February 1998

In this issue

3 Little-known features of API and SPI
9 The INQUIRE START command’s AT option
16 Setting the VSE return code – part 2
29 Converting macros to define statements
46 Screen viewing utility and extended attributes
48 CICS news

© Xephon plc 1998
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 £165.00 in the UK; $250.00 in the USA and Canada; £171.00 in Europe; £177.00 in Australasia and Japan; and £175.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1994 issue, are available separately to subscribers for £14.50 ($21.50) each including postage.

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.
INTRODUCTION
The Application Programming Interface (API) and the System Programming Interface (SPI) for CICS contain a very rich set of functionality. In fact, these two programming interfaces are so fertile that it is hard to keep track of all the useful options.

The API and SPI have evolved as the obsolete macro-level interface was eliminated and customer demands rose. This is why there are so many different possibilities. Many of the options are obscure, but quite useful in the appropriate circumstances.

This is the first of a series of articles to illustrate some of the more useful, but not commonly used, options and features of the API and SPI. The second article starts on page 9 of this issue, and three more will follow in due course. The discussion is based on actual field experience over the years and emphasizes how these esoteric properties can be applied to system and application requirements. A partial discussion of these commands and programs was presented at Xephon’s CICS Update 97 conference, held in London in December 1997. This article includes the full program source code.

The main topic of this article is the use of the RETURN command with the IMMEDIATE option.

The source code language used to illustrate the concepts is COBOL written to ANSI 85 standards.

RETURN IMMEDIATE
A frequent requirement of applications is that they change the CICS transaction code as they enter different phases of processing. This is often needed for integrated menu-driven systems covering a wide variety of application areas. Traditionally, this switch from, say, the menu program to the chosen application subsystem was performed by a START command followed by a terminating RETURN.
Unfortunately, this has an undesirable side-effect, causing a termination of the SNA bracket (conversation) with the terminal, with an ensuing SNA BID to request permission of the terminal (the ‘first-speaker’) to initiate a conversation. Not only does this involve additional network flow, it also allows for the possibility that the user might cause the BID to be rejected by pressing some key before the BID is received.

The solution is to combine the two operations into a single command. IBM has provided the IMMEDIATE option to the RETURN command for just this purpose. Now one RETURN command can be used in place of the START and RETURN sequence. The RETURN IMMEDIATE causes CICS to keep the SNA bracket (conversation) open, which means that no BID is required and no user action can interrupt the application flow.

To illustrate the use of the RETURN IMMEDIATE, and to embellish on the theme, I have written a sample program which implements a ‘shortcut’ CEMT SET PROGRAM() NEWCOPY request. It assumes that the programs being refreshed use a naming convention where the first six characters can be anything (but will be the same as those in the name of the sample program) and end with a two digit numeric value. In addition to discussing the RETURN IMMEDIATE, I shall also consider how you can distinguish formatted input from unformatted input, and the SEND CONTROL command.

The program does not use BMS and expects its initial input to be from a ‘clear’ CICS screen. However, that may not always be the case. The initial input for a transaction may include formatting characters, depending on what the previous transaction sent to the terminal. Therefore, any program expecting unformatted initial input needs to allow for that. If the data does contain 3270 datastream formatting characters, the datastream will be three characters longer than if it does not. The additional characters precede the actual data.

The syntax of the transaction is:

```
TTTTb99
```

where ‘TTTT’ is the transaction code, ‘b’ is a blank (space), and ‘99’ is the program number. The description of the input message area, to allow for both formatted and unformatted input, is coded in the
program as WS-INPUT. After the RECEIVE command, this structure allows the program to determine the type of input by testing for GOT-SBA. (Arbitrarily this program assumes that the program range is from 11 through 88 inclusive. These limits can be changed by modifying the VALUE clauses for WS-LOWER-LIMIT and WS-UPPER-LIMIT.)

After verifying that the syntax of the input is valid, the program then issues a RETURN IMMEDIATE command. This requires the structure coded in the program as WS-CEMT-DATA. CICS then ends the current task and initiates CEMT to process the new action as if the user had entered it directly on his/her terminal. (Note that this program can be used to refresh itself!)

Note that the SEND CONTROL command can be used to manage options that the ordinary SEND FROM command cannot. These include options such as the position of the cursor (CURSOR), the sounding of the audible alarm (ALARM), the releasing of the keyboard (FREEKB), and the resetting of the modified data tags (FRSET). This program uses it to position the cursor if the input is invalid.

PROGRAM SOURCE

IDENTIFICATION DIVISION.
PROGRAM-ID. SAMPLE.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
   01 FILLER.
      03 WS-INPUT-LTH PIC S9(4) COMP.
      03 WS-LOWER-LIMIT PIC S9(4) COMP VALUE 11.
      03 WS-UPPER-LIMIT PIC S9(4) COMP VALUE 88.
      03 WS-CURSOR-POS PIC S9(4) COMP VALUE 4.
      03 WS-SYNTAX-MSG.
         05 FILLER PIC X(24) VALUE
            'Syntax of transaction: '''.
         05 WS-SM-TRAN PIC X(04).
         05 FILLER PIC X(04) VALUE
            ' 99'''.
      03 WS-NEED-PROG-MSG.
         05 FILLER PIC X(30) VALUE
            'Need a program number between '.
         05 WS-NP-LOWER PIC 9(02).
         05 FILLER PIC X(05) VALUE
' and '.
05 WS-NP-UPPER PIC 9(02).
05 FILLER PIC X(10) VALUE ' as input.'.
03 WS-NEED-SPACE-MSG PIC X(52) VALUE 'Need a space between the transaction and the number.'.

01 WS-INPUT.
03 WS-SBA.
05 WS-SBA-POS PIC X(01).
  08 G0T-SBA VALUE X'11'.
05 FILLER PIC X(02).
05 WS-SBA-DATA.
  07 FILLER PIC X(04).
  07 WS-SBA-SPACE PIC X(01).
  07 WS-SBA-PROG PIC X(02).
03 WS-NOSBA REdefines WS-SBA.
05 WS-NOSBA-DATA.
  07 FILLER PIC X(04).
  07 WS-NOSBA-SPACE PIC X(01).
  07 WS-NOSBA-PROG PIC X(02).

01 WS-OUTPUT.
03 WS-INPUT-DATA PIC X(07).
03 FILLER PIC X(05) VALUE SPACES.
03 WS-HELP-MSG PIC X(52).

01 WS-CEMT-DATA.
03 FILLER PIC X(12) VALUE 'CEMT S PROG('.
03 WS-CD-PROGRAM.
05 FILLER PIC X(06).
05 WS-CD-NUMBER-X.
  07 WS-CD-NUMBER PIC 9(02).
03 FILLER PIC X(05) VALUE ') NEW'.

PROCEDURE DIVISION.
  EXEC CICS ASSIGN
        PROGRAM(WS-CD-PROGRAM)
  END-EXEC
  MOVE WS-LOWER-LIMIT TO WS-NP-LOWER
  MOVE WS-UPPER-LIMIT TO WS-NP-UPPER
  *
  * Get the input from the user.
  *
  MOVE LENGTH OF WS-INPUT TO WS-INPUT-LTH
  EXEC CICS RECEIVE
        INTO(WS-INPUT)
        LENGTH(WS-INPUT-LTH)
* Allow for the possibility of formatted or unformatted input.

IF GOT-SBA
  MOVE WS-SBA-DATA TO WS-INPUT-DATA
  IF WS-INPUT-LTH < LENGTH OF WS-SBA
    PERFORM BAD-LENGTH
  ELSE
    IF WS-SBA-SPACE NOT = SPACE
      PERFORM BAD-SPACE
    END-IF
    IF WS-SBA-PROG NOT NUMERIC
      PERFORM BAD-PROG
    ELSE
      MOVE WS-SBA-PROG TO WS-CD-NUMBER-X
    END-IF
  END-IF
ELSE
  MOVE WS-NOSBA-DATA TO WS-INPUT-DATA
  IF WS-INPUT-LTH < LENGTH OF WS-NOSBA
    PERFORM BAD-LENGTH
  ELSE
    IF WS-NOSBA-SPACE NOT = SPACE
      PERFORM BAD-SPACE
    END-IF
    IF WS-NOSBA-PROG NOT NUMERIC
      PERFORM BAD-PROG
    ELSE
      MOVE WS-NOSBA-PROG TO WS-CD-NUMBER-X
    END-IF
  END-IF
END-IF
IF WS-CD-NUMBER < WS-LOWER-LIMIT
  OR WS-CD-NUMBER > WS-UPPER-LIMIT
  PERFORM BAD-PROG
END-IF
EXEC CICS RETURN IMMEDIATE
  TRANSID('CEMT')
  INPUTMSG(WS-CEMT-DATA)
END-EXEC

BAD-LENGTH.
  MOVE EIBTRNID TO WS-SM-TRAN
  MOVE WS-SYNTAX-MSG TO WS-HELP-MSG
  PERFORM BAD-INPUT

BAD-PROG.
  ADD 1 TO WS-CURSOR-POS
Call for papers

Why not share your expertise and earn money at the same time? CICS Update is looking for JCL, macros, program code, etc, that experienced CICS users have written to make their life, or the lives of their users, easier. We will publish it (after vetting by our expert panel) and send you a cheque when the article is published. Articles can be of any length and can be sent or e-mailed to Robert Burgess at any of the addresses shown on page 2. Why not call now for a free copy of our Notes for contributors?
The INQUIRE START command’s AT option

INTRODUCTION
This is the second of the series of articles illustrating some of the options and features of the API and SPI which started with the first article in this issue.

The main topic of this article is the use of the INQUIRE START command with the AT option.

The source code language used to illustrate the concepts is COBOL written to ANSI 85 standards.

INQUIRE START AT
Many installations continually encounter the problem of Auxiliary Temporary Storage (TS) filling up. When this occurs, many systems apparently ‘lock up’ because of the ubiquitous use of TS by applications. Prior to CICS/ESA Version 4.1, the only way to discover what TS queues existed was by implementing a scan of the names via the browsing capabilities of the SPI INQUIRE TSQUEUE command. In Version 4.1 and beyond you can scan them using CEMT.

To overcome the limitations of Version 3, I have written a sample program which performs a display of the Auxiliary TS queues. However, the techniques used in the program can easily be adapted to other requirements.

There are three main aspects of the program I wish to discuss.

The first point is that it uses a GETMAIN command, to obtain storage for what will become DFHCOMMAREA, instead of defining the data twice. The majority of CICS COBOL application programs define the data saved between pseudo-conversational tasks in the WORKING-STORAGE SECTION. This data is refreshed at task start-up from DFHCOMMAREA in the LINKAGE SECTION. The use of the GETMAIN command overcomes the maintenance problem many people experience when changing the size of the area saved between
tasks. It requires DFHCOMMAREA to be defined with the full layout of the data to be saved as coded in the sample program. It also requires the logic as coded in the program in the ‘IF EIBCALEN = ZERO’ statement.

The second detail concerns the interesting fact that a selected set of CICS resources (PROGRAMs, TSQUEUEs, TRANSACTIONs, and TRANsaction CLASSes) are stored by CICS in alphabetic sequence. This means that browsing INQUIRiEs on these can begin at somewhere other than the beginning of the list. In the sample program, the designed display area is limited to 47 queues at a time. So, in order to implement the task in a pseudo-conversational manner, the program needs to be able to position itself into the middle of the list of queues after the first 47 queues have been displayed. It does this by using an INQUIRE TSQUEUE START AT (CA-LAST-QUEUE) command. CA-LAST-QUEUE is initially nulls (LOW-VALUES) as a result of the GETMAIN discussed; it is updated as the task proceeds, and then saved via the COMMAREA option of the RETURN command. Of course, any program using this needs to use the SP translator option, which is why the CBL XOPTS(SP) statement is included as the first line of the source.

The third aspect concerns the fact that TS queues may contain non-printable/displayable EBCDIC characters. A conversion to all printable characters must be done to prevent invalid character sequences being sent to the device. To overcome this problem, techniques are included in the program so that all queue names are displayed in characters suitably translated to printable ones, as well as in hexadecimal characters, to be able to detect the true identity of each queue.

A couple of minor points to note are that the program limits the display to Auxiliary TS queues only, and that the output 3270 datastream contains a 5-character sequence needed to display protected data, beginning in the upper left hand corner of the screen.

