals and the DFSORT programming guide:

- *COBOL for OS/390 and VM, Language Reference, SC26-9046-03*
- *COBOL for OS/390 and VM, Programming Guide, SC26-9049-04*
SOFTWARE PREREQUISITES
The program has been tested with COBOL for OS/390 and VM V1.2 and V2.1. It runs with CICS V4.1. I have no reason to expect problems with any version of CICS/TS, although it has not been tested there because my company has not yet implemented it. Software was:

- COBOL for MVS and VM
- Language Environment
- CICS V4.1 or higher.

Restrictions:
- Maximum key length = 250 bytes
- Maximum record length = 32000 bytes
- Key type = character (only)
- Maximum number of keys = 1
- Duplicate result order = undefined
- Sufficient storage for all keys to be stored in main memory (approximately 1.2MB at the extreme limit).

POSSIBLE ENHANCEMENTS
The program could be enhanced to remove many of its restrictions. The parser of the sort parameters is extremely primitive, and is optimized to the parameter text produced by COBOL with a very restricted subset of available options. A more flexible parser, together with the ability to implement the options, would make the routine more generally useful.

Many COBOL SORT verb options are not supported. These include multiple keys, the **Duplicates In Order** phrase, data types for the keys except for characters and others. These features could be added to the program, but would require a rewrite of the sort algorithm. It
probably needs to have a generalized comparison routine, rather than the distributed comparisons used at various points in the current implementation. An enhanced parser is also needed, so that the more extensive parameter text required to control these options can be interpreted.

The following features would provide useful additions to the tool:

- Implementation of several sort algorithms.
- Selection of an appropriate sort algorithm based on the input data set size.
- Selectable alternative collating sequences.

PROBLEMS

The data to be sorted is stored in a temporary storage queue. As implemented here, a MAIN TS queue is used. This can cause a problem, because the NOSUSPEND option is implemented only for auxiliary TS calls. If the data overflows the available ECDSA, then an SOS will occur in ECDSA, and the region will hang. The call could be changed to remove the MAIN option. This will require an increased TS dataset size, and increase the elapsed time for many sorts. It will, however, mean that the sort cannot send the region SOS, so it should probably be implemented without the MAIN option.

The original implementation of this program (inherited from the base sort function design) was targeted at an application that could run only against a terminal. The temporary storage queue used to save the data while the sort was in progress was named using the EIBTRMID to provide ‘uniqueness’. A subsequent implementation of the routine ran in a program that was initiated by an MQSeries message. This program ran without a primary facility. On several occasions, data belonging to two separate transactions was intermingled by sorts operating concurrently.

This has been resolved by changing the temporary storage queue name so that EIBTASKN is used in the name instead of EIBTRMID. As an additional precaution, an enqueue is performed against the queue name before entering the sort. If the enqueue fails, a failure return code is passed back to the application.
SORT ASSEMBLER

* SORT - This program implements a SORT function
* which can run only inside CICS. It is designed
* to be called via a COBOL SORT statement. The SORT
* statement must conform to the restrictions documented
* in the COBOL for MVS manual, regarding use of SORT
* under CICS.
* *
* Features implemented:
* Single key
* Ascending and Descending
* Fixed and variable length records
* E15 exit for record input
* E35 exit for record output
* In storage sort
* Number of records and Core size parameters
* Storage is acquired and freed as required
* QuickSort
* Key length between 1 and 250
* Record size up to 32000 bytes
*
* Standard sort features not supported:
* Input from and output to files (CICS restriction)
* Intermediate work files (CICS restriction)
* Large data sets. This routine limits the number of
* records input to the sort to 5000.
* Many sort techniques. This routine will always sort
* using a quicksort SORT as described in
* the Xephon Update article from 1990:
* 'A CICS sort utility program'
* Multiple keys
*
* The program parses the SORT CONTROL STATEMENTS, and
* expects it to be in a specific format (which is produced
* by COBOL).
* *
* The format expected is:
* ' SORT FIELDS=(0001,0008,CH,A) '  
* ' RECORD TYPE=F,LENGTH=(000067,,) '  
* ' OPTION MAINSIZE=00001005,FILESZ=E0000003 '  
* for fixed length records
* or
* ' SORT FIELDS=(0001,0008,CH,A) '  
* ' RECORD TYPE=V,LENGTH=(000079,,000071,) '  
* ' OPTION MAINSIZE=00001005,FILESZ=E0000003 '  
* for variable length records
* *
* The only field type is CH, Ascending or Descending
* Key location can be anywhere inside the record.
* Key location + key length must be < min record length
* Maximum key length is 250 bytes
*
* The Parse process for the sort control statements is
* very primitive, and cannot cope with free format
* input. It has been tested with COBOL for MVS, and
* LE for MVS V1.8. Changes to the sort control statement
* format between COBOL or LE versions will not be handled.
* A worthwhile improvement to the code would be to build
* a more capable parser, which at least coped with blank
* space between verbs, and with variable order of terms.
* It should also cope with optional values like those
* available in the LENGTH suboption, and with variable
* length numeric values.
*
* Further possible enhancements would be to choose an
* appropriate sort mechanism based on the file
* size, and to implement different sort algorithms.
* Currently, QuickSort is used. Alternatives would be
* ShellSort, HeapSort, TournamentSort, RadixSort, or
* even BubbleSort for very small data sets (<10-20).
*
* Alternative comparison types, for signed binary or
* packed decimal data would be useful, as would multiple
* independant key parts, so that a sequence could be
* ascending on one key, and descending on another
* subsidiary key.
*
**DFHREGS**

