September 1999

In this issue

3  Using COBOL Debug
4  Y2K support issues
4  Y2K, STCK, and HDS processors
5  Executing job steps based on day of week
11 JES2 subsystem shutdown
14 Displaying Sysplex information
31 GTF data analysis using SAS
44 An MPF command exit
52 Inter-address space access program
57 An IPL subsystem (part 4)
72 MVS news

© Xephon plc 1999
Editor
Jaime Kaminski

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, EXECs, and other contents of this journal before making any use of it.

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

Subscriptions and back-issues
A year’s subscription to MVS Update, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.
Using COBOL Debug

THE PROBLEM
I frequently insert extra code, such as DISPLAYs, into a program when I am debugging it, to provide an indication as to what is happening inside. When I have completed testing the program, I dislike removing this code, because of the effort I have put into it, and I might have to do it all over again the next time I work with the code.

A SOLUTION
With COBOL’s debugging facilities, you do not have to remove the code. Here is how you do it. Set up your environment division to include code similar to the following:

```cobol
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-370 WITH DEBUGGING MODE.
OBJECT-COMPUTER. IBM-370.
```

The third line is the important part. Put a ‘D’ in column seven (the comment column), in any line of code you want to use only for debugging purposes. Make sure that the presence or absence of new code will not change the meaning of the existing surrounding code. Full stops are particularly important here. The new code will compile and execute just as if it were regular code. When you have completed testing, change the environment division to look similar to this:

```cobol
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-370.
OBJECT-COMPUTER. IBM-370.
```

Now, recompile the program again. With the ‘with debugging mode’ phrase missing, the code with the ‘D’ in column seven will now be treated as comments (as if column seven contained an asterisk – *).

Allan Kalar
Systems Programmer (USA)
© Xephon 1999
Y2K, STCK, and HDS processors

INTRODUCTION
The following provides a quick tip for anyone carrying out Y2K testing on HDS processors. It certainly does not seem to me to have been widely publicized.

If you have set up an LPAR for carrying out Y2K testing, it is possible to have an independent clock for that LPAR. This means that STCK instructions (Store Clock) will return true values. To set this up, use the LPRTOD screen on the hardware console to place the date and time required. If you go into this screen, you will also find that it is possible to ramp up the clock speed as well – should you want to simulate time passing without having to wait ‘real time’ periods. All of which can be very useful for any final integrated testing that you are carrying out.

Please note that similar facilities may exist on IBM processors, but because I have not worked on these for some years, I cannot comment.

Systems Programmer (UK) © Xephon 1999

Y2K support issues

In MVS Update 154 July 1999 it was recommended that users recontact suppliers for their latest Y2K product compliance information. As we near the millennium, there is another area that may be worth contacting your suppliers about, and that is one of support. It is essential that all suppliers of products critical to the running of your site are available for support throughout the New Year period at all hours.

Systems Programmer (UK) © Xephon 1999
Executing job steps based on day of week

In situations where I want to perform different batch processing on different days of the week, but do not want to schedule separate jobs for each, I use the simple utility program shown here. It allows me to build a single JCL deck incorporating conditional logic to process different job steps on different days of the week.

The program, SYSTBDOW, computes the weekday using a formula based on something called Zeller’s Congruence. This is a common method used for finding the weekday of a given date and is often covered in undergraduate computer science curriculums.

The general flow of the program is to (1) get the system date, which is in Julian format, (2) convert the Julian date to Gregorian, and then (3) calculate the weekday using Zeller’s Congruence. The conversion of the Julian Date to Gregorian is not robust: it would fail to realize that 1900 was not a leap year. Since the input date is always the system date and since 2000 is a leap year, the program will not produce incorrect results until the year 2100.

The program returns a condition code of zero on Sunday, one on Monday, two on Tuesday, and so on through to six on Saturday. This code can then be referenced in later steps with the COND parameter or the newer IF (stepxx.RC EQ 00) THEN syntax for JCL conditional block logic.

SYSTBDOW

***********************************************************************
* SYSTBDOW: DAY-OF-WEEK CALCULATOR
***********************************************************************

SYSTBDOW CSECT
SAVE  (14,12)
BALR  3,Ø
USING *,3
ST   13,SAVE+4
LA   13,SAVE

OPEN (OUTFILE, OUTPUT))

TIME
ST 1,FW          STORE DATE (JULIAN PACKED)
UNPK DAY+Ø(5),FW+1(3)  UNPACK JULIAN DATE
OI  DAY+4,X'FØ'     REMOVE SIGN
MVC OUTMSG+Ø(Ø),SPACES CLEAR OUT OUTPUT LINE
MVC OUTMSG+13(5),DAY MOVE JULIAN DATE TO OUTPUT LINE

PACK WKDATEYY(2),DAY+Ø(2)
PACK DIVD(4),DAY+Ø(2)
CP  DIVD,=P'5Ø'
BH  CC19
CC2Ø  ZAP WKDATECC(2),=P'2Ø'
      B  CHKLEAP
CC19  ZAP WKDATECC(2),=P'19'
      DP DIVD,=P'4'
CHKLEAP CP DIVD+3(1),=P'Ø'
BNE NEXT1
AP  FEB,=P'1'
      B  NEXT1
NEXT1 UNPK CHDATECC(2),WKDATECC(2)
      OI CHDATECC+1,X'FØ'
      UNPK CHDATEYY(2),WKDATEYY(2)
      OI CHDATEYY+1,X'FØ'

PACK JULDAY(2),DAY+2(3)
CP  JULDAY,JAN
BH  CHKFEB
ZAP WKDATEMM,=P'Ø1'
ZAP WKDATEDDD,JULDAY
      B  CHKDONE
CHKFEB SP JULDAY,JAN
CP  JULDAY,FEB
BH  CHKMAR
ZAP WKDATEMM,=P'Ø2'
ZAP WKDATEDDD,JULDAY
      B  CHKDONE
CHKMAR SP JULDAY,FEB
CP  JULDAY,MAR
BH  CHKAPR
ZAP WKDATEMM,=P'Ø3'
ZAP WKDATEDDD,JULDAY
      B  CHKDONE
CHKAPR SP JULDAY,MAR
CP  JULDAY,APR
BH  CHKMAY
ZAP WKDATEMM,=P'Ø4'
ZAP WKDATEDDD,JULDAY
      B  CHKDONE
CHKMAY SP JULDAY,APR
CP  JULDAY,MAY
CP WKDATEMM,=P'10'
BH FIXDATE
B DATEOK

FIXDATE AP WKDATEMM,=P'12'
SP WKDATEYY,=P'1'
CP WKDATEYY,=P'0'
BL FIXYEAR
B DATEOK

FIXYEAR ZAP WKDATEYY,=P'99'
SP WKDATECC,=P'1'

***********************************************************************

DATEOK ZAP TEMP,WKDATEMM
MP TEMP,=P'26'
SP TEMP,=P'2'
ZAP TEMP2,TEMP
DP TEMP2,=P'10'
ZAP TEMP,TEMP2+Ø(6)
AP TEMP,WKDATEDD
AP TEMP,WKDATEYY
ZAP TEMPYY,WKDATEYY
DP TEMPYY(3),=P'4'
ZAP TEMPCC,WKDATECC
DP TEMPCC(3),=P'4'
AP TEMP,TEMPYY+Ø(2)
AP TEMP,TEMPCC+Ø(2)
SP TEMP,WKDATECC
SP TEMP,WKDATECC
DP TEMP,=P'7'
MVC ANS(1),TEMP+5
CP ANS,=P'0'
BL ADD7
B NUMFND

ADD7 AP ANS,=P'7'

***********************************************************************

NUMFND LA 5,DAYLIT
ZAP COUNT,=P'0'

LUP CP COUNT,ANS
BE DAYFND
LA 5,3(5)
AP COUNT,=P'1'
B LUP

DAYFND MVC OUTMSG+Ø(3),Ø(5)
PUT OUTFILE,OUTMSG

***********************************************************************

CLOSE (OUTFILE)
L 13,SAVE+4

***********************************************************************

CP ANS,=P'0'
BE RCSUN
CP ANS,=P'1'
BE RCMON
CP ANS,=P'2'
BE RCTUE
CP ANS,'P'3'
BE RCWED
CP ANS,'P'4'
BE RCTHU
CP ANS,'P'5'
BE RCFRI
CP ANS,'P'6'
BE RCSAT
B ERROR
***********************************************************************
RCSUN LA 15,0
RETURN (14,12),,RC=(15)
***********************************************************************
RCMON LA 15,1
RETURN (14,12),,RC=(15)
***********************************************************************
RCTUE LA 15,2
RETURN (14,12),,RC=(15)
***********************************************************************
RCWED LA 15,3
RETURN (14,12),,RC=(15)
***********************************************************************
RCTHU LA 15,4
RETURN (14,12),,RC=(15)
***********************************************************************
RCFRI LA 15,5
RETURN (14,12),,RC=(15)
***********************************************************************
RCSAT LA 15,6
RETURN (14,12),,RC=(15)
***********************************************************************
ERROR LA 15,8
RETURN (14,12),,RC=(15)
***********************************************************************
SAVE DS 18F
CHDATE DS ØCL8
CHDATECC DS CL2
CHDATEYY DS CL2
CHDATEMM DS CL2
CHDATEDDD DS CL2
W KDAT E CC DS PL2
W KDAT E YY DS PL2
W KDAT E MM DS PL2
W KDAT E DDD DS PL2
DIVD DS PL4
JULDAY DS PL2
TEMP DS PL6
TEMP2 DS PL8
TEMPYY DS PL3
TEMPCC DS PL3
ANS DS PL1
COUNT DS PL1
CHANS DS CL1
DAY    DS   CL5
PRMADR DS   F
FW     DS   F
OUTMSG DS   CL80
SPACES DC   BX'40'

***********************************************************************
JAN    DS   ØPL2
       DC   P'Ø31'
FEB    DS   ØPL2
       DC   P'Ø28'
MAR    DS   ØPL2
       DC   P'Ø31'
APR    DS   ØPL2
       DC   P'Ø30'
MAY    DS   ØPL2
       DC   P'Ø31'
JUN    DS   ØPL2
       DC   P'Ø30'
JUL    DS   ØPL2
       DC   P'Ø31'
AUG    DS   ØPL2
       DC   P'Ø31'
SEP    DS   ØPL2
       DC   P'Ø30'
OCT    DS   ØPL2
       DC   P'Ø31'
NOV    DS   ØPL2
       DC   P'Ø30'
DEC    DS   ØPL2
       DC   P'Ø31'

***********************************************************************
DAYLIT DS   ØCL21
       DC   C'SUN'
       DC   C'MON'
       DC   C'TUE'
       DC   C'WED'
       DC   C'THU'
       DC   C'FRI'
       DC   C'SAT'

***********************************************************************
OUTFILE DCB X
       DDNAME=OUT, X
       DSORG=PS, X
       BLKSIZE=2792Ø, X
       LRECL=8Ø, X
       MACRF=(PM)

***********************************************************************
END

Tom Sager
Systems Programmer (USA)
© Xephon 1999
JES2 subsystem shutdown

In Issue 155 of *MVS Update* we considered a utility that automated TSO and system shutdown. The following utility called JESSHUT is used to assist in system shutdown. The primary function is to wait for the message ‘$HASP099 ALL AVAILABLE FUNCTION COMPLETE’, and then issue the ‘$PJES2’ on behalf of the operator. It is an automation aid. To invoke this exit, add the following line to your MPFLSTxx member:

```
$HASP099 AUTO(no),SUP(YES),EXIT(JESSHUT)
```

Use MPFLSTxx to associate message with this exit.