PROGRAM SOURCE

CBL XOPTS(SP)
IDENTIFICATION DIVISION.
PROGRAM-ID. SAMPLE.
ENvironMent DIVISION.
DATA DIVISION.
WORKING-STORAGe SECTION.
  Ø1 WS-OUTPUT.
  Ø3 FILLER                      PIC X(Ø1) VALUE X'11'.
  Ø3 FILLER                      PIC X(Ø2) VALUE 'A'.
  Ø3 FILLER                      PIC X(Ø1) VALUE X'1D'.
  Ø3 FILLER                      PIC X(Ø1) VALUE 'Ø'.
  Ø3 FILLER                      PIC X(16) VALUE
  '(PF3 to Exit)'.
  Ø3 WS-MORE                     PIC X(Ø7) VALUE SPACES.
  Ø3 FILLER                     PIC X(15) VALUE SPACES.
  Ø3 FILLER                  VALUE LOW-VALUES.
  Ø5 WS-QUEUE-INFO                     OCCURS 47
    INDEXED BY WS-QI-INDEX.
    Ø7 WS-QI-C                     PIC X(Ø1).
    Ø7 WS-QI-A1                    PIC X(Ø1).
    Ø7 WS-QI-CHAR                   PIC X(Ø1) OCCURS 8
      INDEXED BY WS-QC-INDEX.
    Ø7 WS-QI-A2                    PIC X(Ø1).
    Ø7 FILLER                     PIC X(Ø2).
    Ø7 WS-QI-X                     PIC X(Ø1).
    Ø7 WS-QI-A3                    PIC X(Ø1).
    Ø7 WS-QI-HEX                   PIC X(Ø2) OCCURS 8
      INDEXED BY WS-QH-INDEX.
    Ø7 WS-QI-A4                    PIC X(Ø1).
    Ø7 FILLER                     PIC X(Ø8).
  Ø1 FILLER.
  Ø3 WS-LOC                      PIC S9(8) COMP.
  Ø3 WS-TABLE-IX                PIC S9(8) COMP.
  Ø3 FILLER REDEFINES WS-TABLE-IX.
    Ø5 FILLER                  PIC X(Ø3).
    Ø5 WS-TI-VAL               PIC X(Ø1).
  Ø3 WS-EBCDIC-TABLE.
    Ø5 FILLER                PIC X(16) VALUE
      '.................'.
    Ø5 FILLER                PIC X(16) VALUE
      '.................'.
    Ø5 FILLER                PIC X(16) VALUE
      '.................'.
    Ø5 FILLER                PIC X(16) VALUE
      '.................'.
    Ø5 FILLER                PIC X(16) VALUE
      '.................'.
    Ø5 FILLER                PIC X(16) VALUE
      '\.\<\(+|'.
    Ø5 FILLER                PIC X(16) VALUE
      '&...........|$*);,'.
    Ø5 FILLER                PIC X(16) VALUE
      '-/........&_?'.
COPY DFHAID.

LINKAGE SECTION.

PROCEDURE DIVISION.
  IF EIBAID = DFHPF3
    PERFORM DONE-EM
  END-IF
  IF EIBCALEN = ZERO
    EXEC CICS GETMAIN
    LENGTH(LENGTH OF DFHCOMMAREA)
    SET (ADDRESS OF DFHCOMMAREA)
    INITIMG(WS-INIT-VAL)
  END-EXEC
  END-IF
  IF CA-LAST-QUEUE = HIGH-VALUES
    PERFORM DONE-EM
  END-IF
  EXEC CICS INQUIRE
    TSQUEUE START
    AT(CA-LAST-QUEUE)
    NOHANDLE
  END-EXEC
  SET WS-QI-INDEX TO 1

  * This program is fairly basic in its handling of the display.
  *
  PERFORM UNTIL EIBRESP = DFHRESP(END)
    OR WS-QI-INDEX > 47
    EXEC CICS INQUIRE
    TSQUEUE(CA-LAST-QUEUE)
    NEXT
    LOCATION(WS-LOC)
    NOHANDLE
  END-EXEC
EVALUATE EIBRESP
  WHEN DFHRESP(NORMAL)
    IF WS-LOC = DFHVALUE(AUXILIARY)
      MOVE SPACES TO WS-QUEUE-INFO(WS-QI-INDEX)
      MOVE 'C' TO WS-QI-C (WS-QI-INDEX)
      MOVE 'X' TO WS-QI-X (WS-QI-INDEX)
      MOVE WS-APOST TO WS-QI-A1 (WS-QI-INDEX)
                        WS-QI-A2 (WS-QI-INDEX)
                        WS-QI-A3 (WS-QI-INDEX)
                        WS-QI-A4 (WS-QI-INDEX)
      SET WS-QC-INDEX TO 1
      SET WS-QH-INDEX TO 1
      PERFORM VARYING CA-QC-INDEX FROM 1 BY 1
        UNTIL CA-QC-INDEX > 8
        MOVE CA-LAST-QUEUE-CHAR(CA-QC-INDEX)
           TO WS-TI-VAL
        ADD 1 TO WS-TABLE-IX
        MOVE WS-EBCDIC-CHAR(WS-TABLE-IX)
           TO WS-QI-CHAR(WS-QI-INDEX, WS-QC-INDEX)
        MOVE WS-HEX-CHARS (WS-TABLE-IX)
           TO WS-QI-HEX (WS-QI-INDEX, WS-QH-INDEX)
      SET WS-QC-INDEX UP BY 1
      SET WS-QH-INDEX UP BY 1
      END-PERFORM
      SET WS-QI-INDEX UP BY 1
    END-IF
  WHEN DFHRESP(END)
    MOVE HIGH-VALUES TO CA-LAST-QUEUE
  WHEN OTHER
    PERFORM FATAL-ERROR
  END-EVALUATE
END-PERFORM
IF CA-LAST-QUEUE NOT = HIGH-VALUES
  MOVE 'More...' TO WS-MORE
END-IF
EXEC CICS SEND
  FROM(WS-OUTPUT)
  ERASE
END-EXEC
EXEC CICS RETURN
  TRANSID(EIBTRNID)
  COMMAREA(DFHCOMMAREA)
END-EXEC
.
DONE-EM.
EXEC CICS SEND
  FROM(WS-END)
  ERASE
END-EXEC
PERFORM RET
The next article in this series will continue the theme of using some of the useful but uncommonly used options and features of the API and SPI.

Jerry Ozaniec
Circle Computer Group (UK) © Xephon 1998

Subscribers who want copies of the code from this issue can call our Web site — www.xephon.com — and ask for the article they require. The article will then be e-mailed to them. This service is free to subscribers. Subscribers will need their user-id (which is on the mailing label on the envelope containing this issue), and they will need a copy of this issue so that they can answer a simple question (this is to prevent non-subscribers accessing information that subscribers have paid for).
Setting the VSE return code – part 2

This month we continue the program to set the VSE return code during CICS start-up and normal shut-down, so that conditional JCL can be used to restart it automatically if the CICS system has terminated abnormally. It also determines whether DTSANALS needs to be run and, if it does, submits a job to perform a RECOVER function.

```
CKBTH EQU *
CLI INPPGMN,C' ' IS FIRST POSITION OF PROGRAM NAME BL
BE CKPRG1 YES-BRANCH TO CKPRG1.
CLI INPJOBN,C' ' IS FIRST POSITION OF JOB NAME BLANK.
BE CKJOB1 YES-BRANCH TO CKJOB1.

CKBTH3 EQU *
CLI ESASW,C'Ø' ARE WE RUNNING UNDER VSE/ESA.
BE CKBTH5 NO-BRANCH TO CKBTH5.
BAL RB,CKDYN PERFORM CKDYN ROUTINE.
BAL RB,CKSJBJ+4 PERFORM CKSJBJ+4 ROUTINE.
BAL RB,CKMTC PERFORM CKMTC ROUTINE.
CLI MTCSW,C'1' DID WE HAVE A MATCH ON JOB NAME.
BNE CKBTH3 NO-BRANCH TO CKBTH3.
BAL RB,CKSPG+4 PERFORM CKSPG+4 ROUTINE.
BAL RB,CKMTC PERFORM CKMTC ROUTINE.
CLI MTCSW,C'1' DID WE HAVE A MATCH ON PROGRAM NAME.
BNE CKBTH3 NO-BRANCH TO CKBTH3.
MVI INPRCDE,C'3' INDICATE WE'VE FOUND JOB/PROGRAM NAME
MVC Ø(L'PIBLOGID,R9),2(R7) MVE SYSLOG ID.
LA R9,L'PIBLOGID(R9) INCREMENT REG 9 TO NEXT POSITION.
BAL RB,CKMVE6 PERFORM CKMVE6 ROUTINE.
B CKBTH3 BRANCH TO CKBTH3.

CKBTH5 EQU *
LM R5,R8,SVREGS RESTORE REGS 5 THRU 8.

CKBTH53 EQU *
BAL RB,CKLOP PERFORM CKLOP ROUTINE.
BAL RB,CKSJBJ PERFORM CKSJBJ ROUTINE.
BAL RB,CKMTC PERFORM CKMTC ROUTINE.
CLI MTCSW,C'1' DID WE HAVE A MATCH ON JOB NAME.
BNE CKBTH53 NO-BRANCH TO CKBTH53.
BAL RB,CKSPG PERFORM CKSPG ROUTINE.
BAL RB,CKMTC PERFORM CKMTC ROUTINE.
CLI MTCSW,C'1' DID WE HAVE A MATCH ON PROGRAM NAME.
BNE CKBTH53 NO-BRANCH TO CKBTH53.
MVI INPRCDE,C'3' INDICATE WE'VE FOUND JOB/PROGRAM NAME
MVC Ø(L'PIBLOGID,R9),2(R7) MVE SYSLOG ID.
LA R9,L'PIBLOGID(R9) INCREMENT REG 9 TO NEXT POSITION.
BAL RB,CKMVE PERFORM CKMVE ROUTINE.
B CKBTH53 BRANCH TO CKBTH53.
```
**CKPRG**  EQU  *  
CLI  INPPGMN,C' '  IS FIRST POSITION OF PROGRAM NAME BL
BNE  CKPRG3  NO-BRANCH TO CKPRG3.
CKPRG1  EQU  *  
MVI  INPRCDE,C'6'  INDICATE PROGRAM NAME ERROR.
B  CKRTN9  BRANCH TO CKRTN9.
CKPRG3  EQU  *  
CLI  ESASW,C'Ø'  ARE WE RUNNING UNDER VSE/ESA.
BE  CKPRG5  NO-BRANCH TO CKPRG5.
BAL  RB,CKDYN  PERFORM CKDYN ROUTINE.
BAL  RB,CKSPG+4  PERFORM CKSPG+4 ROUTINE.
BAL  RB,CKMTC  PERFORM CKMTC ROUTINE.
CLI  MTCSW,C'1'  DID WE HAVE A MATCH ON PROGRAM NAME.
BNE  CKPRG3  NO-BRANCH TO CKPRG3.
MVI  INPRCDE,C'2'  INDICATE WE'VE FOUND PROGRAM NAME.
MVC  Ø(L'PIBLOGID,R9),2(R7)  MVE SYSLOG-ID.
LA  R9,L'PIBLOGID(R9)  INCREMENT REG 9 TO NEXT POSITION.
BAL  RB,CKMVE6  PERFORM CKMVE6 ROUTINE.
B  CKPRG3  BRANCH TO CKPRG3.
CKPRG5  EQU  *  
LM  R5,R8,SVREGS  RESTORE REGS 5 THROUGH 8.
CKPRG53  EQU  *  
BAL  RB,CKLOP  PERFORM CKLOP ROUTINE.
BAL  RB,CKSPG  PERFORM CKSPG ROUTINE.
BAL  RB,CKMTC  PERFORM CKMTC ROUTINE.
CLI  MTCSW,C'1'  DID WE HAVE A MATCH ON PROGRAM NAME.
BNE  CKPRG53  NO-BRANCH TO CKPRG53.
MVI  INPRCDE,C'2'  INDICATE WE'VE FOUND PROGRAM NAME.
MVC  Ø(L'PIBLOGID,R9),2(R7)  MVE SYSLOG-ID.
LA  R9,L'PIBLOGID(R9)  INCREMENT REG 9 TO NEXT POSITION.
BAL  RB,CKMVE6  PERFORM CKMVE ROUTINE.
B  CKPRG53  BRANCH TO CKPRG53.
CKJOB  EQU  *  
CLI  INPJOBN,C' '  IS FIRST POSITION OF JOB NAME BLANK.
BNE  CKJOB3  NO-BRANCH TO CKJOB3.
CKJOB1  EQU  *  
MVI  INPRCDE,C'6'  INDICATE JOB NAME ERROR.
B  CKRTN9  BRANCH TO CKRTN9.
CKJOB3  EQU  *  
CLI  ESASW,C'Ø'  ARE WE RUNNING UNDER VSE/ESA.
BE  CKJOB5  NO-BRANCH TO CKJOB5.
BAL  RB,CKDYN  PERFORM CKDYN ROUTINE.
BAL  RB,CKSJB+4  PERFORM CKSJB ROUTINE.
BAL  RB,CKMTC  PERFORM CKMTC ROUTINE.
CLI  MTCSW,C'1'  DID WE HAVE A MATCH ON JOB NAME.
BNE  CKJOB3  NO-BRANCH TO CKJOB3.
MVI  INPRCDE,C'1'  INDICATE WE'VE FOUND JOB NAME.
MVC  Ø(L'PIBLOGID,R9),2(R7)  MVE SYSLOG-ID.
LA  R9,L'PIBLOGID(R9)  INCREMENT REG 9 TO NEXT POSITION.
BAL  RB,CKMVE6  PERFORM CKMVE ROUTINE.