Build register equates

<table>
<thead>
<tr>
<th><strong>SORTPARM</strong></th>
<th><strong>DS</strong></th>
<th><strong>DSECT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>SORCNARL</td>
<td>D</td>
<td>DS</td>
</tr>
<tr>
<td>SCLEN</td>
<td>D</td>
<td>H</td>
</tr>
<tr>
<td>SCAREA</td>
<td>D</td>
<td>0C</td>
</tr>
</tbody>
</table>

* **SORTCMTL**

Sort Control Statement area

* **DFHEISTG**

Dynamic storage area

<table>
<thead>
<tr>
<th><strong>DFHEISTG</strong></th>
<th><strong>DSECT</strong></th>
<th><strong>DS</strong></th>
<th><strong>F</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RETCODE</strong></td>
<td><strong>DS</strong></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td><strong>PROCRET1</strong></td>
<td><strong>DS</strong></td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
<table>
<thead>
<tr>
<th>DS</th>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCRET2</td>
<td>A</td>
<td>Address to return after subroutine</td>
</tr>
<tr>
<td>PROCRET3</td>
<td>A</td>
<td>Address to return after subroutine</td>
</tr>
<tr>
<td>PARMLIST</td>
<td>A</td>
<td>Address of extended sort parm list</td>
</tr>
<tr>
<td><strong>PARSDATA</strong></td>
<td>F</td>
<td>Parsed information follows</td>
</tr>
<tr>
<td>KEYLEN</td>
<td>H</td>
<td>Length of the sort key</td>
</tr>
<tr>
<td>MAXKEY</td>
<td>EQU 250</td>
<td>Maximum size of key to sort</td>
</tr>
<tr>
<td>KEYOFFST</td>
<td>H</td>
<td>Location in the rec of sort key</td>
</tr>
<tr>
<td>CNTLGDIR</td>
<td>C</td>
<td>Control Statements parsed correctly</td>
</tr>
<tr>
<td>SORTKW</td>
<td>C</td>
<td>SORT Keyword handled ok</td>
</tr>
<tr>
<td>RECORDKW</td>
<td>C</td>
<td>RECORD Keyword handled ok</td>
</tr>
<tr>
<td>OPTIONKW</td>
<td>C</td>
<td>OPTION Keyword handled ok</td>
</tr>
<tr>
<td>SORTDIR</td>
<td>C</td>
<td>What direction is the sort (A/D)</td>
</tr>
<tr>
<td>RCFCM</td>
<td>C</td>
<td>Fixed or Variable</td>
</tr>
<tr>
<td>KEYLENA2</td>
<td>H</td>
<td>Length of the sort key</td>
</tr>
<tr>
<td>LRECL</td>
<td>H</td>
<td>Length of record (maximum)</td>
</tr>
<tr>
<td>MINRECL</td>
<td>H</td>
<td>Length of record (minimum)</td>
</tr>
<tr>
<td>QRECLEN</td>
<td>H</td>
<td>Length of record from TS queue</td>
</tr>
<tr>
<td>QCOUNT</td>
<td>H</td>
<td>Number of items in SORT queue</td>
</tr>
<tr>
<td>MAXSIZE</td>
<td>EQU 32760</td>
<td>Maximum size of record to sort</td>
</tr>
<tr>
<td>MAINSIZE</td>
<td>F</td>
<td>Recommended size of mainstore</td>
</tr>
<tr>
<td>MAXRECS</td>
<td>EQU 5000</td>
<td>Maximum records into sort.</td>
</tr>
<tr>
<td>TABPTR</td>
<td>A</td>
<td>Address of sort table</td>
</tr>
<tr>
<td>STCKPTR</td>
<td>A</td>
<td>Address work stack</td>
</tr>
<tr>
<td>STCKLEN</td>
<td>F</td>
<td>Length of work stack</td>
</tr>
<tr>
<td>SORTQNM</td>
<td>0CLB</td>
<td>Name of SORT TSQ</td>
</tr>
<tr>
<td>SORTQSR</td>
<td>CL4</td>
<td>Constant = NBST</td>
</tr>
<tr>
<td>SORTQTSK</td>
<td>PL4</td>
<td>Follow with EIBTASKN</td>
</tr>
<tr>
<td>FILESIZE</td>
<td>H</td>
<td>Number of records in file to sort</td>
</tr>
<tr>
<td><strong>PARSLEN</strong></td>
<td>EQU -PARSDATA</td>
<td>Length of the parsed information</td>
</tr>
<tr>
<td><strong>CURVALUE</strong></td>
<td>CL250</td>
<td>Current Value of key in sort</td>
</tr>
<tr>
<td><strong>TEMP</strong></td>
<td>CL252</td>
<td>Work space for swap</td>
</tr>
<tr>
<td><strong>RIGHT</strong></td>
<td>H</td>
<td>Pointer to right side of partition</td>
</tr>
<tr>
<td><strong>LEFT</strong></td>
<td>H</td>
<td>Pointer to left side of partition</td>
</tr>
<tr>
<td><strong>AIDLEFT</strong></td>
<td>H</td>
<td>Temp hold for left pointer</td>
</tr>
<tr>
<td><strong>WORKPACK</strong></td>
<td>PL8</td>
<td>Set alignment for CVB</td>
</tr>
<tr>
<td>E15PARM</td>
<td>0F</td>
<td>E15 exit parm list</td>
</tr>
<tr>
<td>E15NEWA</td>
<td>A</td>
<td>Address of new record</td>
</tr>
<tr>
<td>E15UEAC</td>
<td>A</td>
<td>Address of user exit address const</td>
</tr>
<tr>
<td>E35PARM</td>
<td>0F</td>
<td>E35 exit parm list</td>
</tr>
<tr>
<td>E35NEWA</td>
<td>A</td>
<td>Address of new record</td>
</tr>
<tr>
<td>E35OUTA</td>
<td>A</td>
<td>Address of record in out buffer</td>
</tr>
<tr>
<td>E35UEAC</td>
<td>A</td>
<td>Address of user exit address const</td>
</tr>
</tbody>
</table>