**SOURCE**

```//jobcard
//DOIT   EXEC ASMHCL,PARM.C='NODECK',REGION=4096K,
// PARM.L='RENT,REUS'
//C.SYSPRINT DD SYSOUT=*
//C.SYSIN  DD *
JESSHUT TITLE 'JESSHUT - WTO/WTOR EXIT ROUTINE'
** NAME           = JESSHUT *
** DESCRIPTIVE    = ISSUE '$PJES2' AFTER $HASP099 MESSAGE *
** *
** LKED ATTRIB    = RENT,REUS *
******************************************************************************
TITLE 'JESSHUT - WTO/WTOR EXIT'
PRINT NOGEN
JESSHUT CSECT
JESSHUT AMODE 24
******************************************************************************
** STANDARD ENTRY LINKAGE *
******************************************************************************
BALR R15,Ø
BCTR R15,Ø
BCTR R15,Ø
SAVE (14,12)       SAVE REGISTERS
LR R12,R15          ESTABLISH BASE
USING JESSHUT,R12
LR R2,R1            SAVE R1
GETMAIN RC,LV=WORKLEN,SP=230,LOC=BEG
LTR R15,R15         GETMAIN IS OK?
BNZ NOWORK          BRANCH ON NO
USING WORKAREA,R1
ST R13,WSAVEREG+4   CHAIN
```
ST R13,WSAVEREG+8       SAVE
ST R1,8(R13)           AREAS
DROP R1
LR R13,R1           ESTABLISH BASE FOR WORKAREA
USING WORKAREA,R13
ST R2,WR1          SAVE R1
B START          BRANCH AROUND LITERAL
SPACE
DC C'JESSHUT-'      NAME
DC C'&SYSDATE'
DC C'&SYSTIME'
SPACE 2
TITLE 'JESSHUT - MAINLINE'
*********************************************************************
**       MAINLINE                                                   *
*********************************************************************
START    DS ØH
L     R11,WR1
L     R11,Ø(R11)         ESTABLISH ADDRESSABILITY
USING CTXT,R11        TO THE CTXT
L     R2,CTXTTXPJ        ESTABLISH ADDRESSIBILITY
USING CTXTATTR,R2      TO THE MSG ATTRIBUTE
LA     R3,CTXTTMSG      GET ADDRESS OF TEXT AREA
USING MSGTEXT,R3
CLC   MHASPØ99,MSGID    MESSAGE $HASPØ99?
BNE   EXIT            NO? GET THE HELL OUT
DROP  R3
XC    MGCRPL(MGCRPLTH),MGCRPL  CLEAR PARM LIST
MVC   MGCRTEXT(L'TXTINSRT),TXTINSRT  MOVE REPLY TO BUFFER
LA     R1,(MGCRTEXT-MGCRPL)+L'TXTINSRT  GET MSG LENGTH
STC   R1,MGCRLGTH        SAVE LENGTH
SR     RØ,RØ
MGCR  MGCRPL
SR     R15,R15
B     EXIT
TITLE 'JESSHUT - NOWORK'
*********************************************************************
**       NOWORK                                                     *
*********************************************************************
NOWORK   DS ØH
WTO   'HASPØ99 == HASPØ99 ;  GETMAIN FAILED'
SR     R15,R15
B     EXITNOW
TITLE 'JESSHUT - EXIT'
*********************************************************************
**       EXIT                                                       *
*********************************************************************
EXIT     DS ØH
LR     R2,R13
L     R3,4(R13)
FREEMAIN RC,LV=WORKLEN,A=(2),SP=23Ø
LTR     R15,R15
BZ    FREEOK
WTO '$HASPO99 == $HASPO99 ; GETMAIN FAILED'

FREEOK DS 0H
LR R13,R3
SR R15,R15
EXITNOW DS 0H
RETURN (14,12),RC=(15)
TITLE 'JESSHUT - CONSTANTS'
DS 0F
MHASPO99 DC C'$HASPO99'              ALL AVAILABLE FUNCTIONS
TXTINSRT DC C'$PJES2'             TITLE 'JESSHUT - WORK AREA DSECT'
WORKAREA DSECT
WSAVEREG DS 18A                      REGISTER SAVE AREA
WR1 DS A                        R1 SAVE AREA
IEZMGCR DSECT=NO
ORG MGRTEXT
COMMAND DS CL20                REPLY COMMAND
ORG
WORKLEN EQU *-WORKAREA
TITLE 'JESSHUT - MSGTEXT'
MSGTEXT DSECT
MSGID DS CL8                   TITLE 'JESSHUT - EQUATES'
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
TITLE 'JESSHUT - IBM SYSTEM DSECTS'
IEZVX100
CVT DSECT=YES,LIST=YES
END

//L.SYSLMOD DD DSN=xxxxx.xxxxx.loadlib,DISP=OLD
//L.SYSIN DD *
SETCODE AC(1)
NAME JESSHUT(R)
/*

Salah Balboul
Systems Programmer (USA) © Xephon 1999
Displaying Sysplex information

INTRODUCTION
The IXCQUERY service is useful for displaying Sysplex information. It provides information similar to that provided by the command ‘D XCF’, except you can get more information. The report consists of a list of Sysplex objects:

• Basic configuration (systems in the Sysplex).
• Coupling facility information.
• Groups and members of the groups (a group is a set of related members that a Sysplex-wide application defines to XCF).
• Structures and connections (a structure is an object used by MVS to manage storage on a coupling facility).

For the CF and structure inquiry, the program may return a RC 12, reason 324 if no CF (or equivalent) is installed on your site. However, group information will be displayed if it is a monoplex.

EXAMPLE OUTPUT
A shortened example of what the program would display on a three-system Sysplex is shown below. The only structure here is IEFAUTOS, used for automatic tape switching (it stores the status of on-line automatically switchable tape devices across the Sysplex).

SYSPLEX CONFIGURATION ON 17/06/1999 AT 10:52:09 - SYSTEM SYS1 OS/390 V 02 R 04
PLEXNAME=XYPLEX0 CFLVL= 5 SYS MAX/CURRENT= 32 / 10

* IXQUERY SYSPLEX - RC= 0 REASON= 0 RECORDS= 3
SYS1 (S1) SLOT= 1 INTERVAL= 30 OPCODE= 35 ACTIVE STAT-UPD:17/06/1999
10:52:08
SYS2 (S2) SLOT= 2 INTERVAL= 30 OPCODE= 35 ACTIVE STAT-UPD:17/06/1999
10:52:05
SYS3 (S3) SLOT= 3 INTERVAL= 30 OPCODE= 35 ACTIVE STAT-UPD:17/06/1999
10:52:07

* IXQUERY CF - RC= 0 REASON= 0 RECORDS= 1
CF CF01 ID=009672 IBM0100000005321 DUMP SZ= 2048K CFRMPOL=CFRM01
STRUCTURES= 1 SYSTEMS= 3
* IXCQUERY GROUPS - RC= Ø REASON= Ø RECORDS= 18
GROUP SYSGRS ——— 3 MEMBER(S)
  SYS1 ACT ON/SYS1 JOINED/GRS ACTIVE UPD:13/06/1999 11:25:43
  SYS2 ACT ON/SYS2 JOINED/GRS ACTIVE UPD:13/06/1999 11:25:43
  SYS3 ACT ON/SYS3 JOINED/GRS ACTIVE UPD:13/06/1999 11:28:10
GROUP SYSMCS ——— 8 MEMBER(S)
  SYSMCS$MCS ACT ON/ JOINED/ CREATED UPD:16/06/1999 17:57:51
  SYSMCS$CL1 ACT ON/ JOINED/ CREATED UPD:17/06/1999 10:50:16
  SYSMCS$CL2 ACT ON/ JOINED/ CREATED UPD:17/06/1999 10:41:44
  SYSMCS$CL3 ACT ON/ JOINED/ CREATED UPD:17/06/1999 10:49:23
  SYSMCS$EMCS ACT ON/ JOINED/ CREATED UPD:17/06/1999 10:29:12
GROUP IXCLO000 ——— 3 MEMBER(S)
  M30 ACT ON/SYS1 JOINED/ALLOCAS ACTIVE UPD:13/06/1999 08:35:13
  M29 ACT ON/SYS2 JOINED/ALLOCAS ACTIVE UPD:13/06/1999 06:54:48
  M32 ACT ON/SYS3 JOINED/ALLOCAS ACTIVE UPD:13/06/1999 11:28:21
GROUP ISTCFS01 ——— 3 MEMBER(S)
  XXYZY5000FRXYZS00 ACT ON/SYS1 JOINED/VTAM1 ACTIVE UPD:13/06/1999 08:38:52
  CDRM00S$FRXYZS00 ACT ON/SYS3 JOINED/VTAM3 ACTIVE UPD:13/06/1999 11:33:28
  CDRM02S$FRXYZS00 ACT ON/SYS2 JOINED/VTAM2 ACTIVE UPD:13/06/1999 07:01:04
GROUP SYSRMF ——— 3 MEMBER(S)
  SYSRMFaaaaSYS1 ACT ON/SYS1 JOINED/RMF ACTIVE UPD:13/06/1999 08:39:12
  SYSRMFaaaaSYS3 ACT ON/SYS3 JOINED/RMF ACTIVE UPD:13/06/1999 11:33:55
  SYSRMFaaaaSYS2 ACT ON/SYS2 JOINED/RMF ACTIVE UPD:13/06/1999 07:01:30
GROUP DFHIR000 ——— 8 MEMBER(S)
  CIC9 ACT ON/SYS1 JOINED/CICSR9 ACTIVE UPD:16/06/1999 03:28:25
  CIC0 ACT ON/SYS1 JOINED/CICSR0 ACTIVE UPD:16/06/1999 03:29:30
  CIC4 ACT ON/SYS1 JOINED/CICSR4 ACTIVE UPD:17/06/1999 03:31:22
  CIC7 ACT ON/SYS1 JOINED/CICSR7 ACTIVE UPD:16/06/1999 03:31:16
  CIC5 ACT ON/SYS1 JOINED/CICSR5 ACTIVE UPD:16/06/1999 08:53:04
  CIC3 ACT ON/SYS1 JOINED/CICSR3 ACTIVE UPD:17/06/1999 06:01:16
  CIC1 ACT ON/SYS1 JOINED/CICSR1 ACTIVE UPD:17/06/1999 06:01:20
  CIC2 ACT ON/SYS1 JOINED/CICSR2 ACTIVE UPD:17/06/1999 06:01:27

* IXCQUERY STRUCTURES - RC= Ø REASON= Ø RECORDS= 1
IEFAUTOS IN CF POLICYSZ= 600K CFRMPOL=CFRM01
REBUILD:N/A
- CONNECTION IEFAUTOSSYS1 ACTIVE SYSTEM=SYS1 JOBN=ALLOCAS VERS=0001000C
- CONNECTION IEFAUTOSSYS2 ACTIVE SYSTEM=SYS2 JOBN=ALLOCAS VERS=00020013
- CONNECTION IEFAUTOSSYS3 ACTIVE SYSTEM=SYS3 JOBN=ALLOCAS VERS=0003000F