B     CKJOB3  BRANCH TO CKJOB3.
CKJOB5  EQU *  BRANCH TO CKJOB5.
LM  R5,R8,SVREGS  RESTORE REGS 5 THROUGH 8.
CKJOB53  EQU *  BRANCH TO CKJOB53.
BAL  R8,CKLOP  PERFORM CKLOP ROUTINE.
BAL  R8,CKSJB  PERFORM CKSJB ROUTINE.
BAL  R8,CKMTC  PERFORM CKMTC ROUTINE.
CLI  MTCSW,C'1'  DID WE HAVE A MATCH ON JOB NAME.
BNE  CKJOB53  NO-BRANCH TO CKJOB53.
MVI  INPRCDE,C'1'  INDICATE WE'VE FOUND JOB NAME.
MVC  Ø(L'PIBLOGID,R9),2(R7)  MVE SYSLOG-ID.
LA  R9,L'PIBLOGID(R9)  INCREMENT REG 9 TO NEXT POSITION.
BAL  R8,CKMVE  PERFORM CKMVE ROUTINE.
B  CKJOB53  BRANCH TO CKJOB53.
CKRTN  EQU *  BRANCH TO CKRTN.
CLI  OPTN2,C'Y'  DO WE RETURN OUTPUT IN PRTY ORDER.
BNE  CKRTN5  NO-BRANCH TO CKRTN5.
CLI  INPPCNT+1,X'Ø1'  MORE THAN ONE (1) PARTITION.
BNH  CKRTN5  NO-BRANCH TO CKRTN5.
LA  R6,TAB  LOAD ADDRESS OF TAB TO REG 6.
LA  R7,INPPIDS  LOAD ADDRESS OF INPPIDS TO REG 7.
CKRTN3  EQU *  BRANCH TO CKRTN3.
CLI  1(R6),C' '  IS PARTITION-ID BLANK.
BE  CKRTN33  YES-BRANCH TO CKRTN33.
MVC  Ø(L'PIBLOGID,R7),1(R6)  REPLACE PARTITION-ID WITH PRTY OR
LA  R7,L'PIBLOGID(R7)  INCREMENT REG 7 TO NEXT POSITIONS.
CKRTN33  EQU *  BRANCH TO CKRTN33.
LA  R6,L'TAB(R6)  INCREMENT REG 6 TO NEXT POSITIONS.
CLI  Ø(R6),X'FF'  ARE WE DONE.
BNE  CKRTN3  NO-BRANCH TO CKRTN3.
CKRTN5  EQU *  BRANCH TO CKRTN5.
CLI  NUMPRM,X'Ø2'  WERE TWO (2) PARAMETERS PASSED.
BNE  CKRTNW  NO-BRANCH TO CKRTNW.
CLI  OPTN2,C'Y'  DO WE RETURN OUTPUT IN PRTY ORDER.
BNE  CKRTN7  NO-BRANCH TO CKRTN7.
CLI  INPPCNT+1,X'Ø1'  MORE THAN ONE (1) PARTITION.
BNH  CKRTN7  NO-BRANCH TO CKRTN7.
LA  R6,TAB  LOAD ADDRESS OF TAB TO REG 6.
LA  R7,INPJBPG  LOAD ADDRESS OF INPJBPG TO REG 7.
LA  R8,INPCOMR  LOAD ADDRESS OF INPCOMR TO REG 8.
CKRTN6  EQU *  BRANCH TO CKRTN6.
CLI  1(R6),C' '  IS PARTITION-ID BLANK.
BE  CKRTN63  YES-BRANCH TO CKRTN63.
MVC  Ø(L'INPJOBN+L'INPPGMN,R7),5(R6)  REPLACE JOB/PROGRAM NAME
MVC  Ø(L'PIBLOGID+2,R8),21(R6)  REPLACE PARTITION COMMUNICATION
LA  R7,L'INPJOBN+L'INPPGMN(R7)  INCREMENT REG 7 TO NEXT POSITION.
LA  R8,L'PIBLOGID+2(R8)  INCREMENT REG 8 TO NEXT POSITIONS.
CKRTN63  EQU *  BRANCH TO CKRTN63.
LA  R6,L'TAB(R6)  INCREMENT REG 6 TO NEXT POSITIONS.
CLI  Ø(R6),X'FF'  ARE WE DONE.
BNE CKRTN6  NO-BRANCH TO CKRTN6.

CKRTN7 EQU *
L R5,SVR5  RESTORE CONTENTS OF REG 5.
MVC Ø(L'INPFLD2,R5),INPFLD2

CKRTN9 EQU *
CLI WAISW,C'1'  WAIT DEADLOCK CHECK.
BE CKRTNA3  YES-BRANCH TO CKRTNA3.
CLI INPFUNC,C'1'  WAS CHECK SPECIFIED.
BE CKRTNW  YES-BRANCH TO CKRTNW.
CLI INPFUNC,C'C'  WAS CHECK SPECIFIED.
BE CKRTNW  YES-BRANCH TO CKRTNW.
CLI INPRCDE,C'Ø'  ANYTHING TO WAIT FOR.
BE CKRTNW  NO-BRANCH TO CKRTNW.
CLI INPRCDE,C'3'  WERE THERE ANY ERRORS.
BH CKRTNW  YES-BRANCH TO CKRTNW.

CKRTNA EQU *
MVC INPFLD1S(L'INPFLD1),INPFLD1 SVE CHECK FIELDS.
MVC OPTN123S,OPTN1 SVE OPTION FIELDS.
MVC INPJOBN,INPFLD1S+1 MVE EXECUTION JOB NAME TO INPJOBN.
MVC INPPGMN,INPFLD1S+9 MVE EXECUTION PROGRAM NAME TO INPPGM
MVI OPTN1,C' '  INDICATE NO BYPASS OF EXECUTION OR O
MVI OPTN2,C'Y'  INDICATE RETURN IN PRTY ORDER.
MVI OPTN3,C' '  INDICATE NO EX/INCLUDE PARTITION-ID'
MVI WAISW,C'1'  INDICATE WAIT DEADLOCK CHECK.
B CKRTNQ  BRANCH TO CKRTNQ.

CKRTNA3 EQU *
MVI WAISW,C'Ø'  INDICATE NO WAIT DEADLOCK CHECK.
MVC OPTN1(L'OPTN123S),OPTN123S RESTORE OPTION FIELDS.
CLI INPPCNT+1,X'Ø2'  ARE WE DEADLOCKED. (WAITING ON OURSE
BL CKRTNG  NO-BRANCH TO CKRTNG.
LA RB,INPCOMR  LOAD ADDRESS OF PARTITION COMMUNICAT

CKRTND EQU *
L RC,Ø(RB)  LOAD PARTITION COMMUNICATIONS ADDRES
CLI 12(RC),X'Ø2'  IS PARTITION RUNNING.
BE CKRTNE  YES-BRANCH TO CKRTNE.
CLI 12(RC),X'Ø1'  IS PARTITION WAITING.
BNE CKRTNF  NO-BRANCH TO CKRTNF.
LA RB,4(RB)  INCREMENT REG 11 TO NEXT POSITIONS.
CLC =F'Ø','.0(RB)  ARE WE DONE.
BNE CKRTND  NO-BRANCH TO CKRTND.
S RB,=F'4'  BACKUP FOUR (4) BYTES.
* BCTR RB,RØ  REDUCE REG 11 BY ONE (1).
* BCTR RB,RØ  ...
* CLI ESASW,C'Ø'  ARE WE RUNNING UNDER VSE/ESA.
* BNE CKRTND3  NO-BRANCH TO CKRTND3.
* BCTR RB,RØ  ...
* BCTR RB,RØ  ...
* RTND3 EQU *
L RC,Ø(RB)  LOAD PARTITION COMMUNICATIONS ADDRES
CLC =F'Ø',13(RC)  DOES SOMEBODY ALREADY OWN IT.
BNE CKRTNE YES-BRANCH TO CKRTNE.
BAL RA,CKSKEY PERFORM CKSKEY ROUTINE.
MVI 12(RC),X'02' INDICATE PARTITION RUNNING.
MVC 13(4,RC),0(RB) MVE OWNERS COMRG.
BAL RA,CKRKEY PERFORM CKRKEY ROUTINE.

CKRTNE EQU *
  L RA,13(RC) LOAD SAVED PARTITION COMMUNICATIONS
  C RA,ACOMRG IS IT MINE.
  BE CKRTNW YES-BRANCH TO CKRTNW.
  B CKRTNG BRANCH TO CKRTNG.

CKRTNF EQU *
  BAL RA,CKSKEY PERFORM CKSKEY ROUTINE.
  MVI 12(RC),X'01' INDICATE PARTITION WAITING.
  BAL RA,CKRKEY PERFORM CKRKEY ROUTINE.
  B CKRTND BRANCH TO CKRTND.

CKRTNG EQU *
  MVC INPFLD1,INPFLD1S RESTORE CHECK FIELDS.
  CLC INPFLD1S+L'INPFLD1(L'INPPIDS),INPPIDS DO PARTITION-ID'S
  BE CKRTNO YES-BRANCH TO CKRTNO.
  MVE OWNERS COMRG.
  MVC INPFLD1S+L'INPFLD1(L'INPPIDS),INPPIDS SVE CHECK PARTITION-ID'S
  MVC CNWK,BLANKS CLEAR CONSOLE WORK AREA.
  * MVC CNWK(3),CKJPMS MVE 'JOB' LITERAL TO CONSOLE WORK AREA.
  * LA RC,CNWK+4 LOAD ADDRESS OF CNWK+4 TO REG 12.
  * MVC 0(L'JOBNAME,RC),JOBNAME MVE JOB NAME TO CONSOLE WORK AREA.
  * LA RC,L'JOBNAME(RC) LOAD LENGTH OF JOBNAME TO REG 12.
  * BAL RA,CKSHFT PERFORM CKSHFT ROUTINE.
  * BCTR RC,R0 REDUCE REG 12 BY ONE (1).
  * MVC 0(11,RC),=C'WAITING ON' MVE 'WAITING ON' LITERAL TO CONSOLE WORK AREA.
  * LA RC,12(RC) INCREMENT REG 12 TO NEXT POSITION.
  LA RC,CNWK LOAD ADDRESS OF CNWK TO REG 12.
  MVC 0(10,RC),=C'WAITING ON' MVE 'WAITING ON' LITERAL TO CONSOLE WORK AREA.
  LA RC,11(RC) INCREMENT REG 12 TO NEXT POSITION.
  LA RB,INPPIDS LOAD ADDRESS OF PARTITION-IDS TO REG 12.
  CLI INPJOBN,C' ' WAS JOB NAME SPECIFIED.
  BE CKRTNH NO-BRANCH TO CKRTNH.
  CLI INPPGMN,C' ' WAS PROGRAM NAME SPECIFIED.
  BE CKRTNH NO-BRANCH TO CKRTNH.
  MVC 0(L'CKJPMS,RC),CKJPMS MVE 'JOB/PROGRAM' LITERAL TO REG 12.
  MVC 12(L'INPJOBN,RC),INPJOBN MVE JOB NAME TO REG 12.
  LA RC,L'CKJPMS+L'INPJOBN(RC) INCREMENT REG 12 TO NEXT POSITION.
  BAL RA,CKSHFT PERFORM CKSHFT ROUTINE.
  BCTR RC,R0 REDUCE REG 12 BY ONE (1).
  MVI (RC),'/ ' MVE SLASH (/) TO REG 12.
  MVC 1(L'INPPGMN,RC),INPPGMN MVE PROGRAM NAME TO REG 12.
  LA RC,L'INPPGMN+1(RC) INCREMENT REG 12 TO NEXT POSITIONS.
  BAL RA,CKSHFT PERFORM CKSHFT ROUTINE.
  B CKRTNJ BRANCH TO CKRTNJ.