* 
SORT AMODE 31
SORT RMODE ANY
SORT DFHEIENT CODEREG=(2,11),
   DATAREG=(13),
   EIBREG=(3)
*
SORTINIT EQU *
   ST R1,PARMLIST     Save parm list pointer
   ST R1,PARMLIST     Get the EIB address
*
EXEC CICS ADDRESS EIB(DFHEIBR)
   ST DFHEIBR,DFHEIBP
   LA R4,8           Set bad return code for testing
   SR R4,R4          Clear a work register
   ST R4,RETCODE     Set default return code to zero
   MVC SORTQSR,='C'NBST'    Set the constant bit
   MVC SORTQTSK,EIBTASKN Set the variable bit
*
EXEC CICS ENQ
   RESOURCE(SORTQNM)  X
   LENGTH(=H'8')      X
   NOSUSPEND          X
   NOHANDLE
   CLC EIBRESP,DFHRESP(NORMAL) Was the ENQ successful?
   BNE RETURN         No... So sad. A return code has
                      already been set up.
*
SORTMAIN EQU *
   BAL R14,VALIDATE    Validate the sort parameter list
   BAL R14,PARSCNTL    Parse the control statements
   BAL R14,GETSTG      Acquire storage for the sort
   BAL R14,GETRECS     Get the records from the E15 exit
   BAL R14,DOSORT      Do the sort of the record pointers
   BAL R14,PUTRECS     Give the output to the E35 exit
   BAL R14,RELSTG      Release storage from the sort
   SR R14,R14         Zero
   ST R14,RETCODE     Zero RETCODE unless early term.
*
RETURN EQU *  
* Delete the queue (just in case)  
EXEC CICS DELETEQ TS  
           QUEUE(SORTQNM)  
           NOHANDLE  
* Delete the ENQ so the task can  
           Delete the ENQ so the task can  
           sort again.  
EXEC CICS DEQ  
           RESOURCE(SORTQNM)  
           LENGTH(=H'8')  
           NOHANDLE  
           L  R15,RETCODE  Get return code to pass back  
           L  R10,4(,R13)  Get previous save area  
           ST  R15,16(,R10)  Set the return code  
*  
           B  EXITPT  Give up and return to caller  
*  
* SUBROUTINES FOLLOW.  
*  
**********************************************************************  
VALIDATE EQU *  
* Validate that the parameter list  
* matches what we can cope with.  
*  
           ST  R14,PROCRETI  Save the subroutine return address  
           *  
VAL000 EQU *  
           Check parm list pointer  
           L  R5,PARMLIST  Address the sort parm list  
           USING  SORTPARM,R5  Map the sort parameters  
           LTR  R5,R5  Any Parm list passed?  
           BNZ  VAL010  Yes, continue checks  
           LA  R0,8  Generate value 8  
           ST  R0,RETCODE  Propagate into RETCODE  
EXEC CICS WRITEQ TD  
           QUEUE(NBUG)  
           FROM(SRTERRR00)  