SOURCE
* DISPLAYING SYSPLEX INFORMATION THROUGH THE IXCQUERY SERVICE
* SYSPLEX CSECT
* THIS MACRO EXPANDS A BINARY STRING (&BIN), LENGTH 4 BYTES
  INTO AN HEXADECIMAL STRING, LENGTH 9 BYTES
MACRO
&NLA PRINTHX &BIN,&DISPL
&NLA DS ØH
UNPK &DISPL.(9),&BIN.(5) EXPAND : X'1D' -> X'F1FD'
NC &DISPL.(8),=XL'0F0F0F0F0F0F0F0F' X'F1FD' -> X'010D'
TR &DISPL.(8),=C'0123456789ABCDEF' GET HEXADECIMAL DATA
MVI &DISPL+8,C' " LAST BYTE IS MEANINGLESS
MEXIT
MEND
* THIS MACRO ACHIEVES AN ADDRESSING-MODE SWITCH
MACRO
&NAME SETAMOD &MODE,&WORKREG=R1
AIF ("&MODE" EQ "24").AMODE24
AIF ("&MODE" EQ "31").AMODE31
MNOTE 8,'*** ERROR *** MODE MUST BE 24 OR 31'
MEXIT
.AMODE24 ANOP
&NAME LA &WORKREG,*+6 LOAD ADDR WITH BIT Ø OFF
BSM Ø,&WORKREG BRANCH AND SET AMODE 24
MEXIT
.AMODE31 ANOP
&NAME ICM &WORKREG,15,*+6 LOAD ADDR WITH BIT Ø ON
BSM Ø,&WORKREG BRANCH AND SET AMODE 31
DC AL4(*+4+X'80000000')
MEXIT
MEND
* THIS MACRO PERFORMS A MOVE CONDITIONAL, BASED ON VALUES OF &TEST ZONE
MACRO
&NLA MVCCASE &TO=,&TEST=,&TSTCOD=CLI,&FAILCOD=BNE,&IF=,&MOVE=
&NLA DS ØH
LCLA &NBR,&I
&NBR SETA N'&IF HOW MANY VALUES TO TEST?
&I SETA 1
.LOOP &TSTCOD &TEST,&IF(&I) TEST FOR VALUE
&FAILCOD NO&SYSNDX.&I
MVC &TO,&MOVE(&I) MATCH, MOVE THE VALUE
B ENDM&SYSNDX AND GO OUT
NO&SYSNDX.&I DS ØH
&I SETA &I+1
AIF (&I LE &NBR).LOOP
ENDM&SYSNDX DS ØH
MEXIT
MEND
* THIS MACRO CONVERTS A BINARY FULLWORD TO DECIMAL (1Ø BYTES)
MACRO
EXTENDW &BIN,&ETEND
XR R1,R1 SET REGISTER TO ZERO
ICM R1,15,&BIN LOAD BINARY ZONE
CVD R1,DOUBLEWORD CONVERT TO DECIMAL
MVC &ETEND.(10),MASK2 LOAD MASK FOR ED INSTRUCTION
ED &ETEND.(10),PACKED5 CONVERSION FROM PACKED
* SHIFT DATA 5 BYTES LEFT (SO WE SUPPORT ONLY &BIN=Ø TO 999999 |)

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
MVC  &ETEND.(5),&ETEND+5  0000NNNNNNN -> NNNNNNNNNN
MVC  &ETEND+5(5),=CL5' " NNNNNNNNNN -> NNNNN
MEND

* THIS MACRO CONVERTS A BINARY HALFWORD TO DECIMAL

MACRO
EXTEND  &BIN,&ETEND
XR    R1,R1                  SET REGISTER TO ZERO
ICM   R1,3,&BIN              LOAD BINARY ZONE
CVD   R1,DBLEWORD            CONVERT TO DECIMAL
MVC   &ETEND.(6),MASK1       LOAD MASK FOR ED INSTRUCTION
ED    &ETEND.(6),PACKED3     CONVERSION FROM PACKED TO DECIMAL
MEND

* BASE REGISTERS ARE R12 AND R11

STM   R14,R12,12(R13)        SAVE REGISTERS
LR    R12,R15
LA    R11,4Ø95(Ø,R15)
LA    R11,1(Ø,R11)
USING SYSPLEX,R12,R11        R12, R11 BASE REGISTERS
ST    R13,SAVE+4
LA    R13,SAVE

* APF VERIFY

TESTAUTH FCTN=1
LTR   R15,R15
BZ    APFOK
WTO   "SYSPLEX PROGRAM NOT APF-AUTHORIZED, ENDING",ROUTCDE=11
B     FINISH

APFOK    DS    ØH

* GET A BUFFER FOR THE IXQUERY FUNCTION

L     R9,SIZE                SIZE OF BUFFER
GETMAIN  R,LV=(9)            SAVE ADDRESS OF ACQUIRED ZONE
ST    R1,ADDZONE             SAVE ADDRESS OF ACQUIRED ZONE

* GET A SECOND BUFFER FOR THE IXQUERY FUNCTION (NESTED QUERY)

L     R9,SIZE                SIZE OF BUFFER
GETMAIN  R,LV=(9)            SAVE ADDRESS OF ACQUIRED ZONE
ST    R1,ADDZONE2            SAVE ADDRESS OF ACQUIRED ZONE

* OPEN OUTPUT FILE

OPEN  (SYSPRINT,(OUTPUT))

* TIME AND DATE

TIME DEC,WORD4,LINKAGE=SYSTEM,DATETYPE=DDMYYYY
PRINTHX  TIME1,TIMEØ            TIMEØ = HHHHMMSS
MVC   TIMEØ+6(2),TIMEØ+4        HHMM??SS
MVC   TIMEØ+4(1),TIMEØ+3        HH?MM?SS
MVC   TIMEØ+3(1),TIMEØ+2        HH?MM?SS
MVI   TIMEØ+2,C':'             HH:MM?SS
MVI   TIMEØ+5,C':'             HH:MM:SS
PRINTHX  DATE1,DATEØ           DATEØ = DDMYYYY??
MVC   DATEØ+8(2),DATEØ+6       DDM??YYYY
MVC   DATEØ+6(2),DATEØ+4       DDM??YYYY
MVC   DATEØ+4(1),DATEØ+3       DD?MM?YYYY
MVC   DATEØ+3(1),DATEØ+2       DD?MM?YYYY
MVI   DATEØ+2,C'/'             DD/MM?YYYY
MVI   DATEØ+5,C'/'             DD/MM/YYYY

* SYSTEM INFORMATION
L     R2,16                  CVT ADDRESS
MVC   SYSTNAME,X'154'(R2)    CVTSNAME FOR SYSTEM NAME
L     R3,8'C'(0,R2)          ECVT ADDRESS
SETAMOD 31
MVC   SYSTOS39,X'1F6'(R3)    SYSTEM INFORMATION FROM E-CVT
MVC   SYSTVERS,X'200'(R3)    OS/390 VERSION
MVC   SYSTREL,X'202'(R3)     OS/390 RELEASE
SETAMOD 24
MVC   RECORD,RECORDT         MOVE RECORD IN OUTPUT ZONE
BAL   R10,PUTREC             PRINT THE DATA

* INVOKE IXCQUERY FOR COUPLE INFO

* MODE SUP
  MODESET MODE=SUP,KEY=ZERO
  SETAMOD 31
  IXCQUERY REQINFO=COUPLE,
          SYSPLEXID=SYSPLEXID,
          PLEXNAME=PLEXNAME,
          MAXSYS=MAXSYS,
          CURRMAXSYS=CURRMAX,
          CFLEVEL=CFLEVEL
  SETAMOD 24
  MODESET MODE=PROB,KEY=NZERO
  EXTENDW CFLEVEL,$CFLV
  EXTENDW MAXSYS,$MAXSYS
  EXTENDW CURRMAX,$CURRENT
  SYSPLEXI A TRAITER ***************
  MVC   RECORD,RECORDC
  BAL   R10,PUTREC

* INVOKE IXCQUERY FOR SYSPLEX INFO

* MODE SUP
  BAL   R8,CLRZONE
  MODESET MODE=SUP,KEY=ZERO
  L     R2,ADDZONE
  IXCQUERY REQINFO=SYSPLEX,ANSAREA=(R2),
          ANSLEN=SIZE,
          RETCODE=RETURN,
          RSNCODE=REASON
  MODESET MODE=PROB,KEY=NZERO
  MVC   $HTYPE,=CL10"SYSPLEX"
  BAL   R8,PROCESS_HEADER
  LTR   R9,R9
  RECORD PRESENT?
  BZ    NOSYSP
  LOOPSYSP BAL   R8,PROCESS_SYSPLEX
  LTR   R9,R9
  NEXT RECORD PRESENT?
  BNZ   LOOPSYSP
  NOSYSP DS 0H

* INVOKE IXCQUERY FOR CF INFO
* MODE SUP
  BAL R8,CLRZONE BUFFER CLEARING
  MODESET MODE=SUP,KEY=ZERO
  L R2,ADDZONE OBTAIN ADDRESS OF OUTPUT AREA
  IXCQUERY REQINFO=CF_ALLDATA,ANSAREA=(R2), X
  ANSLEN=SIZE,RETCODE=RETURN, X
  RSNCODE=REASON
  MODESET MODE=PROB,KEY=NZERO
  MVC $HTYPE,=CL1Ø"CF''
  BAL R8,PROCESS_HEADER
  LTR R9,R9 RECORD PRESENT?
  BZ NOCFREC

LOOPCF BAL R8,PROCESS_CF
LTR R9,R9 NEXT RECORD PRESENT?
BNZ LOOPCF

NOCFREC DS ØH

*————————————————————————————————————————————————————————————————————*
* INVOKE IXCQUERY FOR GROUP INFO
*————————————————————————————————————————————————————————————————————*

* MODE SUP
  BAL R8,CLRZONE BUFFER CLEARING
  MODESET MODE=SUP,KEY=ZERO
  L R2,ADDZONE OBTAIN ADDRESS OF OUTPUT AREA
  IXCQUERY REQINFO=GROUP,ANSAREA=(R2), X
  ANSLEN=SIZE,RETCODE=RETURN, X
  RSNCODE=REASON
  MODESET MODE=PROB,KEY=NZERO
  MVC $HTYPE,=CL1Ø"GROUPS''
  BAL R8,PROCESS_HEADER
  LTR R9,R9 RECORD PRESENT?
  BZ NOGRP

LOOPGRP BAL R8,PROCESS_GROUP
  * NESTED LOOP FOR MEMBERS IN THE GROUP - BEGINNING
  STM R7,R1Ø,SAVE4 SAVE REGS FOR MAIN GROUP LOOP
  BAL R8,CLRZONE2 BUFFER CLEARING
  MODESET MODE=SUP,KEY=ZERO
  L R2,ADDZONE2 ADDRESS OF OUTPUT AREA
  IXCQUERY REQINFO=GROUP,ANSAREA=(R2),GRPNAME=$GRPN, X
  ANSLEN=SIZE,RETCODE=RETURN, X
  RSNCODE=REASON
  MODESET MODE=PROB,KEY=NZERO
  L R9,ADDZONE2 ADDRESS DATA
  L R7,(QUAH$REC-QUAHDR)(R9) NUMBER OF MEMBER RECORDS
  LTR R7,R7
  BZ NORECMØ NO MEMBER RECORD
  A R9,(QUAHSGOF-QUAHDR)(R9) OFFSET