CKRTNH EQU *
  CLI INPJOBN,C' ' WAS PROGRAM NAME ONLY SPECIFIED.
  BE CKRTNI YES-BRANCH TO CKRTNI.
MVC Ø(3,RC),CKJPMS MVE 'JOB' LITERAL TO REG 12.
MVC 4(L'INPJOBN,RC),INPJOBN MVE JOB NAME TO REG 12.
LA RC,L'INPJOBN+4(RC) INCREMENT REG 12 TO NEXT POSITIONS.
BAL RA,CKSHFT PERFORM CKSHFT ROUTINE.
B CKRTNJ BRANCH TO CKRTNJ.

CKRTNI EQU *

MVC Ø(7,RC),CKJPMS+4 MVE 'PROGRAM' LITERAL TO REG 12.
MVC Ø(L'INPPGMN,RC),INPPGMN MVE PROGRAM NAME TO REG 12.
LA RC,L'INPJOBN+8(RC) INCREMENT REG 12 TO NEXT POSITIONS.

CKRTNJ EQU *

MVC Ø(2,RC),=C'IN' MVE 'IN' LITERAL TO REG 12.
LA RC,3(RC) INCREMENT REG 12 TO NEXT POSITIONS.

CKRTNK EQU *

MVC Ø(2,RC),Ø(RB) MVE PARTITION-ID TO CONSOLE WORK ARE
MVI 2(RC),C',,' MVE COMMA (,).
LA RB,2(RB) INCREMENT REG 11 TO NEXT POSITIONS.
LA RC,3(RC) INCREMENT REG 12 TO NEXT POSITIONS.
CLI Ø(RB),C,' ARE WE DONE.
BNE CKRTNK NO-BRANCH TO CKRTNK.
BCTR RC,RØ REDUCE REG 12 BY ONE (1).

MVC Ø(RC),C,'.' MVE PERIOD (.)
BAL RA,CKCPUT PERFORM CKCPUT ROUTINE.

CKRTNO EQU *

ST RB,SVRB1 SVE CONTENTS OF REG 11.
LA RB,3Ø SET SECONDS TO 3Ø.
BAL RA,CKSTIM PERFORM CKSTIM ROUTINE.
L RB,SVRB1 RESTORE CONTENTS OF REG 11.

CKRTNQ EQU *

CLI NUMPRM,X'Ø2' WERE TWO (2) PARAMETERS PASSED.
BNE CKSTR NO-BRANCH TO CKSTR.
L R5,SVR5 RESTORE CONTENTS OF REG 5.
B CKLST3 BRANCH TO CKLST3.

CKRTNW EQU *

CLI INPFUNC,C'1' WAS CHECK SPECIFIED.
BE CKRTNZ YES-BRANCH TO CKRTNZ.
CLI INPFUNC,C'C' WAS CHECK SPECIFIED.
BE CKRTNZ YES-BRANCH TO CKRTNZ.
MVI INPRCDE,C'Ø' INDICATE NO JOB/PROGRAM FOUND.
MVC INPOPTN,OPTN3 RESTORE OPTION BYTE.
MVC INPPIDS,BLANKS CLEAR PARTITION-ID'S.
CLI NUMPRM,X'Ø2' WERE TWO (2) PARAMETERS PASSED.
BNE CKRTNZ NO-BRANCH TO CKRTNZ.
L R5,SVR5 RESTORE CONTENTS OF REG 5.
MVC INPBPG,BLANKS CLEAR JOB/PROGRAM FIELD.
XC INPCOMR(48),INPCOMR CLEAR PARTITION COMMUNICATIONS ADDRESS.
MVC Ø(L'INPFDLD2,R5),INPFDLD2

CKRTNZ EQU *

MVC INPFLD1S,BLANKS CLEAR PARAMETER 1 SAVE AREA.
MVC Ø(L'INPFDLD1,R4),INPFDL1
* PDUMP DPCKJPS,DPCKJPM
L R6,SAVEAREA+4
RETURN (14,12)

CKMVE EQU *
USING COMREG,RA
ST R6,SVR6B SVE CONTENTS OF REG 6.
LA R6,TAB LOAD ADDRESS OF TAB TO REG 6.

CKMVE5 EQU *
CLC PIBLOGID,PIDS IS THIS PIK OF MATCHED BG PARTITION.
BNE CKMVE53 NO-BRANCH TO CKMVE53.
CLC 3(L'PIBLOGID,R6),=X'ØØ1Ø' IS THIS PIK OF MATCHED PARTITION.
BE CKMVE7 YES-BRANCH TO CKMVE7.

CKMVE53 EQU *
CLC 3(L'PIBLOGID,R6),PID IS THIS PIK OF MATCHED PART (X'2E')
BE CKMVE7 YES-BRANCH TO CKMVE7.
LA R6,L'TAB(R6) INCREMENT REG 6 TO NEXT POSITIONS.
CLI Ø(R6),X'FF' ARE WE AT THE END OF THE TABLE.
BNE CKMVE5 NO-BRANCH TO CKMVE5.
DC X'ØØØØ'

CKMVE6 EQU *
ST R6,SVR6B SVE CONTENTS OF REG 6.
LA R6,TAB LOAD ADDRESS OF TAB TO REG 6.

CKMVE63 EQU *
CLI 1(R6),C' ' IS THIS AN OPEN SLOT.
BE CKMVE7 YES-BRANCH TO CKMVE7.
LA R6,L'TAB(R6) INCREMENT TO NEXT POSITION.
CLI Ø(R6),X'FF' ARE WE AT THE END OF THE TABLE.
BNE CKMVE63 NO-BRANCH TO CKMVE63.
B CKSTR33 YES-BRANCH TO CKSTR33.

CKMVE7 EQU *
MVC 1(L'PIBLOGID,R6),PIBLOGID MVE PARTITION-ID TO REG 6.
MVC 5(L'INPJOBN,R6),COMNAME MVE PARTITION JOB NAME T (X'18')
MVC 13(L'INPPGMN,R6),IJBPHNAM MVE PARTITION PROGRAM N (X'D8')
MVC 21(L'PIBLOGID+2,R6),SVRA MVE PARTITION COMMUNICATIONS AD
CKMVE9 EQU *
LH RF,COUNT INSERT PARTITION COUNTER TO REG 15.
LA RF,1(RF) INCREMENT REG 15 TO NEXT POSITION.
STH RF,COUNT STORE IT BACK.
CLC COUNT,=H'12' HAVE WE EXCEEDED MAX TABLE SIZE.
BH CKSTR33 YES-BRANCH TO CKSTR33.
MVC INPPCNT,COUNT MVE IT TO COUNT.
L R6,SVR6B RESTORE CONTENTS OF REG 6.
CLI NUMPRM,X'Ø2' WERE TWO (2) PARAMETERS PASSED.
BNER RB NO-RETURN TO CALLER.
ST R5,SVR5C SVE CONTENTS OF REG 5.
L R5,SVR5A RESTORE CONTENTS OF REG 5.
MVC Ø(L'INPJOBN,R5),COMNAME MVE PARTITION JOB NAME T (X'18')
MVC 8(L'INPPGMN,R5),IJBPHNAM MVE PARTITION PROGRAM N (X'D8')
LA R5,L'INPJOBN+L'INPPGMN(R5) INCREMENT REG 5 TO NEXT POSITION.
ST R5,SVR5A SVE CONTENTS OF REG 5.
L     R5,SVR5B            RESTORE CONTENTS OF REG 5.
MVC   Ø(L'PIBLOGID+2,R5),SVRA MVE PARTITION COMMUNICATIONS ADD
LA    R5,L'PIBLOGID+2(R5) INCREMENT REG 5 TO NEXT POSITIONS.
ST    R5,SVR5B            SVE CONTENTS OF REG 5.
L     R5,SVR5C            RESTORE CONTENTS OF REG 5.
CLI   INPFUNCS,C'G'  
BE    CKMVE96          
CLI   INPFUNCS,C'P'  
BE    CKMVE97          
BR    RB                  RETURN TO CALLER.

CKMVE96  EQU   *             
CLI   COMUSCR+3,X'00'  
BER   RB            
L     R1,=X'FFØØØØØØ'  SET ENABLE STORAGE PROT KEY.
SVC   13            GO DO IT.
MVC   INPPIDS(8),COMUSCR+3  
MVC   COMUSCR+3(8),=8X'00'  
L     R1,=X'FFØØØØFF'  RESET ENABLE STORAGE PROT KEY.
SVC   12            GO DO IT.
MVC   COUNT,=H'12'  
BR    RB            

CKMVE97  EQU   *             
L     R1,=X'FFØØØØØØ'  SET ENABLE STORAGE PROT KEY.
SVC   13            GO DO IT.
MVC   COMUSCR+3(8),INPPIDSS 
L     R1,=X'FFØØØØFF'  RESET ENABLE STORAGE PROT KEY.
SVC   12            GO DO IT.
MVC   COUNT,=H'12'  
BR    RB            
DROP  RA                  (COMREG).

CKCPUT   EQU   *              CONSOLE PUT ROUTINE.
MVI   CCW,X'09'            SET CCW TO WRITE.
LA    R1,CCB              LOAD ADDRESS OF CCB.
EXCP  (R1)                EXECUTE IT.
WAIT  (R1)                WAIT FOR COMPLETION.
MVC   CNWK,CNWK-1         CLEAR CONSOLE WORK AREA.
BR    RA                  RETURN TO CALLER.

CKSTIM   EQU   *              SET TIMER AND WAIT SPECIFIED SECONDS ROUT
SETIME (RB),TIMOUT        SET TO NNN SECONDS.
WAIT  TIMOUT              WAIT TILL NNN SECONDS HAS ELAPSED.
BR    RA                  RETURN TO CALLER.

CKSKEY   EQU   *              SET STORAGE PROTECT KEY ROUTINE.
L     R1,=X'FFØØØØØØ'  SET ENABLE STORAGE PROT KEY.
SVC   13            GO DO IT.
BR    RA                  RETURN TO CALLER.

CKRKEY   EQU   *              RESET STORAGE PROTECT KEY ROUTINE.
L     R1,=X'FFØØØØFF'  RESET ENABLE STORAGE PROT KEY.
SVC   12            GO DO IT.
BR    RA                  RETURN TO CALLER.

CKSHFT   EQU   *              SHIFT LEFT CONSOLE MESSAGE ROUTINE.
CLI Ø(RC),C' ' IS THIS POSITION BLANK.
BNE CKSHFT3 NO-BRANCH TO CKSHFT3.
BCTR RC,RØ REDUCE REG 12 BY ONE (1).
B CKSHFT BRANCH TO CKSHFT.

CKSHFT3 EQU *
LA RC,2(RC) INCREMENT REG 12 TO NEXT POSITION.
BR RA RETURN TO CALLER.

CKDYN EQU *
ST RB,SVRB SVE CONTENTS OF REG 11.
L R1,=X'FF000000' SET ENABLE STORAGE PROT KEY.
SVC 13 GO DO IT.

CKDYN1 EQU *
LA R6,L'PCBSTAP(R6) INCREMENT REG 6 TO NEXT ENTRY.
CLI Ø(R6),X'FF' ARE WE AT THE END.
BE CKDYN9C YES-BRANCH TO CKDYN9C.
ICM RC,15,Ø(R6) INSERT PCBADR TO REG 12.
BZ CKDYN1 ZERO-BRANCH TO CKDYN1.

* CLC =X'01AC0000',Ø(RC) IS PARTITION STILL ACTIVE.
* BNE CKDYN1 NO-BRANCH TO CKDYN1.

LA R7,PCEPIB LOAD PIB ADDRESS TO REG 7. (X'5A').
LA R8,PCEPIB2 LOAD PIB2 ADDRESS TO REG 8. (X'7C').
L RA,PCECOMRA LOAD ADDRESS OF ACTIVE DYNAMI (X'190
DROP RC (PCBADR).
MVC SVPCBA,Ø(RC) SVE 100 BYTES.
USING PCBADR,RC MAP TO PCB/PCE.
L R7,PCEPIB LOAD PIB ADDRESS TO REG 7. (X'5A').
L R8,PCEPIB2 LOAD PIB2 ADDRESS TO REG 8. (X'7C').
L RA,PCECOMRA LOAD ADDRESS OF ACTIVE DYNAMI (X'190

BNE CKDYN13 NO-BRANCH TO CKDYN13.
CLC PART,2(R7) IS THIS THE PARTITION-ID WE'RE RUNN
BE CKDYN1 YES-BRANCH TO CKDYN1.