           LENGTH(=AL2(L'SRTERRR00))  
           NOHANDLE  
*  
           B  RETURN  Give up immediately.  
           *  
VAL010 EQU *  
           Check control statement pointer  
*  
EXEC CICS WRITEQ TD  
           QUEUE(NBUG)  
* FROM(SPCSAA)  
* LENGTH(40)  
* NOHANDLE  
*
L R6,SPCSAA          Get Control statement area
USING SORTCNTL,R6   Map the sort control statements
LTR R6,R6           Any Sort Control Area passed?
BNZ VAL020          Yes, continue checks
LA R0,8             Generate value 8
ST R0,RETCODE       Propagate into RETCODE
EXEC CICS WRITEQ TD  X
   QUEUE(NBUG)       X
   FROM(SRTERR01)    X
   LENGTH(=AL2(L'SRTERR01)) X
   NOHANDLE
*
   B RETURN          Give up immediately.
*
VAL020 EQU *        Check E15 exit address
*
EXEC CICS WRITEQ TD  X
   QUEUE(NBUG)       X
   FROM(SORTCNTL)    X
   LENGTH(SCLEN)     X
   NOHANDLE
*
L R1,SPE15A         Get E15 exit address
C R1,='-1'          Has end of list been reached
BE VAL025           Yes, but too early - error
LTR R1,R1           Is it zero?
BNZ VAL030          Address provided, so continue
VAL025 EQU *        
LA R0,8             Generate value 8
ST R0,RETCODE       Propagate into RETCODE
EXEC CICS WRITEQ TD  X
   QUEUE(NBUG)       X
   FROM(SRTERR02)    X
   LENGTH(=AL2(L'SRTERR02)) X
   NOHANDLE
*
   B RETURN          Give up immediately.
*
VAL030 EQU *        Check E35 exit address
L R1,SPE35A         Get E35 exit address
C R1,='F'-1'       Has end of list been reached
BE VAL032           Yes, but too early - error
LTR R1,R1           Is it zero?
BNZ VAL035          Address provided, so continue
VAL032 EQU *        
LA R0,8             Generate value 8
ST R0,RETCODE       Propagate into RETCODE
EXEC CICS WRITEQ TD  X
QUEUE(NBUG) X
FROM(SRTRERR03) X
LENGTH(=AL2(L'SRTRERR03)) X
NOHANDLE

*  
B RETURN Give up immediately.

*  
VAL035 EQU * Check User Exit Address Constant
SR R1,R1 Null pointer
ST R1,E15UEAC Save default UEAC value
L R1,SPUEAC Get User Exit Addr Constant
C R1,='F'-1 Has end of list been reached
BE VAL500 Yes, so accept the parm list
ST R1,E15UEAC Save passed UEAC value

*  
VAL040 EQU * Check ALTSEQ translate table
L R1,SPALTSEQA Get ALTSEQ table address
C R1,='F'-1 Has end of list been reached
BE VAL500 Yes, so accept the parm list
LTR R1,R1 Is it zero?
BZ VAL045 No table, so continue
LA R0,8 Generate value 8
ST R0,RETCODE Propagate into RETCODE
EXEC CICS WRITEQ TD X
QUEUE(NBUG) X
FROM(SRTRERR04) X
LENGTH(=AL2(L'SRTRERR04)) X
NOHANDLE

*  
B RETURN Give up immediately.