LOOPMEM BAL R8,PROCESS_MEMBER
  LTR R9,R9 NEXT MEMBER RECORD PRESENT?
  BNZ LOOPMEM

NORECMØ DS ØH
  LM R7,R1Ø,SAVE4 RESTORE REGS FOR MAIN GROUP LOOP
  * NESTED LOOP FOR MEMBERS IN THE GROUP - END
LTR   R9,R9                  NEXT GROUP RECORD PRESENT?
BNZ   LOOPGRP

* INVOKE IXCQUERY FOR STRUCTURE INFO

BAL   R8,CLRZONE             BUFFER CLEARING
MODESET MODE=SUP,KEY=ZERO
L     R2,ADDZONE             OBTAIN ADDRESS OF OUTPUT AREA
IXCQUERY REQINFO=STR,ANSAREA=(R2), X
   ANSLEN=SIZE,RETCODE=RETURN, X
   RSNCODE=REASON
MODESET MODE=PROB,KEY=NZERO
MVC $HTYPE,=CL1"STRUCTURES'
BAL   R8,PROCESS_HEADER
LTR   R9,R9                  RECORD PRESENT?
BNZ   NOSTRUC
LOOPSTR BAL   R8,PROCESS_STRUCTURE

* NESTED LOOP FOR CONNECTIONS TO THE STRUCTURE - BEGINNING
STM   R7,R10,SAVE4        SAVE REGS FOR MAIN LOOP
BAL   R8,CLRZONE2         BUFFER CLEARING
MODESET MODE=SUP,KEY=ZERO
L     R2,ADDZONE2         ADDRESS OF OUTPUT AREA
IXCQUERY REQINFO=STR,ANSAREA=(R2), X
   ANSLEN=SIZE,RETCODE=RETURN,STRNAME=$STRN, X
   RSNCODE=REASON
MODESET MODE=PROB,KEY=NZERO
L     R9,ADDZONE2         ACCESS DATA
L     R7,(QUAH$REC-QUAHDR)(R9)   NUMBER OF MEMBER RECORDS
LTR   R7,R7
BZ    NORECSØ             NO MEMBER RECORD
A     R9,(QUAH$GOF-QUAHDR)(R9)   Offset

* POSITIONING ON CONNECTOR RECORD IN THE BUFFER
LOOPSEEK   CLI   Ø(R9),X'00'         NO MORE INFORMATION?
BE    NORECSØ             YES,NO CONNECTION RECORD FOUND
CLI   Ø(R9),X'24'         CONNECTION RECORD?
BE    FOUNDCON          YES
CLI   Ø(R9),X'A4'         CONNECTION RECORD?
BE    FOUNDCON          YES
AH    R9,2(R9)            GO TO NEXT RECORD IN BUFFER
B     LOOPSEEK
FOUNDCON   BAL   R8,PROCESS_CONNECTOR
LTR   R9,R9                  NEXT CONNECTION RECORD PRESENT?
BNZ   FOUNDCON

NORECSØ   DS   ØH
LM    R7,R10,SAVE4        RESTORE REGS FOR MAIN LOOP

* NESTED LOOP FOR CONNECTIONS TO THE STRUCTURE - END
LTR   R9,R9                  NEXT STRUCTURE RECORD PRESENT?
BNZ   LOOPSTR

NOSTRUC   DS   ØH
THEEND   CLOSE   (SYSPRINT)
L     R10,ADDZONE
* SUBROUTINE TO PUT A RECORD

```
PUTFREC  PUT   SYSPRINT,RECORD
MVC   RECORD,$SPACES
BR    R1Ø
```

* HEADER PROCESSING
* INPUT : ADDZONE
* RETURNS R9=ADDRESS OF 1ST DATA RECORD (=Ø IF NONE)
* RETURNS R2=NUMBER OF RECORDS

```
PROCESS_HEADER DS ØH
XR    R2,R2
L     R7,ADDZONE            ADDRESS OF OUTPUT AREA
USING QUAHDR,R7             HEADER SECTION
L     R2,QUAH$REC           NUMBER OF RECORDS WHICH FOLLOW
LTR   R2,R2
BNZ   RECEXIST              RECORDS EXIST
XR    R9,R9                 NO RECORD
B     NOREC                 NO RECORD
RECEXIST L     R9,ADDZONE            ACCESS DATA
A     R9,QUAHSGOF        OFFSET FROM QUAHDR TO 1ST DATA RECORD
NOREC    DS    ØH
EXTENDW RETURN,$HRC         RETURN CODE
EXTENDW REASON,$HRS         REASON CODE
EXTENDW QUAH$REC,$HREC      RECORDS
MVC   RECORD,$SPACES        MOVE BLANK LINE IN OUTPUT ZONE
BAL   R1Ø,PUTREC            PRINT A BLNK LINE
MVC   RECORD,RECORDH
BAL   R1Ø,PUTREC            PRINT THE DATA
BR    R8
DROP  R7
```

* SYSPLEX RECORD PROCESSING
* INPUT : R9 (RECORD ADDRESS)
* RETURNS R9=ADDRESS OF NEXT SYSPLEX RECORD (=Ø IF NONE)

```
PROCESS_SYSPLEX DS ØH
LR    R7,R9
USING QUASYS,R7             SYSPLEX SECTION
* PREPARE NEXT RECORD
XR    R9,R9                 LAST SYSPLEX RECORD
CLI   QUASTYPE,X'82'
BE    LASTS
CLI   QUASTYPE,X'Ø2'
BNE   NORECS                 NOT A SYSPLEX RECORD
LR    R9,R7
```
HALFWORD,HALFWORD CLEAR WORK HALFWORD
MVC HALFWORD+1(1),QUASNUM SLOT NUMBER
EXTEND HALFWORD,WORK6 EXTEND NUMBER
MVC $SYSSLOT,WORK6+4

* STATUS
MVC $SYSSTAT,$SPACES NOT SPECIFIED BY DEFAULT
MVCCASE TEST=QUASSTAT,TO=$SYSSTAT,TSTCOD=TM,FAILCOD=BNO,
IF=(QUASACTV,QUASSUM,QUASSYPT,QUASLOCL,QUASCLUP),
MOVE=(-CL2Ø"ACTIVE",
-CL2Ø"STAT-UPDATE MISSING ",
-CL2Ø"SYSPLEX PARTITIONING",
-CL2Ø"SINGLE SYS- NO CP DS",
-CL2Ø"PART-ED, IN CLEANUP ")

* MONITOR INTERVAL
XR RØ,RØ RØ/R1 = DIVIDEND
L R1,QUASINTV HUNDREDTHS OF SECONDS
LA R2,1Ø0 DIVISOR
DR RØ,R2
ST R1,QUASINTV STORE QUOTIENT
EXTENDW QUASINTV,$SYSINTV

* OPERATOR INTERVAL
XR RØ,RØ RØ/R1 = DIVIDEND
L R1,QUASOPIN HUNDREDTHS OF SECONDS
LA R2,1Ø0 DIVISOR
DR RØ,R2
ST R1,QUASOPIN STORE QUOTIENT
EXTENDW QUASOPIN,$SYSOPIN

* TOD OF LAST UPDATE
STCKCONV STCKVAL=QUASSUTO,CONVVAL=WORD4,DATETYPE=DDMMYYYY
PRINTHX TIME1,$SYSTIME
MVC $SYSTIME+6(2),$SYSTIME+4 HHMM??SS
MVC $SYSTIME+4(1),$SYSTIME+3 HH?MM?SS
MVC $SYSTIME+3(1),$SYSTIME+2 HH?MM?SS
MVI $SYSTIME+2,C':' HH:MM?SS
MVI $SYSTIME+5,C':' HH:MM:SS
PRINTHX DATE1,$SYSDATE
MVC $SYSDATE+8(2),$SYSDATE+6 DDMM??YYYY
MVC $SYSDATE+6(2),$SYSDATE+4 DDMM??YYYY
MVC $SYSDATE+4(1),$SYSDATE+3 DD?MM?YYYY
MVC $SYSDATE+3(1),$SYSDATE+2 DD?MM?YYYY
MVI $SYSDATE+2,C'/' DD/MM?YYYY
MVI $SYSDATE+5,C'/' DD/MM/YYYY

* PRINT THE DATA
MVC RECORD,RECORDS MOVE RECORD IN OUTPUT ZONE
BAL R1Ø,PUTREC PRINT THE DATA
NORECS DS ØH
BR R8
DROP R7

* GROUP RECORD PROCESSING
* INPUT : R9 (RECORD ADDRESS)
* RETURNS R9=ADDRESS OF NEXT GROUP RECORD (=Ø IF NONE)

PROCESS_GROUP DS ØH
LR R7,R9
USING QUAGRP,R7

* PREPARE NEXT RECORD
XR R9,R9
CLI QUAGTYPE,X'80'
BE LASTG
CLI QUAGTYPE,X'00'
BNE NORECG
LR R9,R7
AH R9,QUAGLEN

LASTG DS ØH
MVC $GRPN,QUAGNAME

* NUMBER OF MEMBERS IN THE GROUP
EXTENDW QUAG$MEM,$GRPNB

* PRINT THE DATA
MVC RECORD,RECORDG
BAL R1Ø,PUTREC

NORECG DS ØH
BR R8
DROP R7

* MEMBER RECORD PROCESSING
* INPUT : R9 (RECORD ADDRESS)
* RETURNS R9=ADDRESS OF NEXT MEMBER RECORD (=Ø IF NONE)

PROCESS_MEMBER DS ØH
LR R7,R9
USING QUAMEM,R7

* PREPARE NEXT RECORD
XR R9,R9
CLI QUAMTYPE,X'81'
BE LASTM
CLI QUAMTYPE,X'01'
BNE NORECM
LR R9,R7
AH R9,QUAMLEN

LASTM DS ØH
MVC $MEMN,QUAMNAME

* MEMBER STATE
MVC $MEMST,$SPACES
MVCCASE TEST=$MEMST,TO=$MEMST,TSTCOD=CLI,FAILCOD=BNE,
 IF=(QUAMSCRE,QUAMSACT,QUAMSQUI,QUAMSFLD),
 MOVE=(=CL8"CREATED'.,=CL8"ACTIVE',=CL8"QUIESCED',=CL8"FAILED'

* ADDITIONAL STATUS INFORMATION

MVC $MEMST2,$SPACES NOT SPECIFIED BY DEFAULT
MVCCASE TEST=QUAMSTA2,TO=$MEMST2,TSCOD=TM,FAILC=BN0,
 IF=(QUAMSSSM,QUAMSTRM,QUAMSMSM,QUAMSLD),
 MOVE=("=CL18"SYS STAT UPD MISS ",
 =CL18"TERMINATING ",
 =CL18"MBR STAT UPD MISS ",
 =CL18"UPD MISS DETECTED ")