CKDYN13 EQU *
CLI OPTN1,C'Y' DO WE BYPASS PARTITION-ID WE'RE RUNN
BNE CKDYN13 NO-BRANCH TO CKDYN13.
CLC PART,2(R7) IS THIS THE PARTITION-ID WE'RE RUNN
BE CKDYN1 YES-BRANCH TO CKDYN1.

CKDYN15 EQU *
CLC Ø(2,RB),2(R7) DO WE EXCLUDE THIS PARTITION.
BE CKDYN1 YES-BRANCH TO CKDYN1.
LA RB,2(RB) INCREMENT TO NEXT PARTITION-ID.
BCT RF,CKDYN15 BRANCH TO CKDYN15 UNTIL REG 15 ZERO.
B CKDYN9 BRANCH TO CKDYN9.

CKDYN3 EQU *
CLI OPTN3,C'I' DO WE INCLUDE PARTITION-IDS.
BNE CKDYN9 NO-BRANCH TO CKDYN9.
LA RB,OPTN3+1 LOAD ADDRESS OF PARTITION-IDS TO RE
LA RF,12 LOAD BRANCH COUNTER TO REG 15.

CKDYN35 EQU *
CLC Ø(2,RB),2(R7) DO WE INCLUDE THIS PARTITION.
BE CKDYN9 YES-BRANCH TO CKDYN9.
LA RB,2(RB) INCREMENT TO NEXT PARTITION-ID.
BCT RF,CKDYN35 BRANCH TO CKDYN35 UNTIL REG 15 ZERO.
B CKDYN1 BRANCH TO CKDYN1.

CKDYN9 EQU *
L R1.=X'FF0000FF' RESET ENABLE STORAGE PROT KEY.
SVC 12 GO DO IT.
L RB,SVRB RESTORE CONTENTS OF REG 11.
BR RB RETURN TO CALLER.

CKDYN9C EQU *
L R1.=X'FF0000FF' RESET ENABLE STORAGE PROT KEY.
SVC 12 GO DO IT.
B CKRTN BRANCH TO CKRTN.

CKLOP EQU *
ST RB,SVRB SVE CONTENTS OF REG 11.

CKLOP1 EQU *
LA R6,1(R6) INCREMENT REG 6 BY ONE (1).
CR R6,R5 ARE WE ABOVE NPARTS VALUE.
BH CKRTN YES-BRANCH TO CKRTN.
LA R7,16(R7) INCREMENT REG 7 TO NEXT PIB TABLE EN
LA R8,16(R8) INCREMENT REG 8 TO NEXT PIB2 TABLE E
MVC PIBLOGID,2(R7) SVE SYSLOG ID.
CLI OPTN1,C'Y' DO WE BYPASS PARTITION-ID WE'RE RUNN
BNE CKLOP13 NO-BRANCH TO CKLOP13.
CLC PART,2(R7) IS THIS THE PARTITION-ID WE'RE RUNNI
BE CKLOP1 YES-BRANCH TO CKLOP1.

CKLOP13 EQU *
CLI OPTN3,C'E' DO WE EXCLUDE PARTITION-IDS.
BNE CKLOP3 NO-BRANCH TO CKLOP3.
LA RB,OPTN3+1 LOAD ADDRESS OF PARTITION-IDS TO RE
LA RF,12 LOAD BRANCH COUNTER TO REG 15.

CKLOP15 EQU *
CLC Ø(2,RB),2(R7) DO WE EXCLUDE THIS PARTITION.
BE CKLOP1 YES-BRANCH TO CKLOP1.
LA RB,2(RB) INCREMENT TO NEXT PARTITION-ID.
BCT RF,CKLOP15 BRANCH TO CKLOP15 UNTIL REG 15 ZERO.
B CKLOP1 BRANCH TO CKLOP1.

CKLOP3 EQU *
CLI OPTN3,C'I' DO WE INCLUDE PARTITION-IDS.
BNE CKLOP9 NO-BRANCH TO CKLOP9.
LA RB,OPTN3+1 LOAD ADDRESS OF PARTITION-IDS TO RE
LA RF,12 LOAD BRANCH COUNTER TO REG 15.

CKLOP35 EQU *
CLC Ø(2,RB),2(R7) DO WE INCLUDE THIS PARTITION.
BE CKLOP9 YES-BRANCH TO CKLOP9.
LA RB,2(RB) INCREMENT TO NEXT PARTITION-ID.
BCT RF,CKLOP35 BRANCH TO CKLOP35 UNTIL REG 15 ZERO.
B CKLOP1 BRANCH TO CKLOP1.

CKLOP9 EQU *
L RB,SVRB RESTORE CONTENTS OF REG 11.
BR RB RETURN TO CALLER.
LH RA,Ø(R8)   LOAD PARTITION COMMUNICATIONS ADDRESS
ST RA,SVRA   SVE IT.
USING COMREG,RA  INFORM ASSEMBLER.
LA RC,INPJOBN LOAD ADDRESS OF JOB NAME TO REG 12.
ST RC,SVRC1   SVE IT.
LA RD,COMNAME LOAD ADDRESS OF PARTITION JOB NAME TO REG 13.
ST RD,SVRD   SVE IT.
LA RE,INPJOBN+L'INPJOBN-1 LOAD BACK END OF JOB NAME TO REG 15.
BR RB   RETURN TO CALLER.
DROP RA (COMREG).
CKSPG EQU *
LH RA,Ø(R8)   LOAD PARTITION COMMUNICATIONS ADDRESS
ST RA,SVRA   SVE IT.
USING COMREG,RA  INFORM ASSEMBLER.
LA RC,INPPGMN LOAD ADDRESS OF PROGRAM NAME TO REG 12.
ST RC,SVRC1   SVE IT.
LA RD,IJBAFCB LOAD ADDRESS OF IJABFCB TO RE (X'B4'E).
ST RD,AIJABFCB SVE IT.
LA RD,IJBPHNAM LOAD ADDRESS OF PARTITION PR (X'D8'E).
ST RD,SVRD   SVE IT.
LA RE,INPPGMN+L'INPPGMN-1 LOAD BACK END OF PROGRAM NAME TO REG 15.
BR RB   RETURN TO CALLER.
DROP RA (COMREG).
CKMTC EQU *
MVI MTCSW,C'Ø'  INDICATE NO MATCH.
CLC =C'--CICS--',Ø(RC)  IS THIS SPECIAL PROGRAM NAME --CICS-. 
BNE CKMTCØ  NO-BRANCH TO CKMTCØ.
L RØ,AIJABFCB  LOAD ADDRESS OF IJABFCB TO REG 0.
LTR RØ,RØ   IS CICS RUNNING IN THIS PARTITION.
BNZ CKMTC9   YES-BRANCH TO CKMTC9.
BR RB   RETURN TO CALLER.
CKMTCØ  EQU *
CLI INPFUNC,C'1'  IS FUNCTION '1'. (SAME AS 'C').
BE CKMTCØC  YES-BRANCH TO CKMTCØC.
CLI INPFUNC,C'2'  IS FUNCTION '2'. (SAME AS 'W').
BNE CKMTCØE  NO-BRANCH TO CKMTCØE.
CKMTCØC  EQU *
BAL RE,MATCH  PERFORM MATCH ROUTINE.
BR RB   RETURN TO CALLER.
CKMTCØE  EQU *
LA RØ,8  LOAD LENGTH OF JOB/PROGRAM NAME TO REG 15.
CLC =C'*',Ø(RC)  DOES FIELD BEGIN WITH ASTERISK.
BE CKMTC9  YES-BRANCH TO CKMTC9.
CLC =C'+',Ø(RC)  DOES FIELD BEGIN WITH PLUS SIGN.
BE CKMTC9  YES-BRANCH TO CKMTC9.
CLI Ø(RC),C'*'  DOES FIELD BEGIN WITH ASTERISK.
BE CKMTC6  YES-BRANCH TO CKMTC6.
ST RC,SVRC2   SVE CONTENTS OF REG 12.
LR RF,RØ   LOAD LENGTH OF FIELD TO REG 15.
CKMTCØH EQU *
CLI Ø(RC),C'+'  // Does field contain a plus sign.
BE CKMTC2  // Yes-branch to CKMTC2.
LA RC,1(RC)  // Increment reg 12 to next position.
BCT RF,CKMTC0H  // Branch to CKMTC0H until reg 15 zero.
L RC,SVRC2  // Restore contents of reg 12.
LR RF,R0  // Load length of field to reg 15.

CKMTC1 EQU *  // Does field contain an asterisk.
CLI Ø(RC),C'*'  // Yes-branch to CKMTC1.
LA RC,1(RC)  // Increment reg 12 to next position.
BCT RF,CKMTC1  // Branch to CKMTC1 until reg 15 zero.
L RC,SVRC2  // Restore contents of reg 12.
B CKMTC8  // Branch to CKMTC8. (No * or +).

CKMTC1C EQU *  // Load begin pointer to reg 14.
L RE,SVRC2  // Load current pointer to reg 15.
SR RF,RE  // Calculate length.
L RC,SVRC2  // Restore begin pointer to reg 12.
BCTR RF,0  // Make it machine length.
EX RF,CKMTC6I  // Execute CLC at CKMTC6I.
BE CKMTC9  // Yes-branch to CKMTC9.
BR RB  // Return to caller.

CKMTC2 EQU *  // Does field contain an asterisk.
CLI Ø(RC),C'*'  // Yes-branch to CKMTC7. (Error).
BE CKMTC7  // Increment reg 12 to next position.
BCT RF,CKMTC2  // Branch to CKMTC2 until reg 15 zero.
L RC,SVRC2  // Restore contents of reg 12.

CKMTC3 EQU *  // Do we check this position.
CLI Ø(RC),C'+'  // Yes-branch to CKMTC5.
BNE CKMTC5  // Are we done.

CKMTC4 EQU *  // Increment reg 12 to next position.
LA RC,1(RC)  // Are we done.
CR RC,RE  // Yes-return to caller.
BHR RB  // Increment reg 13 to next position.

CKMTC5 EQU *  // Are we done.
CLI Ø(RC),C''  // Yes-return to caller.
BER RB  // Does this position match.
CLC Ø(1,RC),Ø(RD)  // Yes-skip next two (2) inst.
BE *+10  // Indicate no match.
MVI MTCSW,C'0'  // Return to caller.
BR RB  // Indicate match.
MVI MTCSW,C'1'  // Branch to CKMTC4.
B CKMTC4

CKMTC6 EQU *  // Increment reg 12 past asterisk.
LA RC,1(RC)  // Is position after asterisk blank.
CLI Ø(RC),C''  // Yes-return to caller.
ST    RC,SVRC2      SVE CONTENTS OF REG 12.

CKMTC6B EQU  *  
CLI  Ø(RC),C' '  ARE WE AT THE END OF FIELD.  
BE  CKMTC6C  YES-BRANCH TO CKMTC6C.  
CLI  Ø(RC),C''  ARE WE AT THE END OF FIELD.  
BE  CKMTC6C  YES-BRANCH TO CKMTC6C.  
CLI  Ø(RC),C'+'  DOES FIELD CONTAIN A PLUS SIGN.  
BE  CKMTC7  YES-BRANCH TO CKMTC7. (ERROR).  
LA  RC,1(RC)  INCREMENT REG 12 TO NEXT POSITION.  
CR  RC,RE  ARE WE AT THE END OF FIELD.  
BNH  CKMTC6B  NO-BRANCH TO CKMTC6B.  

CKMTC6C EQU  *  
L  RE,SVRC2  LOAD BEGIN POINTER TO REG 14.  
LR  RF,RC  LOAD CURRENT POINTER TO REG 15.  
SR  RF,RE  CALCULATE LENGTH.  
BCTR  RF,Ø  MAKE IT MACHINE LENGTH.  
L  RC,SVRC2  RESTORE BEGIN POINTER TO REG 12.  

CKMTC6G EQU  *  
EX  RF,CKMTC6I  EXECUTE CLC AT CKMTC6I.  
BE  CKMTC9  YES-BRANCH TO CKMTC9.  
LA  RD,1(RD)  INCREMENT REG 13 TO NEXT POSITION.  
BCT  RØ,CKMTC6G  BRANCH TO CKMTC6G UNTIL REG Ø ZERO.  
BR  RB  RETURN TO CALLER. (NO MATCH).  

CKMTC6I EQU  *  
CLC  Ø(1,RC),Ø(RD)  DO WE HAVE A MATCH.  

CKMTC7 EQU  *  
MVI  MTCSW,C'Ø'  INDICATE NO MATCH.  
MVI  INPRCDE,C'4'  INDICATE */+ OR +/* ERROR.  
B  CKRTN9  BRANCH TO CKRTN9.  