*  
VAL045 EQU * Check ESTAE routine address
L R1,SPESTAEA Get ESTAE address
C R1,='F'-1 Has end of list been reached
BE VAL500 Yes, so accept the parm list

*  
VAL050 EQU * Check E18 exit address
L R1,STEP18A Get E18 exit address
C R1,='F'-1 Has end of list been reached
BE VAL500 Yes, so accept the parm list
LTR R1,R1 Is it zero?
BZ VAL060 No exit, so continue
LA R0,8 Generate value 8
ST R0,RETCODE Propagate into RETCODE
EXEC CICS WRITEQ TD X
QUEUE(NBUG) X
FROM(SRTRERR05) X
LENGTH(=AL2(L'SRTRERR05)) X
NOHANDLE

B   RETURN  Give up immediately.

*  VAL060  EQU  *  Check E99 exit address
L   R1,SPE39A  Get E99 exit address
C   R1,=F'-1'  Has end of list been reached
BE  VAL500  Yes, so accept the parm list
LTR  R1,R1  Is it zero?
BZ  VAL070  No exit, so continue
LA  R0,8  Generate value 8
ST  R0,RETCODE  Propagate into RETCODE
EXEC CICS WRITEQ TD  X
  QUEUE(NBUG)  X
  FROM(SRTERR06)  X
  LENGTH(=AL2(L'SRTERR06))  X
  NOHANDLE

*  B  RETURN  Give up immediately.

*  VAL070  EQU  *  Check CALL-ID field
CLC  SPCCALLID,=F'-1'  Get CALLID value
BE  VAL500  Yes, so accept the parm list

*  VAL080  EQU  *  Check correct end indicator
CLC  SPEND,=F'-1'  Is End indicator = x'FFFFFFFF'?  
BE  VAL500  Yes, so continue on
LA  R0,8  Generate value 8
ST  R0,RETCODE  Propagate into RETCODE
EXEC CICS WRITEQ TD  X
  QUEUE(NBUG)  X
  FROM(SRTERR07)  X
  LENGTH(=AL2(L'SRTERR07))  X
  NOHANDLE

*  B  RETURN  Give up immediately.

*  VAL500  EQU  *  Validation is done, return
L   R14,PROCRET1  Restore return address
BR  R14  Return to caller

*******************************************************************************
*  Check that we can understand the  
*  sort control statements. Set up  
*  control variables so that we do  
*  what we are told.  
*
**
*  PARSCNTL  EQU  *  Parse the Sort control statements
ST  R14,PROCRET1  Save the subroutine return address
**  **
*  **  Write out the sort control

**  PARS000  EQU  *  Set up everything before we start
LA R4,SCAREA
LH R7,SCLEN
MVI CNTLGOD,C'Y'
MVI SORTKW,C'N'
MVI RECORDK,C'N'
MVI OPTIONS,K,C'N'
BAL R14,PARS0

* PARS100 EQU *
  CLC =C'SORT ',0(R4)
  BNE PARS200
  BAL R14,PARS0

* PARS110 EQU *
  CLC =C'RECORD ',28(R4)
  BNE PARS200
  BAL R14,PARS0

* PARS120 EQU *
  CLC =C'OPTION ',61(R4)
  BNE PARS40
  BAL R14,PARS0

* We are done with the parse, now check how it went.

PARS200 EQU *
  CLI SORTKW,C'Y'
  BNE PARS40
  CLI RECORDK,C'Y'
  BNE PARS40
  CLI OPTIONS,K,C'Y'
  BNE PARS40
  B PARS500

* PARS400 EQU *
  MVI CNTLGOD,C'N'

* Write (txn, trm etc.)