* SYSTEM NAME THAT MEMBER WAS LAST ACTIVE ON
MVC $MEMSYS,QUAMSYS SYSTEM NAME
* PRIMARY ASID CURRENT AT JOIN TIME
MVC $MEMJOB,QUAMJOB JOB NAME
* TOD OF LAST UPDATE
STCKCONV STCKVAL=QUAMTOD,CONVVAL=WORD4,DATETYPE=DDMMYYYY
PRINTHX TIME1,$MEMTIME
MVC $MEMTIME+6(2),$MEMTIME+4 HHMM??SS
MVC $MEMTIME+4(1),$MEMTIME+3 HH?MM?SS
MVC $MEMTIME+3(1),$MEMTIME+2 HHMM?SS
MVI $MEMTIME+2,C':' HH:MM?SS
MVI $MEMTIME+5,C':' HH:MM:SS
PRINTHX DATE1,$MEMDATE
MVC $MEMDATE+8(2),$MEMDATE+6 DDMM??YYYY
MVC $MEMDATE+6(2),$MEMDATE+4 DDMM??YYYY
MVC $MEMDATE+4(1),$MEMDATE+3 DD?MM?YYYY
MVC $MEMDATE+3(1),$MEMDATE+2 DD/MM?YYYY
MVI $MEMDATE+2,C:'/ DD/MM/YYYY
MVI $MEMDATE+5,C:'/ DD/MM/YYYY

* PRINT THE DATA
MVC RECORD,RECORDM MOVE RECORD IN OUTPUT ZONE
BAL R1Ø,PUTREC PRINT THE DATA
NORECM DS ØH
BR R8
DROP R7

*————————————————————————————————————————————————————————————————————*
* STRUCTURE RECORD PROCESSING
* INPUT : R9 (RECORD ADDRESS)
* RETURNS R9=ADDRESS OF NEXT STRUCTURE RECORD (=Ø IF NONE)
*————————————————————————————————————————————————————————————————————*
PROCESS_STRUCTURE DS ØH
LR R7,R9
USING QUASTR,R7 STRUCTURE SECTION
* PREPARE NEXT RECORD
XR R9,R9 LAST STRUCTURE RECORD
CLI QUASTRTYP,X'AØ'
BE LASTSTR
CLI QUASTRTYP,X'20'
BNE NORECST NOT A STRUCTURE RECORD
LR R9,R7
AH R9,QUASTRLLEN NEXT RECORD POSITION
LASTSTR DS ØH
MVC $STRN,QUASTRNAME STRUCTURE NAME
* SIZE OF STRUCTURE AS SPECIFIED IN CFRM ACTIVE POLICY (MULT OF 4K)
ICM R1,15,QUASTRSIZE SIZE IS IN 4K MULTIPLE
MH R1.=H'4000'       NOW IN BYTES
XR R0,R0
D R0.=F'1000'        NOW IN K
STCM R1,15,QUASTRSIZE SIZE IN KBYTES
EXTENDW QUASTRSIZE,*STRSZ EXTEND NUMBER
MVI $STRSZ+5,C'K'    MENTION «K»-BYTES

* THIRD BYTE OF STATE INDICATORS
MVC $STRSTAT,$SPACES SET TO ALL BLANKS
MVCCASE TEST=QUAARSTATE3,TSTCOD=TM,FAILCOD=BNO, X
TO=$STRSTAT, X
IF=(QUAARSTDISP,QUAARSTREBLD,QUAARSTREBLDSTOP, X
QUAARSTALTER,QUAARSTINCLEANUP), X
MOVE=(=CL19"STROISP=KEEP", X
=CL19"REBUILD IN PROGRESS", X
=CL19"REBUILD STOPPED ", X
=CL19"ALTER IN PROGRESS ", X
=CL19"CLEANUP IN PROGRESS")

* INDICATES STRUCTURE IN COUPLING FACILITY
MVC $STRINCF,$SPACES SET TO ALL BLANKS
MVCCASE TEST=QUAARINHDW,TSTCOD=TM,FAILCOD=BNO, X
TO=$STRINCF, X
IF=(QUAARINHDWON),MOVE=(=CL5"IN CF")

* INITSIZE OF STRUCT AS SPECIFIED IN CFRM ACTIVE POLICY (MULT OF 4K)
* EXTENDW QUASRINITSIZE,*STRSZIN EXTEND NUMBER
* CFRM POLICY NAME
MVC $STRCFRM,QUASTRPOLNAME

* PHASE FOR THE REBUILD STRUCTURE PROCESS
MVC $STRSTRE,=CL20"N/A'
MVCCASE TEST=QUAARSTREPHASE,TSTCOD=TM,FAILCOD=BNO, X
TO=$STRSTRE, X
IF=(QUAARSTREQUIESCE,QUAARSTRECOMPLETE, X
QUAARSTRECLEANUP,QUAARSTREBLDSTOP), X
MOVE=(=CL17"INITIATED", X
=CL17"IN PROGR/COMPLETE", X
=CL17"IN PROGR/CLEANUP ", X
=CL17"HAS BEEN STOPPED ")

* PRINT THE DATA
MVC RECORD,RECORDST MOVE RECORD IN OUTPUT ZONE
BAL R10,PUTREC PRINT THE DATA

* CONNECTOR RECORD PROCESSING
* INPUT : R9 (RECORD ADDRESS)
* RETURNS R9=ADDRESS OF NEXT CONNECTOR RECORD (=Ø IF NONE)

PROCESS_CONNECTOR DS ØH
LR R7,R9
USING QUASRUSER,R7 CONNECTOR SECTION
* PREPARE NEXT RECORD
XR R9,R9 LAST CONNECTOR RECORD
CLI QUASTRUSERTYP,X'A4'
BE LASTCON
CLI QUASTRUSERTYP,X'24'
BNE NORECCO NOT A CONNECTOR RECORD
LR R9,R7
AH R9,QUASTRUSERLEN NEXT RECORD POSITION

LASTCON DS ØH
MVC $CONAME,QUASTRUSERCNAME CONNECTOR NAME
MVC $COSYS,QUASTRUSERSYS SYSTEM NAME
MVC $COJOB,QUASTRUSERJOB JOB NAME
PRINTHX QUASTRUSERCONVERSION,$COVERS CONNECTION VERSION

* CONNECTOR STATUS
MVC $COST,$SPACES NOT SPECIFIED BY DEFAULT
MVCCASE TEST=QUASTRUSERFLG1,TO=$COST,TSTCOD=TM,FAILCOD=BNO, X
IF=(QUASTRUSERACT,QUASTRUSERFAIL,QUASTRUSERTERM, X
QUASTRUSERSUSC), X
MOVE=(=CL8"ACTIVE",=CL8"FAILED",=CL8"FAILING", X
=CL8"DISCONN")

* PRINT THE DATA
MVC RECORD,RECORDCO MOVE RECORD IN OUTPUT ZONE
BAL R1Ø,PUTREC PRINT THE DATA

NORECCO DS ØH
BR R8
DROP R7

*————————————————————————————————————————————————————————————————————*
* CF RECORD PROCESSING
* INPUT : R9 (RECORD ADDRESS)
* RETURNS R9=ADDRESS OF NEXT CF RECORD (=Ø IF NONE)
*————————————————————————————————————————————————————————————————————*

PROCESS_CF DS ØH
LR R7,R9
USING QUACF,R7 CF SECTION

* PREPARE NEXT RECORD
XR R9,R9 LAST CF RECORD
CLI QUACFRTYP,X'90'
BE LASTCF
CLI QUACFRTYP,X'10'
BNE NORECCF NOT A CF RECORD
LR R9,R7
AH R9,QUACFLEN NEXT RECORD POSITION

LASTCF DS ØH
MVC $CFN,QUACFNAME CF NAME
MVC $CFID,QUACFID CF ID

* DUMP SPACE SIZE
ICM R1,15,QUACFDUMPSIZE SIZE IS IN 4K MULTIPLE
MH R1=H'4096' NOW IN BYTES
XR RØ,RØ
D RØ.=F'1000' NOW IN K
STCM R1,15,QUACFDUMPSIZE SIZE IN KBYTES
EXTENDW QUACFDUMPSIZE,$CFSZ CF DUMP SIZE (MULT OF 4K BYTES)
MVI $CFSZ+5,'K' MENTION <K>-BYTES
MVC $CFSZ2,$SPACES CF STATUS
MVCCASE TEST=QUACFSTATE2,TSTCOD=TM,FAILCOD=BNO, X
TO=$CFST2,
IF=(QUACFSTRECONCILE,QUACFSTFAILED),
MOVE=(-CL9"RECONCILE",-CL9"FAILED")

EXTENDW QUACFSTR$,,$CFSTR  NUMBER OF RECS FOR STRUCT. IN CF
MVC  $CFCFRM,QUACFPOLNAME  CF CFRM POLICY
EXTENDW QUACFSC$,,$CFSYS     NUMBER OF RECS FOR SYST. CONNECTED

* PRINT THE DATA
MVC  RECORD,RECORDCF          MOVE RECORD IN OUTPUT ZONE
BAL  R10,PUTREC              PRINT THE DATA

NORECCF DS  ØH
BR   R8
DROP R7

* BUFFER CLEARING SUBROUTINES

CLRZONE L     R0,ADDZONE         BUFFER ADDRESS
L     R1,SIZE                BUFFER SIZE
XR    R2,R2                  R2=Ø
XR    R3,R3                  R3=Ø
MVCL  R0,R2                  ZEROIZE
BR   R8                     RETURN TO CALLER

CLRZONE2 L    R0,ADDZONE2        BUFFER ADDRESS
L     R1,SIZE                BUFFER SIZE
XR    R2,R2                  R2=Ø
XR    R3,R3                  R3=Ø
MVCL  R0,R2                  ZEROIZE
BR   R8                     RETURN TO CALLER

SYSPRINT DCB DDNAME=SYSPRINT,DSORG=PS,MACRF=PM,LRECL=133,RECFM=FB
SIZE     DC '200000'
ADDZONE  DC '0'
ADDZONE2 DC '0'
CFLEVEL  DC '0'
MAXSYS   DC '0'
CURRMAX  DC '0'
FEATURES DC '0'
SYSPLEXI DC '0'
RETURN   DC '0'
REASON   DC '0'
SAVE4    DS  4F
SAVE     DS  18F
RECORD   DC CL133' '

* REPORT TITLE
RECORDT DS ØCL133            TITLE RECORD
DC ' SYSPLEX CONFIGURATION ON ''
DATEØ DS CL10                CURRENT DATE
DC ' AT ''
TIMEØ DS CL9                 CURRENT TIME
DC ' - SYSTEM ''
SYSTNAME DC CL8' ''           CVTSNAME
DC ' ''
SYSTOS39 DS CL10             SYSTEM INFO
DC ' V ''

DC CL(133+RECORDG-*)' "

* MEMBER RECORD
RECORDM DS ØCL133 MEMBER RECORD
DC C' "
$MEMN DS CL16 MEMBER NAME
DC C' ACT ON/'
$MEMSYS DS CL8
DC C' JOINED/'
$MEMJOB DS CL8
DC C' "
$MEMST DS CL8 MEMBER STATUS
DC C' UPD:'
$MEMDATE DS CL10 LAST UPDATE DATE
DC C' "
$MEMTIME DS CL9 LAST UPDATE TIME
DC C' "
$MEMST2 DS CL18 MEMBER STATUS2
DC CL(133+RECORDM-*)' "

* CONNECTOR RECORD
RECORDCO DS ØCL133 CONNECTOR RECORD
DC C' - CONNECTION "
$CONAME DS CL16
DC C' "
$COST DS CL8 STATUS
DC C' SYSTEM='
$COSYS DS CL8
DC C' JOBN='
$COJOB DS CL8
DC C' VERS='
$COVERS DS CL9
DC C' "
DC CL(133+RECORDCO-*)' "