CKMTC8 EQU  *  
LR  RF,RØ  LOAD LENGTH OF FIELD TO REG 15.  
BCTR  RF,RØ  MAKE IT MACHINE LENGTH.  
EX  RF,CKMTC6I  EXECUTE CLC AT CKMTC6I.  
BNER  RB  NO-RETURN TO CALLER.  

Editor’s note: this article will be concluded next month with the publication of the remaining code.  

Robert Botsis  
Senior Systems Programmer (USA)  
© Xephon 1998
Converting macros to define statements

THE PROBLEM
The latest versions of CICS do not provide macro resource definitions for defining transaction (PCT) and program (PPT) entries, and VSAM file (FCT) entries must be assembled and then migrated to the CICS System Definition (CSD) file. However, some application vendors still supply macro source for defining these resources.

Implementing such definitions requires one or more of the following:

- Assembling the definitions using a qualifying version of CICS, assuming, of course, that such a version is available. After such an assembly, the entries may be migrated using the prior versions DFHCSDUP migration facility.
- Manual conversion of definitions to equivalent CSD DEFINE statements.
- Manual entry of equivalent definitions with transaction CEDA (RDO facility).
- Allowing programs to be defined with the auto-install facility.

A SOLUTION
Create replacement macros (DFHPCT, DFHPPT, and DFHFCT) that process the obsolete definitions and build CSD DEFINE statements.

The Assembler is used to process the source definitions and produce the CSD DEFINE statements on the SYSPUNCH file.

The GROUP definition (of the CSD DEFINE statement) is defined by the global character set (GBLC) symbols &GROUPC, &GROUPP, and &GROUPF for DFHPCT, DFHPPT, and DFHFCT macros, respectively. These symbols may be defined as follows:

- Manual placement of Assembler SETC statements within the source definitions.
• Using the Assembler SYSPARM parameter.

• Accepting the default values of ‘PCTXXss’, ‘PPTXXss’, and ‘FCTXXss’ for the respective macros, where ‘ss’ is the table suffix as defined in the INITIAL macro SUFFIX keyword parameter.

Special features include:

• Duplicate ENTRY identifiers, for each resource, are eliminated. A comment is inserted into the output to note this deletion.

• A global set symbol (&DESCR) permits the insertion of text into the DESCRIPTION() field of the DEFINE statements. If this information is not provided, a default description is constructed from the above GROUP names. The global definition may be inserted for individual entries or groups of entries.

• If conversion of PCT entries requires PROFILE definitions to supplement the TRANSACTION definitions, these PROFILE definitions are constructed. Comments are inserted into the output source to indicate the parameters that caused the generation. Also, an attempt is made to eliminate any redundant definitions of such PROFILE definitions. These definitions use an entry name created from the above GROUP names and a sequential number.

MACRO SOURCE

The following macros may be inserted into a macro library normally used in assemblies (eg CICSxxx.SDFHMAC), or concatenated into the assembly, as shown in the sample JCL below. If the first option is used, care should be taken to ensure that these macros will not be used when assembling older versions of CICS tables.

DFHFCT MACRO

MACRO

DFHFCT &TYPE=, TYPE OF MACRO *
  &ACCMETH=, ACCESS METHOD IDENTIFICATION *
  &BASE=, BASE SYMBOL FOR BSTRNO TABULATION +
  &BLKKEYL=, PHYSICAL KEY LENGTH (DEFAULT = Ø) *
&BUFNI=,            VSAM INDEX BUFFER NUMBER            *
&BUFND=,            VSAM DATA BUFFER NUMBER            *
&BLKSIZE=,          BLOCK SIZE                          *
&BUFFERS=,          BUFFERS FOR VSAM POOL               *
&BUFSP=,            VSAM BUFFER SPACE                   *
&DATASET=,          NAME OF CICS FILE (SAME AS DDNAME)  *
&FILE=,             NAME OF CICS FILE (SAME AS DDNAME)  *
&FILSTAT=,          FILE STATUS                         *
&GROUP=,            RDO GROUP NAME                      *
&EXTENT=,           NUMBER OF DISK EXTENTS              *
&LRECL=,            LOGICAL RECORD LENGTH               *
&RKP=,              RELATIVE KEY POSITION               *
&KEYLEN=,           KEY LENGTH OF LOGICAL RECORD         *
&RELTYPE=,          TYPE OF RELATIVE RECORD ADDR         *
&VERIFY=,           WRITE VERIFY OPTION                 *
&SRCHM=,            MULTIPLE TRACK SEARCH - KEY         *
&JID=,              JOURNAL IDENTIFICATION              *
&JREQ=,             JOURNAL REQUESTS                    *
&LOG=,              SYSTEM LOG INDICATOR                 *
&MIGRATE=,          RESOURCE DEFINITION ONLINE CALL      *
&OPEN=,             OLD DEFERRED OPEN OPTION              *
&PASSWD=,           VSAM PASSWORD                       *
&RECFORM=,          RECORD FORMAT                       *
&RMTNAME=,          DATASET NAME ON REMOTE SYSTEM        *
&RSCLM=,            RESOURCE PERCENT FOR VSAM POOL      *
&RSL=,              RESOURCE LEVEL SECURITY              *
&SIZE=,             DATA TABLE SIZE                      *
&STRNO=,            VSAM MAXIMUM STRINGS                 *
&STRNOG=,           CICS 'GET ONLY' STRINGS (OS ONLY)     *
&SERVREQ=,          SERVICE REQUEST IDENTIFICATION      *
&LSRPPOOL=,         VSAM RESOURCE-SHARING SPECIFICATION   *
&SUFFIX=,           FILE CONTROL TABLE NAME SUFFIX       *
&STARTER=,          PREGENERATED TABLES ONLY             *
&SYSIDNT=,          REMOTE SYSTEM IDENTIFIER             *
&DSNAME=,           DATA SET NAME FOR DYNAMIC ALLOCATION*
&DSNSHR=,           DOES DSN-SHARING AFFECT R/O ACCESS?  *
&DISP=,             DISPOSITION FOR DATA SET             *
&DUMMY4=,           *                                     *
&DUMMY3=,           *                                     *
&DUMMY2=,           *                                     *
&DUMMY1=,           *                                     *
&DUMMY=             PROTOTYPE DUMMY PARAMETER@15553 @LBC

* GBLC &GROUPF,&SUFXF
  LCLA &I,&J,&K
  LCLC &X,&RDO(99),&P,&ID
  GBLC &IDS(500),&CMTS(500),&DESCR
  GBLA &IDN

* AIF ('&TYPE' NE 'INITIAL').NOINIT

AIF (T'&SUFFIX EQ 'O').NOSUF
&SUFXF SETC '&SUFFIX'
.NOSUFX AIF ('&GROUPF' NE '').NOINIT
AIF (T'&GROUP EQ 'O').NOINIT
&GROUPF SETC '&GROUP'

.NOINIT AIF ('&TYPE' NE 'DATASET' AND '&TYPE' NE 'FILE').END

AIF ('&ACCMETH' EQ 'VSAM').VSAM
MNOTE 4,'ONLY VSAM FILES ARE ELIGIBLE FOR RDO'
AGO .END

.VSAM ANOP
&I SETA $0
AIF (&IDN EQ 0).FIRSTID

.IDLOOP ANOP

&ID SETC 'FILE=&FILE'
AIF (T'&FILE NE 'O').IDED
&ID SETC 'DATASET=&DATASET'
.IDED ANOP

&I SETA &I+1
AIF ('&ID' NE '&IDS(&I)').NOTID
PUNCH '*&ID IS DUPLICATED ABOVE, SEE &CMTS(&I)'
PUNCH '*'
AGO .NOMNT

.NOTID AIF (&I LT &IDN).IDLOOP

.FIRSTID ANOP

AIF ('&GROUPF' NE '').GROUP
AIF ('&SYSPARM' EQ '').NOSPARM
&GROUPF SETC '&SYSPARM'
AGO .GROUP
&GROUPF SETC '&GROUP'
AGO .GROUP

NOSPARM ANOP
&GROUPF SETC 'FCTXX&SUFXF'

.GROUP ANOP

AIF (T'&FILE NE 'O').FILE
AIF (T'&DATASET NE 'O').DATASET

MNOTE 4,'NO FILE= OR DATASET='
ANO END

*
.FILE ANOP
&X SETC '&FILE'
AGO .NAME
.*
.DATASET ANOP
&X SETC '&DATASET'
.*
.NAME ANOP
PUNCH 'DEFINE FILE(&X) GROUP(&GROUPF)'
.*
AIF ('&DESCR NE '').DESCRX
&DESCR SETC 'PPT GROUP=&GROUPC'
.DESCRX ANOP
.*
&IDN SETA &IDN+1
&IDS(&IDN) SETC '&ID'
&CMTS(&IDN) SETC '&DESCR'
.*
&I SETA 0
&J SETA 1
.*
AIF (T'&LSRPOOL EQ 'O').NOLSR
&I SETA &I+1
&RDO(&I) SETC 'LSRPOOLID(&LSRPOOL)'
.*
.NOLSR AIF (T'&BUFND EQ 'O').NOBUFND
&I SETA &I+1
&RDO(&I) SETC 'DATABUFFERS(&BUFND)'
.*
.NOBUFND AIF (T'&BUFNI EQ 'O').NOBUFNI
&I SETA &I+1
&RDO(&I) SETC 'INDEXBUFFERS(&BUFNI)'
.*
.NOBUFNI AIF (T'&DSNSHR EQ 'O').NODSNSH
&I SETA &I+1
&RDO(&I) SETC 'DSNSHARING(MODIFREQS)'
AGO .NODSNSH
.*
.DSNSHA ANOP
&RDO(&I) SETC 'DSNSHARING(ALLREQS)'
.*
.NODSNSH AIF (T'&PASSWD EQ 'O').NOPASS
&I SETA &I+1
&RDO(&I) SETC 'PASSWORD(&PASSWD)'
.*
.NOPASS AIF (T'&STRNO EQ 'O').NOSTRNO
&I SETA &I+1
&RDO(&I) SETC 'STRINGS(&STRNO)'
.*

.NOFSTAT AIF (T'&JID' EQ '0').NOJID
AIF ('&JID' EQ 'NO').JIDNO
AIF ('&JID' EQ 'SYSTEM').JID1

*I
&I SETA &I+1
&RDO(&I) SETC 'JOURNAL(&JID)'
AGO .NOJID

.*
.JID1 ANOP
&I SETA &I+1
&RDO(&I) SETC 'JOURNAL(1)'
AGO .NOJID

.*
.JIDNO ANOP
&I SETA &I+1
&RDO(&I) SETC 'JOURNAL(NO)'

.*
.NOJID AIF (T'&LOG' EQ '0').NOLOG
AIF ('&LOG' EQ 'NO').LOGNO

.*
&I SETA &I+1
&RDO(&I) SETC 'RECOVERY(BACKOUTONLY)'
AGO .NOLOG

.*
.LOGNO ANOP
&I SETA &I+1
&RDO(&I) SETC 'RECOVERY(NONE)'

.*
.NOLOG AIF (T'&JREQ' EQ '0').NOJREQ
&K SETA 1
.JRLOOP AIF ('&JREQ(&K)' EQ 'WN').JRWN
AIF ('&JREQ(&K)' EQ 'RU').JRRU
AIF ('&JREQ(&K)' EQ 'RO').JRRO
AIF ('&JREQ(&K)' EQ 'SYN').JRSYN
AIF ('&JREQ(&K)' EQ 'ASY').JRASY
AIF ('&JREQ(&K)' NE 'WU').JREND