MVC SRTR1B+26(4),EIBTRNID Record transaction
MVC SRTR1B+41(4),EIBTASKN Record terminal

EXEC CICS WRITEQ TD
  QUEUE(CSMT)
  FROM(SRTERR1B)
  LENGTH(=AL2(L'SRTERR1B'))
  NOHANDLE

* Write error message

EXEC CICS WRITEQ TD
  QUEUE(CSMT)
  FROM(SRTERR08)
  LENGTH(=AL2(L'SRTERR08'))
  NOHANDLE

* Write sort control statements

EXEC CICS WRITEQ TD X
    QUEUE(NBUG) X
    FROM(SORTCNTL) X
    LENGTH(SCLEN) X
    NOHANDLE
*
EXEC CICS ABEND X
ABCODE('SORT') X
NOHANDLE
*
  B RETURN Give up immediately.
*
PARS500 EQU *
    L R14,PROCRET1 Restore return address
    BR R14 Return to caller
*
PARS600 EQU * Skip over blanks, but stop at end
    ST R14,PROCRET2 Save the subroutine return address
*
PARS610 EQU * Skip over blanks, but stop at end
    CLI Ø(R4),C ' ' Is it space?
    BNE PARS620 No, so exit
    LA R4,1(R4) Increment pointer
    BCT R7,PARS610 Decrement count and branch back
*
PARS620 EQU *
    L R14,PROCRET2 Restore return address
    BR R14 Go back.
*
PARS700 EQU * Process the SORT keyword options
    ST R14,PROCRET3 Save the subroutine return address
    CLC =C 'FIELDS=('5(R4) Is FIELDS keyword at 6
    BNE PARS780 No - nothing else supported
*
PARS710 EQU * Process the FIELDS data
    PACK WORKPACK,13(4,R4) Assume 4 digit location then ','
pack the location field
    CVB R9,WORKPACK Now in binary
    BCTR R9,0 Decrement to form offset
    STH R9,KEYOFFST Save the key offset value
*
    CLI 17(R4),C ',', Comma next?
    BNE PARS780 Give up, can't parse it.
*
    PACK WORKPACK,18(4,R4) pack the key length field
    CVB R9,WORKPACK Now in binary
    STH R9,KEYLEN Save the key length value
    CH R9,=AL2(MAXKEY) Key length Ok?
    BNH PARS715 Yes, continue parse
* Bad key length - report

  EXEC CICS WRITEQ TD
  QUEUE(NBUG) X
  FROM(SRTERR10) X
  LENGTH(=AL2(L'SRTERR10)) X
  NOHANDLE
*

  B RETURN Give up immediately.
*

  PARS715 EQU *
  CLI 22(R4),C',,' Comma next?
  BNE PARS780 Give up, can't parse it.
*

  CLC =C'CH',23(R4) Is it character comparison?
  BNE PARS780 No, so we can't do it.
*

  CLI 25(R4),C',,' Comma again?
  BNE PARS780 Give up, can't parse it.
*

  CLI 26(R4),C'A' Ascending sort?
  BNE PARS720 No - is it descending?
  B PARS730 Yes, so save it
*

  PARS720 EQU *
  CLI 26(R4),C'D' Descending sort?
  BNE PARS780 No - Error, so give up
  B PARS730 Yes, so save it and skip forward
*

  PARS730 EQU *
  MVC SORTDIR,26(R4) Save the sort direction
  CLI 27(R4),C')' Followed by )? 
  BNE PARS780 No, so we can't cope with it
  MVI SORTKW,C'Y' Parsed the SORT statement OK.
*

  PARS780 EQU *
  SR R7,R7 Set parse length to zero
*

  PARS790 EQU *
  L R14,PROCRET3 Restore return address
  BR R14 Go back.
*

  PARS800 EQU * Process RECORD keyword
  ST R14,PROCRET3 Save the subroutine return address
  MVI RECFM,C' ' Clear RECFM
  SR R8,R8
  STH R8,LRECL Clear LRECL
*

  PARS810 EQU * Loop through RECORD keywords
  CLC =C'TYPE=',36(R4) Is it TYPE keyword?
  BNE PARS880 No - give up