* STRUCTURE RECORD
RECORDST DS ØCL133 STRUCTURE RECORD
DC C' "
$STRN DS CL16
DC C' "
$STRINGCF DS CL5
DC C' POLICYSZ='
$STRSZ DS CL10
DC C' INITSZ='
*$STRSZIN DS CL10
DC C' CFRMPOL='
$STRCFRM DS CL8
DC C' "
$STRSTAT DS CL19
DC C' REBUILD:'
$STRSTRE DS CL17
DC CL(133+RECORDST-*)' "

* CF RECORD
RECORDCF DS ØCL133 CF RECORD
DC C' CF "

* WORK ZONES

WORD4 DS OF
TIME1 DS F HHMMSSTH
TIME2 DS F MIJU0000
DATE1 DS F DDMYYY
WORK6 DS CL6
DBLEWORD DS D DOUBLE WORD FOR INSTRUCTION CVD
ORG DBLEWORD+5
PACKED3 DS PL3
ORG DBLEWORD+3
PACKED5 DS PL5
HALFWORD DS H
MASK1 DC X'402020202121' MASK FOR PACKED LENGTH 3
MASK2 DC X'40202020202020202121' MASK FOR PACKED LENGTH 5
$SPACES DC CL133" 
$ZEROES DC X'0000000000000000'

LTORG
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15

IXCYQUAA, MAPS THE DATA RETURNED BY THE IXCQUERY MACRO

Thierry Falissard
ecic (France)

© Xephon 1999
GTF data analysis using SAS

INTRODUCTION

I had some SAS code which I inherited many years ago from a fellow systems programmer that was used to perform rudimentary editing and reformatting of GTF trace data. In getting ready to publish this code and having it reviewed by some fellow systems programmers, I found the original base code that it came from was part of the MXG software package from Merrill Associates, who were contacted and who agreed to assign copyright to me since the code is sufficiently different from the original Merrill code (see the comments at the beginning of the code for details).

I have kept working with it over the years because I sometimes find it difficult to use the output of the IPCS GTFTRACE command. Each individually traced event formatted by GTFTRACE occupies multiple lines of output, there is no interpretation of parameter list data that is part of SVC calls (except possibly ENQ and DEQ events, which do have their QNAMEs and RNAMEs formatted), and it is difficult to interpret the relationship between related events (such as the flow of events between an SVC interrupt and the redispatch of the program that issued the SVC). I have retained and enhanced the SAS code beyond the very basic information it originally formatted, and it has evolved to keep up with MVS/ESA and OS/390 operating systems.

The code formats information in a one line per event format, with almost all the fields that appear in most GTF trace records appearing as unique variables in the output SAS dataset (like the jobname, ASCB and TCB addresses, module name, event timestamp, and type). I have written the code to produce an intermediate dataset, and allowed a filtering option to create the intermediate dataset using selection criteria. The sample code is set up to extract only SVC interrupt events for ENQ and DEQ SVCs issued by jobs TSOJOBXX or TSOJOBYY between the times of 14:08:43.53 and 14:13:47.48 (expressed as GMT using the EVENTIME variable created, although I also create a local time value called LCLTIME which could just as easily be used) – see the label FILTER in the code for how this is done. The filtering statements could also be used to produce a full report of all the records contained in the input GTF trace dataset.
As for my sources of information, almost all of the record layouts come from currently available IBM manuals, including the *OS/390 MVS Diagnosis: Tools and Service Aids*, and the *OS/390 MVS Diagnosis: Reference*. The only record types which are not currently handled by this SAS code are CCW, SLIP, and RNIO trace records; these were not done because of their higher complexity because they tend to be used for highly-specialized purposes. I felt that, for those types, it was easier just to leave it to the GTFTRACE command. There are a few fields for the records I do handle which are not fully documented in the *Tools and Service Aids* manual. I therefore had to resort to reverse-engineering the data by formatting the data with the GTFTRACE command to determine its content and then locate the equivalent data in the raw GTF trace data in order to map it out via SAS code. IBM is hopefully formatting all of this undocumented data correctly (see below for an exception).

This SAS code does even better than the IPCS GTFTRACE command by formatting many pieces of information in human terms, such as reporting an event type (in SAS variable TYPE) as CLOCK-COMP, indicating a clock comparator interrupt, in addition to reporting the event name (in SAS variable EVENT) as EXT-INT=X1004. Another example is the reporting of SVC 56 (decimal) as an event name of SVC-38/056 (hex/decimal) and an event type of ENQ/RESV for enqueue/reserve, in addition to formatting and decoding the parameter list provided. Similarly for program interrupt (PI) events, the event name is shown as PGM-INT=017 (decimal) and the interrupt code is decoded as the event type, being shown as PAGE-XLATE for a page translation exception.

Interestingly enough, as a result of refining and testing this SAS code, I have recently come across a bug in the IPCS GTFTRACE formatting of PI events. It seems that, as far back as the MVS/XA version of PI event formatting, the GTFTRACE formatting code was incorrectly formatting the contents of general purpose register (GPR) 2. The model that controls the formatting of the GPRs was causing the GPR 6 contents to be formatted twice: once correctly as GPR 6 and once incorrectly as GPR 2. IBM has recently opened APAR OW39456 to fix this, at least at the OS/390 version 2.5 and above levels of code. I guess that there were very few people who ever needed to look at the 16 general purpose register contents during a program interrupt, and fewer that actually looked close enough to verify the data’s accuracy.
The other reason for creating an intermediate SAS dataset is to allow for the calculation of delta time differences between events after filtering has been done (in SAS variable DELTA). This is a personal preference that I chose, although some purists might think it more desirable to have the delta time difference calculated between each event traced as each record is read, whether or not the data is eventually removed by filtering. For example, in the sample code, since I only select ENQ and DEQ SVC interrupt records, the delta time difference is the difference between the issuance of the two SVCs in question. The input trace dataset, however, might contain recordings of all SVC interrupts for jobs TSOJOBXX and TSOJOBYY, or all SVCs issued for every job in the system during the time that the data was collected. The point is to keep in mind what options were used to collect the data as well as what filtering options were used within the SAS code, before making any assessment of the ‘goodness’ or ‘badness’ of delta times. If by chance you do not know what options were used to perform the data collection, the SAS code is also good enough to show that information as provided in the GTF trace dataset timestamp records (which also contain the local time zone offset permitting the calculation of the equivalent local time). Additional information contained in the timestamp record that is formatted by the SAS code also includes data about the system on which the GTF data collection occurred such as the MVS SCP release and FMID, the SMF system identifier, and the CPU serial number as returned by the Assembler language STIDP instruction.

I have chosen to format my own output report rather than use any standard SAS procedures (like PROC PRINT). This is because SAS likes to change the width of columns on each page of a SAS-generated report, which makes it difficult to scan down a listing visually. There are some commented lines in the code to use PROC PRINT to print the data if so desired. The manually formatted report is written out as fixed length 350 byte records, thus making it unprintable. It is rather intended for on-screen viewing via a product such as SDSF. The JCL wrapped around the SAS program below is, I suggest, given a catalogued procedure name of SAS; the JOBPARM specification and the WORK dataset allocation should be capable of handling most reasonably-sized input GTF trace datasets.
SOURCE

/*JOBPARM LINES=9999
//STEP1   EXEC SAS,SOUT='SYSOUT=*'
//WORK     DD SPACE=(CYL,(300,80))
//GTFIN    DD DISP=SHR,DSN=TSOBBOR.TRACE.OS39Ø
//GTFOUT   DD SYSOUT=* 
//SYSIN     DD * 
* THIS WORK IS A DERIVATIVE OF WORK ORIGINALLY COPYRIGHT (C):
* 1985 BY HERBERT W. BARRY MERRILL, D/B/A MERRILL CONSULTANTS;
* DALLAS, TX, USA, DISTRIBUTED AS PART OF MXG SOFTWARE, BUT;
* THIS DERIVATIVE IS SUFFICIENTLY DIFFERENT FROM THE ORIGINAL;
* THAT IT IS RECOGNIZED AS COPYRIGHT (C) 1999 XEPHON;
OPTIONS MISSING=' ' NOCENTER;* OBS=20000;
PROC FORMAT;
VALUE GTFEVENT (MIN=13 MAX=13)
    0000='PAGE-FAULT    0000'
    0001='SRB-DSP       0001'
    0002='LSR-DSP       0002'
    0003='TCB-DSP       0003'
    0004='SVC-DSP       0004'
    4096='SVC-INTRPT    1000'
    8448='PCI-INTRPT    2100'
   16385='SYSEVENT     4001'
   16386='E/STAE       4002'
   16387='FRR          4003'
   16388='SLIP-STD     4004'
   16389='SLIP+USER1   4005'
   16390='SLIP+USER2   4006'
   20736='SIO-INTRPT   5100'
   20737='EOS-SENSE    5101'
   20738='CSCH         5102'
   20739='HSCH         5103'
   20740='MSCH         5104'
   20741='SSCH         5105'
   20742='RSCH         5106'
   20992='I/O-INTRPT   5200'
   20993='CSI/O-INT    5201'
   24833='PGM-INT 1-17* 6101'
   25088='PGM-INT 18   6200'
   25089='EXT-INTRPT   6201'
   33024='RNIO TPIO1I/P 8100'
   33280='RNIO TPIO2O/P 8200'
   58865='PVM          E5F1'
   58868='NETVIEW-E5F4 E5F4'
   58869='NETVIEW-E5F5 E5F5'
   61252='RACF-EF44    EF44'
   61256='IOS-EF48     EF48'
   61257='BDT-EF49     EF49'
   61263='OSAM-EF4F    EF4F'
   61267='OSI-EF53     EF53'
   61268='FSI-EF54     EF54'
   61269='FSI-EF55     EF55'
61270='FSI-EF56' EF56'
61271='FSI-EF57' EF57'
61272='FSI-EF58' EF58'
61273='FSI-EF59' EF59'
61274='FSI-EF5A' EF5A'
61275='FSI-EF5B' EF5B'
61276='FSI-EF5C' EF5C'
61277='FSI-EF5D' EF5D'
61278='ESCON MGR' EF5E'
61279='DB2/VSAM XPRN' EF5F'
61280='JES3-EF60' EF60'
61281='VSAM-BURMGR' EF61'
61282='DYN-OUTPUT-SVC' EF62'
61283='CONV-INTRPRTR' EF63'
61285='GFS-TRACE' EF65'
61286='VTAM-EF66' EF66'
61287='VTAM-EF67' EF67'
61288='VTAM-EF68' EF68'
61289='VTAM-EF69' EF69'
61290='VTAM-EF6A' EF6A'
61291='VSM-EF6B' EF6B'
61292='CICS-EF6C' EF6C'
61344='TCAM-EFA0' EFA0'
61345='TCAM-EFAB' EFA1'
61346='TCAM-EFAC' EFA2'
61347='TCAM-EFAD' EFA3'
61348='TCAM-EFA4' EFA4'
61349='TCAM-EFA5' EFA5'
61350='TCAM-EFA6' EFA6'
61351='TCAM-EFA7' EFA7'
61352='TCAM-EFA8' EFA8'
61353='TCAM-EFA9' EFA9'
61359='CVAF-EFAF' EFAF'
61360='CVAF-EFB0' EFB0'
61361='CVAF-EFB1' EFB1'
61362='CVAF-EFB2' EFB2'
61363='CVAF-EFB3' EFB3'
61364='CVAF-EFB4' EFB4'
61365='CVAF-EFB5' EFB5'
61366='CVAF-EFB6' EFB6'
61367='CVAF-EFB7' EFB7'
61368='CVAF-EFB8' EFB8'
61369='DB2 TRACE' EFB9'
61384='DB2 IRLM-EFC8' EFC8'
61385='DB2 IRLM-EFC9' EFC9'
61409='VTAM-INTNL-VI' EFE1'
61410='VTAM-INTNL-TH' EFE2'
61411='VTAM-INTNL-TR' EFE3'
61412='VTAM-INTNL-RD' EFE4'
61413='JES2-EFE5' EFE5'
61414='JES2-EFE6' EFE6'
61415='JES2-EFE7' EFE7'
61416='JES2-EFE8' EFE8'
61417='JES2-EFE9    EFE9'
61418='JES2-EFEA    EFEA'
61419='JES2-EFEB    EFEB'
61420='JES2-EFEC    EFEC'
61421='JES2-EFED    EFED'
61422='JES2-EFEE    EFEE'
61423='VTAM-TPIO    EFEF'
61424='VTAM-BUFFER   EFF0'
61425='VTAM-CNTL    EFF1'
61426='VTAM-LNETRACE EFF2'
61427='SAM-IGGSP002 EFF3'
61428='SAM-IGGSP008 EFF4'
61429='SAM-IDAAM01   EFF5'
61430='SAM-IGGSP112 EFF6'
61431='SAM-IGGSP215 EFF7'
61432='SAM-IGGSP119 EFF8'
61433='SAM-IGGSP235 EFF9'
61434='SAM-IGGSP239 EFFA'
61435='SAM-IGGSP145 EFFB'
61436='SAM-IGGSP251 EFFC'
61437='SAM-IGGSP451 EFFD'
61438='SAM-IGGSP169 EFFE'
61439='O/C/EOV-TRACE EFFF'
61469='SVCDUMP       F100'
61952='ABDUMP        F200'
-1='TIME-STAMP      FFFF'