.*
&I SETA &I+1
&RDO(&I) SETC 'JNLUPDATE(YES)'
AGO .JREND

.*
.JRWN ANOP
&I SETA &I+1
&RDO(&I) SETC 'JNLADD(BEFORE)'
AGO .JREND

.*
.JRRU ANOP
&I SETA &I+1
&RDO(&I) SETC 'JNLREAD(UPDATEONLY)'
AGO .JREND

.*
JRRO ANOP
&I SETA &I+1
&RDO(&I) SETC 'JRLNREAD(READONLY)' AGO .JREND
.*
JRISYN ANOP
&I SETA &I+1
&RDO(&I) SETC 'JNLSYNCREAD(YES)' AGO .JREND
.*
JRASY ANOP
&I SETA &I+1
&RDO(&I) SETC 'JNLSYNCWRITE(NO)' .*
JREND ANOP
&K SETA &K+1
AIF (&K LE N'&JREQ).JRLOOP
.*
NOJREQ AIF (T'&RECFORM EQ 'O').NORECF
&K SETA 1
RFLOOP AIF ('&RECFORM(&K)' EQ 'VARIABLE').RFVAR
AIF ('&RECFORM(&K)' NE 'FIXED').RFEND
.*
&I SETA &I+1
&RDO(&I) SETC 'RECORDFORMAT(F)' AGO .RFEND
.*
RFVAR ANOP
&I SETA &I+1
&RDO(&I) SETC 'RECORDFORMAT(V)' .*
RFEND ANOP
&K SETA &K+1
AIF (&K LE N'&RECFORM).RFLOOP
.*
NORECF AIF (T'&RSL EQ 'O').NORSL
MNOTE 4,'RSL KEYWORD NOT SUPPORTED BY RDO'
.*
NORSL AIF (T'&SERVREQ EQ 'O').NOSVREQ
&K SETA 1
.*
SVLOOP AIF ('&SERVREQ(&K)' EQ 'ADD').SVADD
AIF ('&SERVREQ(&K)' EQ 'BROWSE').SVBROWS
AIF ('&SERVREQ(&K)' EQ 'DELETE').SVDELETE
AIF ('&SERVREQ(&K)' EQ 'READ').SVREAD
AIF ('&SERVREQ(&K)' NE 'UPDATE').SVEND
.*
&I SETA &I+1
&RDO(&I) SETC 'UPDATE(YES)'
AGO .SVEND

.*

.SVADD ANOP
&I SETA &I+1
&RDO(&I) SETC 'ADD(YES) ' 
AGO .SVEND

.*

.SVBROWS ANOP
&I SETA &I+1
&RDO(&I) SETC 'BROWSE(YES) ' 
AGO .SVEND

.*

.SVDELET ANOP
&I SETA &I+1
&RDO(&I) SETC 'DELETE(YES) ' 
AGO .SVEND

.*

.SVREAD ANOP
&I SETA &I+1
&RDO(&I) SETC 'READ(YES) ' 
AGO .SVEND

.*

.SVEND ANOP
&K SETA &K+1
AIF (&K LE N'&SERVREQ).SVLOOP

.*

.NOSVREQ AIF (T'&BASE EQ 'O').NOBASE
&I SETA &I+1
&RDO(&I) SETC 'NSRGROUP(&BASE) ' 
.*

.NOBASE AIF (T'&RMTNAME EQ 'O').NORMTN
&I SETA &I+1
&RDO(&I) SETC 'REMOTENAME(&RMTNAME) ' 
.*

.NORMTN AIF (T'&SYSIDNT EQ 'O').NOSYSID
&I SETA &I+1
&RDO(&I) SETC 'REMOTESYSTEM(&SYSIDNT) ' 
.*

AIF (T'&LRECL EQ 'O').NOLRECL
&I SETA &I+1
&RDO(&I) SETC 'RECORDSIZE(&LRECL) ' 
.*

.NOLRECL AIF (T'&KEYLEN EQ 'O').NOSYSID
&I SETA &I+1
&RDO(&I) SETC 'KEYLENGTH(&KEYLEN) ' 
.*

.NOSYSID ANOP
.*

.* KEYWORDS PROCESSED, PUNCH RDO DATA
.*

.BUILD ANOP
&X SETC ''
.*
.NEXT AIF (K'&X+K'&RDO(&J) LT 72).CONCAT
PUNCH '&X'
AGO .BUILD
.*
.CONCAT ANOP
&X SETC '&X&RDO(&J)'
&J SETA &J+1
.*
AIF (&J LE &I).NEXT
AIF (K'&X LE 6).DESCR
PUNCH '&X'
.*
.DESCRIPTION ANOP
.*
PUNCH ' DESCRIPTION(&DESCR) '
.*
&X SETC ''
.*
AIF (T'&BLKKEYL EQ 'O').NOBKL
&X SETC '&X'. 'BLKKEYL=&BLKKEYL'
.*
.NOBKL AIF (T'&BLKSIZE EQ 'O').NOBSK
&X SETC '&X'. 'BLKSIZE=&BLKSIZE'
.*
.NOBSK AIF (T'&BUFFERS EQ 'O').NOBSF
&X SETC '&X'. 'BUFFERS=&BUFFERS'
.*
.NOBSF AIF (T'&BUFSP EQ 'O').NOBUFSP
&X SETC '&X'. 'BUFSP=&BUFSP'
.*
.NOBUFSP AIF (T'&RKP EQ 'O').NORKP
&X SETC '&X'. 'RKP=&RKP'
.NOORKP AIF (T'&RELTYPE EQ 'O').NORELT
&X SETC '&X'. 'RELTYPE=&RELTYPE'
.*
.NORELT AIF (T'&VERIFY EQ 'O').NOVERIFY
&X SETC '&X'. 'VERIFY=&VERIFY'
.*
.NOVERIFY AIF (T'&SRCHM EQ 'O').NOSRCHM
&X SETC '&X'. 'SRCHM=&SRCHM'
.*
.NOSRCHM AIF (T'&OPEN EQ 'O').NOOPEN
&X SETC '&X'. 'OPEN=&OPEN'
.*
.NOOPEN AIF (T'&RSCLMT EQ 'O').NORSCLM
&X SETC '&X'. 'RSCLMT=&RSCLMT'
.*
.NORSCLM AIF (T'&SIZE EQ 'O').NOSIZE
&X SETC '&X'.'SIZE=&SIZE '
.
.NOSIZE AIF (T'&STRNOG EQ '0').NOSTRNG
&X SETC '&X'.'STRNOG=&STRNOG '
.
.NOSTRNG AIF (T'&STARTER EQ '0').NOSTRTR
&X SETC '&X'.'STARTER=&STARTER '
.
.NOSTRTR AIF (T'&DUMMY EQ '0').NODUMMY
&X SETC '&X'.'DUMMY=&DUMMY '
.
.NODUMMY AIF (T'&DUMMY1 EQ '0').NODUM1
&X SETC '&X'.'DUMMY1=&DUMMY1 '
.
.NODUM1 AIF (T'&DUMMY2 EQ '0').NODUM2
&X SETC '&X'.'DUMMY2=&DUMMY2 '
.
.NODUM2 AIF (T'&DUMMY3 EQ '0').NODUM3
&X SETC '&X'.'DUMMY3=&DUMMY3 '
.
.NODUM3 AIF (T'&DUMMY4 EQ '0').NODUM4
&X SETC '&X'.'DUMMY4=&DUMMY4 '
.
.NODUM4 AIF ('&X' EQ '').NOMNT
MNOTE 4,'THE FOLLOWING PARAMETERS WERE IGNORED &X'
.
.NOMNT ANOP
PUNCH '*'
.
.END MEND
GBLC &DESCR

DFHPCT MACRO

MACRO
&NAME DFHPCT &TYPE=, TYPE OF ENTRY *
&SUBSET=, REDUNDANT, DOS ONLY *
&CICS=, OBSOLETE PARAMETER *
&TRANSID=, TRANSACTION I.D. *
&TASKREQ=, 327Ø AID CHAR-TRAN ID *
&TXTRANID=, NON-LATIN-ALPHABETIC ALIAS *
&SCRNSZE=, SCREEN SIZE SELECTION *
&PTRCOMP=, 327Ø PRINTER COMPATIBILITY @07*
&SPURGE=, NO* STALL PURGE INDICATOR *
&TPURGE=, NO* TERM ERROR PURGE INDICATOR *
&DTB=, TASK TO BE BACKED OUT *
&COMPAT=, COMPATIBILITY OPTIONS *
&CLASS=, CLASS (NO LONGER SUPPORTED) *
&PRIVATE=, (NO LONGER SUPPORTED) ISOLATED TASK *
&TRNSTAT=, TRANSACTION STATUS *
&TRNPRTY=, TRANSACTION PRIORITY 7/22/92 KHN *
&TRANSEC=, TRANSACTION SECURITY KEY *
&WASIZE=, TRANSACTION WORK AREA SIZE *
&PRMSIZE=, PRIMED ALLOCATION SIZE *
&ISA=, INIT STORAGE ALLOCATION *
&SUFFIX=, P-C-T NAME SUFFIX *
&PROGRAM=, PROGRAM IDENTIFICATION *
&PROFILE=, PROFILE IDENTIFICATION *
&DVSUPRT=, TERML.DEVICE SUPPORT OPTION *
&RAQ=, READ AHEAD QUEUING REQUIRED *
&EXTRACT=, EXTRACT OPTIONS *
&MSGJRNL=, TERML.MSG.-JRNL.INPUT/OUTPUT *
&FILEID=, NO* TERML.MSG-AUTO.JOURNAL I.D. *
&TIOTYPE=, TERML.MSG-I/O PROCSS'G OPTN. *
&OPTGRP=, OPTION GROUP NAME *
&MSGPREG=, MSG.PROTECT-REQUIRED SPECIF. *
&MSGPOPT=, MSG.PROTECT-OPTIONAL SPECIF. *
&TCLASS=, TRANSACTION CLASS *
&PAGENXD=, PAGE INDEX *
&INDEX=, FULL INDEX OPTION *
&ANTICPG=, NO* ANTICIPATORY PAGING INDICATOR *
&RTOU=, TERMINAL READ TIME OUT *
&DTIMOUT=, DEAD-LOCK TIME OUT *
&RESTART=, NO* AUTO. TASK RESTART *
&DUMP=, YES* TRANSACTION DUMP REQUEST *
&NEPCLASS=, NODE ERROR PROGRAM CLASS *
&INBFMH=, PASS FMH TO APPL.PGM *
&LOGREC=, LOGICAL REC REQ *
&STARTER=, PREGENERATED TABLES ONLY *
&FN=, FUNCTIONS FOR SPECIAL XCTNS *
&KEYID=, KEY-DRIVEN XCTN WITHIN GROUP *
&SISDN=, REMOTE SYSTEM NAME *
&RMTNAME=, NAME ON REMOTE SYSTEM *
&LOCAL=, NO* LOCAL QUEUING AUTHORITY *
&EXTSEC=, EXTERNAL SECURITY PARM. *
&RSL=, RESOURCE SECURITY LEVEL *
&RSLC=, RSL CHECK REQUIRED *
&PARTSET=, PARTITION SET NAME *
&MODENAM=, MODE GROUP NAME *
&TRACE=, YES* TRACE OPTION *
&TRPROF=, TRANSACTION ROUTING PROFILE NAME *
&DUMMY=, DUMMY PARAMETER

* ABOVE * INDICATES DEFAULT VALUED REMOVED (DEFAULT PRECEDES *)
* THESE ARE ALSO THE CSD DEFAULTS AND WOULD ONLY CREATE REDUNDANT
* PARAMETERS. THESE (AND OTHERS) MAY BE MODIFIED FOR INDIVIDUAL
* PREFERENCES.

LCLA &I,&J,&PI,Pmax,&K
LCLC &X,&RDO(99),&P,&IS,&PF,&ID
GBLC &GROUPC,&SUFFIX
GBLA &NP
GBLC &DSUPC(50),&DTPC(50),&RTIMOC(50)
GBLC &SCRNSZC(50),&INBFMHC(50),&JFILEIC(50),&LOGRECC(50)
GBLC &MODENMC(50),&MSGJRNMC(50),&NEPC(50),&RAQC(50)
GBLC &PFID(50),&PFX(50),&PFDEF
GBLC &IDS(500),&CMTS(500),&DESCR
GBLA &IDN