* PARS820 EQU * Process the TYPE data
  * * *
  CLI 41(R4),C'F' Is it TYPE=F?
  BE PARS830 Yes, so we deal with it.
  CLI 41(R4),C'V' Is it TYPE=V?
  BNE PARS880 No, so we can't deal with it.
  * PARS830 EQU * Save the record type
  MVC RECFM,41(R4)
  CLI 42(R4),C',' Is it followed by comma
  BNE PARS880 No, so we must be done
  * CLC =C'LENGTH=(',43(R4) Is it LENGTH keyword
  BNE PARS880 No - nothing else supported
  * PARS840 EQU * Process the LENGTH=( data
  * * *
  PACK WORKPACK,51(6,R4) Pack the length value
  CVB R9,WORKPACK Make it binary
  STH R9,LRECL Save the (Max) record length
  STH R9,MINRECL Save the (possible) min lrecl
  CLI RECFM,C'F' Fixed format?
  BE PARS845 Yep, no more to do
  * * Variable format, so grab min len
  PACK WORKPACK,60(6,R4) Pack the length value
  CVB R9,WORKPACK Make it binary
  STH R9,MINRECL Save the (Min) record length
  LA R4,8(,R4) Increment over the min rec length
  * PARS845 EQU *
  LH R10,KEYLEN Get key length
  AH R10,KEYOFFST Add to position
  CR R9,R10 Key inside record?
  BNL PARS850 Yes, so continue
  * EXEC CICS WRITEQ TD X
    QUEUE(NBUG) X
    FROM(SRTERR11) X
    LENGTH(AL2(L'SRTERR11)) X
    NOHANDLE
  *
  B RETURN Give up immediately.
  *
  PARS850 EQU * Check record length v. max
  CH R9,=AL2(MAXSIZE) Is it <= max?
  BNH PARS860
  * * *
  EXEC CICS WRITEQ TD X
    QUEUE(NBUG) X
FROM(SRERR12) X
LENGTH(AL2(L'SRERR12)) X
NOHANDLE

* 
B RETURN Give up immediately.
*

PARS860 EQU * Keep on checking
CLC =C',),',58(R4) Is it followed by ',)?
BNE PARS880 No, so we can't handle it
B PARS890 Everything is done
*

PARS880 EQU *
SR R7,R7 Set parse length to zero
*
*
PARS890 EQU *
CLI RECFM,C' ' RECFM is set?
BE PARS895 No, so no good - don't set good
CLC LRECL,=H'0' LRECL set?
BE PARS895 No, so no good - don't set good
MVI RECORDKW,C'Y' Everything Ok. Flag good parms
*

PARS895 EQU *
L R14,PROCRET3 Restore return address
BR R14 Go back.
*
*
PARS900 EQU * Process OPTION keyword
ST R14,PROCRET3 Save the subroutine return address
SR R8,R8
STH R8,FILESIZE Clear File record count
ST R8,MAINSIZE Clear File record count
*

PARS910 EQU * Loop through OPTION keywords
CLC =C' MAINSIZE=' ,68(R4) Is it MAINSIZE?
BNE PARS980 Yes - process it
*

PARS920 EQU * Process the MAINSIZE data
* 
PACK WORKPACK,78(B8,R4) Pack it into work field
CVB R9,WORKPACK Convert it to binary
ST R9,MAINSIZE Save the mainstore size
CLI B6(R4),C',) Is it followed by comma
BNE PARS980 No, so we must be done

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

Neil Casey
Senior Systems Programmer (CICS and MQSeries)
National Australia Bank (Australia) © National Australia Bank 2001
# December 1999 – November 2001 index