VALUE SVCNAME (MIN=10 MAX=10)
00='EXCP/XDAP      ' 01='WAIT/WAITR     '
02='POST         03='EXIT         '
04='GETMAIN      05='FREEMAIN     '
06='LINK/LINKX   07='XCTL/XCTLX   '
08='LOAD        09='DELETE      '
10='REGMAIN     11='TIME        '
12='SYNCH       13='ABEND       '
14='SPIE        15='ERREXCP     '
16='PURGE       17='RESTORE     '
18='BLDL/FIND   19='OPEN        '
20='CLOSE       21='STOW        '
22='OPEN/J      23='CLOSE/T     '
24='DEVTYPE     25='TRKBAI      '
26='CATTG/LOC   27='OBTAIN      '
28='SVCO28      29='SCRATCH     '
30='RENAME      31='FEOV        '
32='REALLOC     33='IOHALT      '
34='MGRQ/QEDIT  35='WTO/WTOR    '
36='WTL         37='SEGLD/SEGW  '
38='SVCO38      39='LABEL       '
40='EXTRACT     41='IDENTIFY   '
42='ATTACH     43='CIRB        '
44='CHAP        45='OVLYBRCH   '
46='TIMER       47='STIMER      '
48='DEQUEUE     49='SVCO49      '
50='SVCO50      51='SNAP/SDUMP   '
52='RESTART     53='REALEX      '
54='DISABLE     55='EOV         '
56='ENO/ESVR    57='FREEBUF     '
58='REL/REQBUF  59='OLTEP       '
60='E/STAE      61='IKJEG56A    '
62='DETACH     63='CHKPT       '
64='RDJFCB      65='SVCO65      '
66='BTAMTEST   67='SVCO67      '
68='SYNADAF/RL '
69='BSP        70='GSRV        '
71='ASGNBFV     72='IEAVCVR     '
73='SPAR       74='DAR         '
75='DQUEUE      76='IFBSVC76   '
77='SVCO77      78='LSPACE      '
79='STATUS     80='SVCO80      '
81='SETPRT     82='SVCO82      '
83='SMFWTM      84='GRAPHICS    '
85='DRSWAP     86='ATLAS       '
87='DOM        88='SVCO88      '
89='SVCO89     90='SVCO90      '
91='VOLSTAT    92='TCPXCP      '
93='TGET/TPUT  94='STCC        '
95='SYSEVENT   96='STAX        '
97='IKJEG59G   98='PROTECT     '
99='DYNALLOC   100='IKEFFIB   '  

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
VALUE PICCODE (MIN=10 MAX=10)

0001='OPERATION 0001' 0002='PRIV-OP 0002'
0003='EXECUTE 0003' 0004='PROTECTION 0004'
0005='ADDRESSING 0005' 0006='SPECIFIC 0006'
0007='DATA 0007' 0008='FIXED-OFLW 0008'
0009='FIXED-DIV 0009' 0010='DEC-OFLOWW 000A'
0011='DEC-DIVIDE 000B' 0012='EXP-OFLOW NN0C'
0013='EXP-UFLOW NN0D' 0014='SIGNIFICNC NN0E'
0015='FLPT-DIVID NN0F' 0016='SGMNT-XLAT 0010'
0017='PAGE-XLATE 0011' 0018='XLATE-SPEC 0012'
0019='SPECIAL-OP 0013'
0021='OPERAND 0015' 0022='TRACE-TABL 0016'
0023='ASN-XLATE 0017'
0025='VECTOR-OP 0019'
0028='SPACE-SW 001C'
0029='UNNORMLIZD NN1E'
0031='PC-XLATE 001F'
0033='ASX-XLATE 0021'
0035='EX-XLATE 0023'
0037='SECN-D-AUTH 0025'
0040='ALET-SPEC 0028'
0041='ALEN-XLATE 0029'
0043='ASTE-VALID 002B'
0045='XTND-AUTH 002D'
0049='STACK-MPTY 0031'
0051='STACK-TYPE 0033'
0054='STACK-FULL 0030'
0050='STACK-SPEC 0032'
0052='STACK-OP 0034'
0055='MONITOR-EV 0040'
0128='PER-EVENT 0080'

OTHER='??????????? ????'

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
VALUE $SRBTYPE (MIN=10 MAX=10)
   'I'='INIT-DISP'        'G'='GLOBAL'
   'L'='LOCAL'           'S'='REDISPATCH'

VALUE XINTTYP (MIN=10 MAX=10)
   Ø64='IRPT-KEY'    Ø4Ø=4Ø99='CLOCK-SYNC 1ØØ3'
   41Ø0='CLOCK-COMP 1ØØ4'   41Ø1='CPU-TIMER 1ØØ5'
   46Ø8='MF-ALERT 12ØØ'   46Ø9='EMER-SIGNL 12Ø1'
   461Ø='EXT-CALL 12Ø2'   5126='ETR-EVENT 14Ø6'
   9217='SERV-SIGNL 24Ø1'

DATA GTF (KEEP=EVENT EVENTIME LCLTIME PSW VARDATA REGDATA
   ASCB RØ R1 R15 TCB CPU JOB MODULE TYPE);
   FORMAT EVENTIME LCLTIME                              DATETIME23.6
       VARDATA GTFOPTS $2ØØ.
       REGDATA    $4Ø.
       PSW        $16.
       EVENT     $13.
       TYPE      $10.
       ASCB TCB JOB MODULE $8.
       CPU       $4.
   RETAIN LCLTZOFF;
   LENGTH DEFAULT=4 EVENTIME LCLTIME 8;
   INFILE GTFIN MISSOVER;

***** MAINLINE CHECKING *****;
INPUT @1 AID      PIB1. @;
IF AID=ØØØ THEN DO;
   LINK CONTROL;
   LINK FILTER;
   RETURN;
END;
IF AID=255 THEN DO;
   LINK EVENTREC;
   LINK FILTER;
   RETURN;
END;
ELSE DO;
   PUT AID HEX2. ' UNEXPECTED FLAG ';
   LIST;
   RETURN;
END;

***** OPTIONAL FILTERING CRITERIA *****;
FILTER:
*IF JOB = 'TSOTCONC';
*IF EVENT=:'SVC ';
*IF TYPE=:'ENQ' OR TYPE=:'DEQ';
*IF ('14:0Ø:43.53'T<=TIMEPART(EVENTIME)<=14:13:47.48'T); **OS/39Ø GMT;
*IF ('13:09:31.63'T<=TIMEPART(EVENTIME)<='13:12:05.46'T); **MVS522 GMT;
RETURN;

***** GTF CONTROL RECORD *****;
CONTROL:
INPUT @2 FID     PIB1. @7 EVENTIME TODSTAMP8. @;
IF FID=Ø THEN DO;
   EVENT='SAVEHOOK';
   TYPE='SAVEHOOK';
END;
IF FID=1 THEN DO;
   EVENT='TIMESTAMP';
   TYPE='TIMESTAMP';
   INPUT @3 TIMEZONE IB4. @15 (GTFOPT1-GTFOPT8) (PIB1.)
   @31 SCPREL $8. @39 SCPFMID $8. @47 SYSNAME $8.
   @55 PRCSRID PIB6. @;
   LCLTZOFF=ROUND((TIMEZONE*1.048576));
   IF GTFOPT1 = '1.......'B THEN GTFOPTS=TRIM(GTFOPTS)||'SYSM,';
   IF GTFOPT1 = '.1......'B THEN GTFOPTS=TRIM(GTFOPTS)||'SYSP,';
   IF GTFOPT1 = '..1.....'B THEN GTFOPTS=TRIM(GTFOPTS)||'USR,';
   IF GTFOPT1 = '...1....'B THEN GTFOPTS=TRIM(GTFOPTS)||'TRC,';
   IF GTFOPT1 = '....1...'B THEN GTFOPTS=TRIM(GTFOPTS)||'DSP,';
   IF GTFOPT1 = '.....1..'B THEN GTFOPTS=TRIM(GTFOPTS)||'PCI,';
   IF GTFOPT1 = '.......1'B THEN GTFOPTS=TRIM(GTFOPTS)||'SVCP,';
   IF GTFOPT2 = '.1......'B THEN GTFOPTS=TRIM(GTFOPTS)||'SIO,';
   IF GTFOPT2 = '1.......'B THEN GTFOPTS=TRIM(GTFOPTS)||'SIOP,';
   IF GTFOPT2 = '.1......'B THEN GTFOPTS=TRIM(GTFOPTS)||'PI,';
   IF GTFOPT2 = '1.......'B THEN GTFOPTS=TRIM(GTFOPTS)||'PI,';
   IF GTFOPT2 = '.1......'B THEN GTFOPTS=TRIM(GTFOPTS)||'IO,';
   IF GTFOPT2 = '1.......'B THEN GTFOPTS=TRIM(GTFOPTS)||'IOP,';
   IF GTFOPT3 = '1.......'B THEN GTFOPTS=TRIM(GTFOPTS)||'EXT,';
   IF GTFOPT3 = '.1......'B THEN GTFOPTS=TRIM(GTFOPTS)||'SMR,';
   IF GTFOPT3 = '1.......'B THEN GTFOPTS=TRIM(GTFOPTS)||'SMR,';
   IF GTFOPT4 = '.1......'B THEN GTFOPTS=TRIM(GTFOPTS)||'SLIP,';
   IF GTFOPT4 = '1.......'B THEN GTFOPTS=TRIM(GTFOPTS)||'CCW,';
   IF GTFOPT4 = '..1.....'B THEN GTFOPTS=TRIM(GTFOPTS)||'CCWP,';
   IF GTFOPT4 = '1.......'B THEN GTFOPTS=TRIM(GTFOPTS)||'ISIO,';
   IF GTFOPT5 = '1.......'B THEN GTFOPTS=TRIM(GTFOPTS)||'ISSCH,';
OPTLEN=LENGTH(GTFOPTS);
VARDATA='SCP REL='||TRIM(SCPREL)||' FMID='||TRIM(SCPFMID)||
' SYSDNAME='||TRIM(SYSNAME)||' CPUID='||PUT(PRCSRID,HEX12.)||
' OPTIONS='||SUBSTR(GTFOPTS,2,OPTLEN-2);
END;
IF FID=2 THEN DO;
EVENT='LOSTEVENT';
INPUT @15 COUNT PIB4. @;
TYPE='CNT='||PUT(COUNT,COMMA6.);
END;
IF FID=3 THEN DO;
EVENT='LOSTBLOCK';
INPUT @15 COUNT PIB4. @;
TYPE='CNT='||PUT(COUNT,COMMA6.);
END;
IF FID=5 THEN DO;
EVENT='FULLBUFF';
TYPE='FULLBUFF';
END;
LCLTIME=EVENTIME+LCLTZOFF;
RETURN;