&PMA    SETA 50 SET TO ABOVE GBLC ARRAY SIZE

.*
AIF ('&TYPE' NE 'INITIAL').NOINIT
AIF (T'SUFFIX EQ 'O').NOINIT
&SUFFIXC SETC '&SUFFIX'
.*
.NOINIT AIF ('&TYPE' NE 'ENTRY').END
.*
&I       SETA 0
AIF (&IDN EQ 0).FIRSTID
.*
.IDLOOP ANOP
.*
&I       SETC 'TRANSID=&TRANSID'
AIF (T'&TRANSID NE 'O').IDED
&I       SETC 'TASKREQ=&TASKREQ'
.IDED ANOP
.*
&I       SETA &I+1
AIF ('&ID' NE '&IDS(&I)').NOTID
PUNCH '*&ID IS DUPLICATED ABOVE, SEE &CMTS(&I)'
PUNCH '*'
AGO .NOMNT
.*
.NOTID AIF (&I LT &IDN).IDLOOP
.*
.FIRSTID ANOP
.*
AIF ('&GROUPC' NE '').GROUP
AIF ('&SYSPARM' EQ '').NOSPARM
&GROUPC SETC '&SYSPARM'
AGO .GROUP
.NOSPARM ANOP
&GROUPC SETC 'PCTXX&SUFFIX'
.*
.GROUP AIF (T'&TRANSID NE 'O').TRANSID
AIF (T'&TASKREQ NE 'O').TASKREQ
MNOTE 8,'NEITHER TRANSID NOR TASKREQ'
AGO .END
.*
.TRANSID ANOP
&RDO(1) SETC 'TRANSACTION(&TRANSID) '
AIF (T'&TASKREQ EQ 'O').DEFINE
.TASKREQ ANOP
&RDO(1) SETC '&RDO(1)TASKREQ(&TASKREQ) '
.DEFINE ANOP
PUNCH 'DEFINE &RDO(1)GROUP(&GROUPC) '
.*
AIF ('&DESCR NE ''').DESCRX
&DESCR SETC 'PPT GROUP=&GROUPC'
.DESCRX ANOP
.*
&IDN SETA &IDN+1
&IDS(&IDN) SETC '&ID'
&CMTS(&IDN) SETC '&DESCR'
.*
&PFX(&PMAX) SETC '&RDO(1)'
.*
&RDO(1) SETC 'PROGRAM(&PROGRAM) '
&I SETA 1
&J SETA 1
.*
AIF (T'&ANTICPG EQ 'O').NOANTIC
MNOTE 4,'ANTICIPATORY PAGING NOT SUPPORTED'
.*
.NOANTIC AIF (T'&CLASS EQ 'O').NOCLASS
MNOTE 4,'CLASS KEYWORD HAS BEEN OBSOLETE SINCE CICS 2.1'
.*
.NOCLASS AIF (T'&DTB EQ 'O').NODTB
&I SETA &I+1
AIF (N'&DTB EQ 2).DTB2
AIF ('&DTB' NE 'NO').DTBYES
MNOTE 4,'THE EQUIVALENT OF DTB=NO IS NOT SUPPORTED'
.DTBYES ANOP
&RDO(&I) SETC 'INDOUBT(BACKOUT) '
AGO .NODTB
.DTB2 AIF ('&DTB(1)' EQ 'WAIT' OR '&DTB(2)' EQ 'WAIT').DTBWAIT
&RDO(&I) SETC 'INDOUBT(COMMIT) '
AGO .NODTB
.DTBWAIT ANOP
&RDO(&I) SETC 'INDOUBT(WAIT) '
.*
.NODTB AIF (T'&DTMOUT EQ 'O').NODTIMO
&I SETA &I+1
&RDO(&I) SETC 'DTMOUT(&DTMOUT) '
.*
.NODTIMO AIF (T'&DUMP EQ 'O').NODUMP
&I SETA &I+1
&RDO(&I) SETC 'DUMP(&DUMP) '
.*
.NODUMP AIF (T'&EXTSEC EQ 'O').NOEXTS
 MNOTE 4,'EXTSEC KEYWORD IS NOT VALID FOR CICS 4.1'
.*
.NOEXTS AIF (T'&PARTSET EQ 'O').NOPSET
 &I SETA &I+1
 &RDO(&I) SETC 'PARTITIONSET(&PARTSET) '
.*
.NOPSET AIF (T'&RESTART EQ 'O').NORSTRT
 &I SETA &I+1
 &RDO(&I) SETC 'RESTART(&RESTART) '
.*
.NORSTRT AIF (T'&RSL EQ 'O').NORSL
 MNOTE 4,'RSL KEYWORD IS NOT VALID IN CICS 4.1'
.*
.NORSL AIF (T'&RSLC EQ 'O').NORSLC
 MNOTE 4,'RSLC KEYWORD IS NOT VALID IN CICS 4.1'
.*
.NORSLC AIF (T'&SPURGE EQ 'O').NOSPURG
 &I SETA &I+1
 &RDO(&I) SETC 'SPURGE(&SPURGE) '
.*
.NOSPURG AIF (T'&TCLASS EQ 'O').NOTCLAS
 MNOTE 4,'TCLASS IS AN OBSOLETE KEYWORD'
.*
.NOTCLAS AIF (T'&TPURGE EQ 'O').NOTPURG
 &I SETA &I+1
 &RDO(&I) SETC 'TPURGE(&TPURGE) '
.*
.NOTPURG AIF (T'&TRACE EQ 'O').NOTRACE
 &I SETA &I+1
 &RDO(&I) SETC 'TRACE(&TRACE) '
.*
.NOTRACE AIF (T'&TRANSEC EQ 'O').NOTRNSC
 MNOTE 4,'TRANSEC KEYWORD IS NOT VALID IN CICS 4.1'
.*
.NOTRNSC AIF (T'&TRNPRTY EQ 'O').NOTRNPR
 &I SETA &I+1
 &RDO(&I) SETC 'PRIORITY(&TRNPRTY) '
.*
.NOTRNPR AIF (T'&TRNSTAT EQ 'O').NOTSTAT
 &I SETA &I+1
 &RDO(&I) SETC 'STATUS(&TRNSTAT) '
.*
.NOTSTAT AIF (T'&TWASIZE EQ 'O').NOTWASZ
 &I SETA &I+1
 &RDO(&I) SETC 'TWASIZE(&TWASIZE) '
.*
.NOTWASZ AIF (T'&XTRANID EQ 'O').NOXTID
 &I SETA &I+1
 &RDO(&I) SETC 'XTRANID(&XTRANID) '

© 1998. Reproduction prohibited. Please inform Xephon of any infringement. 43
.*
.NOPTG AIF (T'&OPTGRP EQ 'O').NOOPTGR
 &I SETA &I+1
 &RDO(&I) SETC 'TRPROF(&TRPROF)'
.*
.NOTRPRF AIF (T'&LOCALQ EQ 'O').NOLQ
 &I SETA &I+1
 &RDO(&I) SETC 'LOCALQ(&LOCALQ)'
.*
.NOLQ ANOP
 &X SETC ''
 &IS SETC 'IS'
.*
.AIF (T'&DVSUPRT EQ 'O').NODSUP
 &X SETC '&X':'DVSUPR=&DVSUPRT'
 &DVSUPR(&PMAX) SETC '&DVSUPRT'
.*
.NODSUP AIF (T'&PTRCOMP EQ 'O').NOPTRC
 &I SETC 'ARE'
 &X SETC '&X':', '
 &IS ANOP
 &X SETC '&X':PTRCOMP=&PTRCOMP'
 &PTRC(&PMAX) SETC '&PTRC'
.*
.NOPTRC AIF (T'&RTIMOUT EQ 'O').NORTOUT
 &I SETC 'ARE'
 &X SETC '&X':', '
 &IS ANOP
 &X SETC '&X':RTIMOUT=&RTIMOUT'
 &RTIMO(&PMAX) SETC '&RTIMOUT'
.*
.NORTOUT AIF (T'&SCRNSZE EQ 'O').NOSCRNS
 &I SETC 'ARE'
 &X SETC '&X':', '
 &IS ANOP
 &X SETC '&X':SCRNSZE=&SCRNSZE'
 &SCRNZ(&PMAX) SETC '&SCRNSZE'
.*
.NOSCRNS AIF (T'&INBFMH EQ 'O').NOINBFM
 &I SETC 'ARE'
 &X SETC '&X':', '
 &IS ANOP
 &X SETC '&X':INBFMH=&INBFM
Editor’s note: this article will be continued next month.

Keith H Nicaise
Technical Services Manager
Touro Infirmary (USA)  © Xephon 1998
Screen viewing utility and extended attributes

A screen viewing utility was first introduced in CICS Update, Issue 105, August 1994. An update to the screen viewing utility was published in CICS Update, Issue 120, November 1995, which introduced the ability to view by user-id instead of by terminal-id. This utility is made up of three programs, beginning with PEEK, which is invoked with a parameter of ‘user-id’ or ‘terminal-id’, depending on which of the two versions of the utility you are referencing. Transaction LOOK is then started on the target terminal, and the screen contents are returned to program LOOK with the following CICS command within that program:

```
EXEC CICS RECEIVE INTO(...) LENGTH(...) BUFFER ASIS LEAVEKB
```

This returns a datastream containing Start Field (SF) attributes (X'1D'). If the terminal uses the extensions to the 3270 datastream, which include such features as underlining, reverse video, blinking, and setting of colours by field or character, then these extended fields cannot be returned in a datastream which simply returns Start Fields.

These extensions to the 3270 datastream are defined in the following ways:

- In the TCT definition for macro level – FEATURE=EXTDS
- In the TYPETERM definition for RDO – EXTENDEDDS(YES).

Using PEEK on a terminal defined with extended attributes returns a screen which looks like the black and white version of a colour television screen. The extended information is simply not returned to the LOOK program.

For this information to be returned in the EXEC CICS RECEIVE command, the terminal has to be told to return Start Field Extended (SFE) attributes (X'27'). This request is made to the terminal by adding one CICS command call before the EXEC CICS RECEIVE within the LOOK program. This command sends a ‘SET REPLY MODE’ structured field to the 3270 terminal, telling it to return SFE fields when it sends the screen buffer back to the program.
The following command should be inserted into program LOOK just prior to the EXEC CICS RECEIVE command:

```
EXEC  CICS SEND FROM(READBUF) LENGTH(READBUF) STRFIELD
```

Additionally, the following needs to be added to the constants section of the program:

```
READBUF  DC    AL2(#READBUF)            LENGTH
 DC    X'09'                    SET REPLY MODE
 DC    X'00'                    PARTITION-ID
 DC    X'02'                    EXTENDED FIELD
 DC    X'41'                    EXTENDED HIGHLIGHTING
 DC    X'42'                    FOREGROUND COLOR

#READBUF EQU   *-READBUF
```

Before this program is modified, you should take a look at the size of the TIOA buffer that will be used to accept the datastream. The SFE attributes add quite a number of extra bytes to all the fields that are returned to the program. The more fields that are on the screen, the more information will be returned. If the TIOA is not large enough, the terminal running program LOOK will suffer an ATNI abend. In my situation, it was not unusual to require a TIOA of greater than 3000 bytes. The size of the TIOA can be determined from these parameters:

- Macro level – TIOAL=(value1,value2)
- Resource Definition Online – IOAREALEN(value1,value2).

where ‘value2’ is the maximum TIOA length and, if not big enough, will result in the ATNI abend. ‘Value2’ could optionally be set to zero, which tells CICS to return the correct size of TIOA for the buffer. Please consult your CICS Resource Definition manuals for more information about these parameters.

Just a final note about program LOOK. The instruction:

```
MOVINCOM   EX  COMMAREA(*-*),Ø(4)          DUMMY FOR EXECUTE
```

will not work for a COMMAREA that is greater than 256 bytes in length. If using a larger COMMAREA, the program should be modified accordingly to use an MVCL instruction.
Microsoft has announced the enterprise version of SNA Server 4.0. This allows the reuse of CICS and IMS transactions as components for new Windows DNA applications. It also provides access to VSAM data files and to OLE/DB applications on an AS/400. Gateway enhancements will double capacity to 30,000 simultaneous sessions per server, with new features making it easier to move to TCP/IP.

For further information contact:
Microsoft, One Microsoft Way, Redmond, WA 98052-6399, USA.
Tel: (206) 882 8080.
Microsoft, Microsoft Place, Winnersh Triangle, Wokingham, Berks, RG11 5TP, UK.
Tel: (01734) 270001.

---

Compuware has announced Release 3.1 of its CICS Abend-AID/FX fault management tool, geared towards resolving transaction and region problems. It provides programmers with on-line access to information about faults, identifying problems, capturing key fault information, listing all concurrent problems, and analysing and diagnosing captured information to pinpoint the cause of the problem.

When transaction abends are compiled with a language/version that isn’t year 2000 ready, the software displays a warning message on the diagnostic summary, program summary information, and program link information screens.

For year 2000 conversions, the company says the product can speed up the diagnosis and resolution of faults that occur when testing changes or migrating to new versions of applications or operating systems, or when the converted applications are in production.

Among the new facilities are full transaction abend and region dump processing support for CICS Transaction Server for OS/390 Version 1.2, and compatibility support for DB2, IMS, and other IBM products. There’s also specific diagnostics for CICS Abend-AID/FX in the sysplex environment, and custom support for Language Environment for MVS and VM Release 1.5 and above.

For further information contact:
Compuware, 31440 Northwestern Highway, PO Box 9080, Farmington Hills, MI 48334-2564.
Tel: (800) 737 7300.
Compuware,163 Bath Road, Slough, Berks, SL1 4AA, UK.
Tel: (01753) 774000.

---

Boole & Babbage has announced a software delivery agreement with IBM Canada which bundles its MainView performance management and automation tools with IBM’s OS/390 SystemPac offering. Products included are MainView for CICS, IMS, DB2, MQSeries, AutoOperator, CMF Monitor, and InTune.

For further information contact your local IBM representative.