Items below are references to articles that have appeared in *CICS Update* since Issue 169, December 1999. References show the issue number followed by the page number(s). Back-issues of *CICS Update* are available back to issue 169 (December 1999). See page 2 for details.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Issue Range</th>
<th>Page Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alter</td>
<td>177.32-47, 178.5-23</td>
<td>Forward recovery</td>
</tr>
<tr>
<td>AOR</td>
<td>174.14-20, 178.23-32</td>
<td>GLUE</td>
</tr>
<tr>
<td>Application development</td>
<td>172.3-10</td>
<td>Hot-pooling (HPJ)</td>
</tr>
<tr>
<td>Auto-install</td>
<td>177.3-9</td>
<td>Integrated Translator</td>
</tr>
<tr>
<td>Batch</td>
<td>172.11-19</td>
<td>ISC</td>
</tr>
<tr>
<td>BMS</td>
<td>187.46-51, 188.18-55</td>
<td>Java</td>
</tr>
<tr>
<td>CEDA</td>
<td>190.11-12</td>
<td>LE/370</td>
</tr>
<tr>
<td>CEDF</td>
<td>172.3</td>
<td>Libraries</td>
</tr>
<tr>
<td>CETR</td>
<td>174.3-14</td>
<td>Log manager</td>
</tr>
<tr>
<td>CICS TS 1.3</td>
<td>188.3-4, 192.25</td>
<td>Log stream</td>
</tr>
<tr>
<td>CICSplex SM API</td>
<td>170.30-47, 171.26-37</td>
<td>Maps</td>
</tr>
<tr>
<td>Cold start</td>
<td>169.16-20, 182.11-16, 184.10-11</td>
<td>172.19-28, 182.17-28, 183.6-18, 190.12-16, 190.24-47, 191.22-28</td>
</tr>
<tr>
<td>COPY</td>
<td>181.28-31</td>
<td>MQSeries</td>
</tr>
<tr>
<td>CPSM</td>
<td>192.8-10</td>
<td>MRO</td>
</tr>
<tr>
<td>CREATE command</td>
<td>173.3-10</td>
<td>National language</td>
</tr>
<tr>
<td>CVDA</td>
<td>185.3-11</td>
<td>Null-use resources</td>
</tr>
<tr>
<td>DB2</td>
<td>182.35-47</td>
<td>PL/I</td>
</tr>
<tr>
<td>DB2ENTRY</td>
<td>191.28-47</td>
<td>PL/I transactions</td>
</tr>
<tr>
<td>Define statements</td>
<td>175.15-31</td>
<td>Printer assignment</td>
</tr>
<tr>
<td>Display</td>
<td>184.5-10</td>
<td>RDO</td>
</tr>
<tr>
<td>Doctemplates</td>
<td>173.14-29, 174.21-29, 175.14</td>
<td>RDO definition</td>
</tr>
<tr>
<td>Documenting programs</td>
<td>188.9-17</td>
<td>Rununits</td>
</tr>
<tr>
<td>Dynamic Transaction Routing</td>
<td>176.26-38</td>
<td>Shared TS queues</td>
</tr>
<tr>
<td>E-business</td>
<td>181.32-47</td>
<td>SIT parameter</td>
</tr>
<tr>
<td>EJB</td>
<td>191.3-7</td>
<td>SMP/E zones</td>
</tr>
<tr>
<td>Enclaves</td>
<td>178.3-4</td>
<td>SMTP</td>
</tr>
<tr>
<td>Faxing</td>
<td>189.25-35</td>
<td>Sort</td>
</tr>
<tr>
<td>FOR</td>
<td>174.14-20, 178.23-32</td>
<td>SORT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Started regions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Storage management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Storage use</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stress</td>
</tr>
<tr>
<td></td>
<td></td>
<td>System logger</td>
</tr>
</tbody>
</table>
Need help with a CICS problem or project?

Maybe we can help:

- If it’s on a topic of interest to other subscribers, we’ll commission an article on the subject, which we’ll publish in *CICS Update*, and which we’ll pay for – it won’t cost you anything.

- If it’s a more specialized, or more complex, problem, you can advertise your requirements (including one-off projects, freelance contracts, permanent jobs, etc) to the thousands of CICS professionals who visit *CICS Update*’s home page every week. This service is also free of charge.

Visit the *CICS Update* Web site, http://www.xephon.com/cics, and follow the link to *Opportunities for CICS specialists*. 
Micro Focus International has announced enhancements to Component Generator that enable users to access CICS data and business processes across the latest platform technology. These updates include support for WAP and J2ME protocols for wireless devices and support for mapping mainframe data to XML Document Type Definitions (DTDs).

Component Generator enables IT organizations to capture complete CICS business processes and produce components in the form of JavaBeans, EJBs, COM, and XML.

Component Generator generates the entire component, middleware, and back-end source code needed to access CICS applications. The company claims that no hand coding or changes to legacy systems are required.

There’s now wireless connectivity to CICS applications and business workflows through WAP- and J2ME-enabled devices, including Palm, PocketPC, and mobile phones.

For further information contact:
Merant, Abbey View, Everard Close, St Albans, Herts, AL1 2PS, UK.
Tel: 01727 812812.

IBM has announced the second release of z/OS, with enhancements in IP networking, Internet and intranet security, and distributed print management for the Web.

A new function in the InfoPrint Server feature is designed to make it easy to incorporate legacy transaction programs into a printing infrastructure and to get output to the people who need it.

Infoprint Server converts SCS data and 3270 output from CICS, IMS, and other VTAM application programs directly to any PCL printer in the network. A new function enables transmission of output as e-mail.

For further information contact your local IBM representative.

---


For further details about CICS Update 2001, or information about Xephon’s complete range of seminars and conferences, call 01635 33823 or browse www.xephon.com/events.