***** GTF EVENT RECORD *****;
EVENTREC:
INPUT  @2 FID      PIB1.  @3 EVENTIME TODSTAMP8. @11 EID     PIB2.
     @13 ASCBADDR PIB4. @17 CPUID   PIB2.  @19 JOB     $8.
     @27 PSWA     PIB4. @31 PSWB    PIB4.      @35 TCBADDR PIB4. @;
PSW=PUT(PSWA,HEX8.)||PUT(PSWB,HEX8.);
CPU=PUT(CPUID,HEX4.);
ASCB=PUT(ASCBADDR,HEX8.);
TCB=PUT(TCBADDR,HEX8.);
EVENT=PUT(EID,GTFEVENT.);
LCLTIME=EVENTIME+LCLTZOFF;
VARDATA=' ';
REGDATA=' ';
TYPE=' ';

***** SVC EVENT *****;
IF EVENT=':SVC-INTRPT' THEN DO;
INPUT @3Ø SVCNM   PIB1. @39 MODULE $8.
     @47 GPR15   PIB4. @51 GPRØ    PIB4. @55 GPR1    PIB4. @;
TYPE=PUT(SVCNM,SVCNAME.);
EVENT=':SVC-'||PUT(SVCNM,HEX2.)||'/'||PUT(SVCNM,Z3.);
R15=PUT(GPR15,HEX8.);
RØ=PUT(GPRØ,HEX8.);
R1=PUT(GPR1,HEX8.);
REGDATA='R15='||R15||' RØ='||RØ||' R1='||R1;

***** SPECIAL SVC PROCESSING *****;
IF TYPE=':ENQ/RESV' OR TYPE=':DEQUEUE ' THEN DO;
IF TYPE=':ENQ/RESV' THEN DO;
INPUT  @59 (PLIST1-PLIST9) (PIB4.)
     @81 ENQFLGS PIB1. @95 QNAME $8. @1Ø3 RNLEN PIB1.
     @1Ø4 RNAME $VARYING2ØØ. RNLEN  @;

PLIST='PLIST='||PUT(PLIST1,HEX8.)||
' '|PUT(PLIST2,HEX8.)||
' '|PUT(PLIST3,HEX8.)||
' '|PUT(PLIST4,HEX8.)||
' '|PUT(PLIST5,HEX8.)||
' '|PUT(PLIST6,HEX8.)||
' '|PUT(PLIST7,HEX8.)||
' '|PUT(PLIST8,HEX8.)||
' '|PUT(PLIST9,HEX8.);
END;

IF TYPE=:'DEQUEUE' THEN DO;
  INPUT @59 (PLIST1-PLIST5) (PIB4.)
    @65 ENQFLGS PIB1. @79 QNAME $8. @87 RNLEN PIB1.
    @88 RNAME $VARYING200. RNLEN @;
  PLIST='PLIST='||PUT(PLIST1,HEX8.)||
' '|PUT(PLIST2,HEX8.)||
' '|PUT(PLIST3,HEX8.)||
' '|PUT(PLIST4,HEX8.)||
' '|PUT(PLIST5,HEX8.);
END;

IF ENQFLGS = '.0..0...B THEN SCOPE='STEP ';
IF ENQFLGS = '.0..1....B THEN SCOPE='RESERVE';
IF ENQFLGS = '.1..0....B THEN SCOPE='SYSTEM ';
IF ENQFLGS = '.1..1....B THEN SCOPE='SYSTEMS';
IF ENQFLGS = '......000'B THEN RET='NONE';
IF ENQFLGS = '......001'B THEN RET='HAVE';
IF ENQFLGS = '......010'B THEN RET='CHNG';
IF ENQFLGS = '......011'B THEN RET='USE ';
IF ENQFLGS = '......100'B THEN RET='ECB*';
IF ENQFLGS = '......110'B THEN RET='TEST';
  REQ='EXC';
IF ENQFLGS = '1......B THEN REQ='SHR';
VARDATA=TRIM(PLIST)||
  ' FLGS='||PUT(ENQFLGS,HEX2.)||
  ' SCOPE='||TRIM(SCOPE)||
  ' REQ='||TRIM(REQ)||
  ' RET='||TRIM(RET)||
  ' QNAME='||TRIM(QNAME)||
  ' RNAME='||TRIM(RNAME);
RETURN;
END;

IF TYPE=:'EXCP/XDAP' OR
  TYPE=:'EXCPVR'
  THEN DO;
  INPUT @59 DDNAME $8. @67 DCBADDR PIB4. @71 DEBADDR PIB4. @;
  VARDATA='DDNAME='||TRIM(DDNAME)||
  ' DCB='||PUT(DCBADDR,HEX8.)||
  ' DEB='||PUT(DEBADDR,HEX8.)||
  ' '||REGDATA;
  REGDATA='';
  RETURN;
END;
IF TYPE=:'PURGE' THEN DO;
  INPUT @59 DDNAME $8. @67 DCBADDR PIB4. @71 PLIST1 PIB4. @75 PLIST2 PIB4. @79 PLIST3 PIB4. @;
  VARDATA='DDNAME='||TRIM(DDNAME)||
    ' DCB='||PUT(DCBADDR,HEX8.)||
    ' PPL='||TRIM(PUT(PLIST1,HEX8.))||
    ' '||PUT(PLIST2,HEX8.)||
    ' '||PUT(PLIST3,HEX8.))||
    ' '||REGDATA;
  REGDATA='';
  RETURN;
END;

IF TYPE=:'WAIT/WAITR' OR TYPE=:'POST' OR
TYPE=:'FREEMAIN' OR
TYPE=:'OPEN' OR
TYPE=:'OPEN/J' OR
TYPE=:'ATTACH' OR
TYPE=:'WTO/WTOR' OR
TYPE=:'RDJFCB' OR
TYPE=:'DYNALLOC'
THEN DO;
  INPUT @59 PLISTLN PIB1. @6Ø (PLIST1-PLIST1Ø) (PIB4.) @;
  IF PLISTLN GT Ø THEN DO;
    VARDATA=TRIM('PLIST='||PUT(PLIST1,HEX8.)||
      ' '||PUT(PLIST2,HEX8.)||
      ' '||PUT(PLIST3,HEX8.))||
      ' '||REGDATA;
  END;
  LINK REGTOVAR;
  RETURN;
END;

IF TYPE=:'XCTL/XCTXL' OR
TYPE=:'LINK/LINKX'
THEN DO;
  INPUT @59 EPDENAME $8. @67 PLIST1 PIB4. @71 PLIST2 PIB4. @75 PLIST3 PIB4. @;
  VARDATA='EP/DE='||TRIM(EPDENAME)||
    ' PLIST='||TRIM(PUT(PLIST1,HEX8.))||
    ' '||PUT(PLIST2,HEX8.)||
    ' '||PUT(PLIST3,HEX8.))||
An MPF command exit

INTRODUCTION
The following utility is an MPF command exit which is invoked for each command that is issued from any console or MCS console attached to, or in, the system. By default, it takes no action unless ‘D RES’ is issued. It then changes the command to ‘D GRS,C’ and invokes GQSCAN to report on all outstanding reserves on the system on which it was issued.

For further information see the MVS installation exits, and the section on MPFLSTXX in *MVS Initialization and Tuning reference*.

MPFXIT

**MPFXIT**

**TITLE 'COMMAND EXIT - PROCESS D RES COMMAND'**

******************************************************************************

* ON ENTRY: REGISTER CONTENTS *
* Ø NOT APPLICABLE *
* 1 ADDRESS OF THE POINTER TO THE CMDX (IEZVX1Ø1) *
* 2 -- 12 NOT APPLICABLE *
* 13 REGISTER SAVE AREA *
* 14 RETURN ADDRESS *
* 15 ENTRY POINT ADDRESS OF THE EXIT ROUTINE *
* *
* USAGE: REGISTER CONTENTS *
* 5 NUMBER OF RIBES TO PROCESS *
* 6 POINTER TO RIBES *
* 7 POINTER TO RIBS *
* 8 NUMBER OF RIBES TO PROCESS *
* 9 FOR BRANCH AND SAVE SUBROUTINE CALL *
* Ø0 POINTER TO CMDXCLIB OBTAINED FROM CMDXCLIP *
* 11 POINTER TO CMDX *
* 12 BASE *
* 13 ADDRESS OF OBTAINED STORAGE (WORKSTOR) & RSA *

******************************************************************************

* SAMPLE DISPLAY:
* ---------------
* Ø....+....1....+....2....+....3....+....4....+....5....+....6....+....

*OUTSTANDING RESERVES*
* QNAME=SYSIGGV2 RNAME=USERCAT.SYX.HURON SCOPE=SYSTEM LOCAL
* OWNING TASKS=ØØ1 TASKS WAITING FOR EXCLUSIVE=ØØ0 FOR SHARE=ØØ0
* DEV JOBNAME ASID SYSTEM STAT TYPE CONV SMC RSV (REQUESTOR)
* ØØ3A CATALOG Ø039 DEVI OWN SHR NO NO YES ABCD1234/NNNN

******************************************************************************
MPFXIT   START
MPFXIT   AMODE 31
MPFXIT   RMODE ANY
USING *,R12

BAKR  R14,Ø              CREATE LINKAGE STACK ENTRY
LR    R12,R15
L     R11,Ø(R1)         POINTER TO CMDX
USING CMDX,R11
L     R1Ø,CMDXCLIP       POINTER TO COMMAND LENGTH AND BUFFER
USING CMDXCLIB,R1Ø
CLC   CMDXCMDI(11),=CL11'DISPLAY RES'
BE    PROCES
CLC   CMDXCMDI(5),=CL5'D RES'
BNE   EXITNOW

* OBTAIN WORKING STORAGE WHICH CONTAINS RSA AND CLEAR IT

PROCES   STORAGE OBTAIN,LENGTH=WORKLEN
LR    R13,R1
USING WORKSTOR,R13
LA    R2,WORKSTOR
LH    R3,=AL2(WORKLEN)
LA    R4,WORKSTOR
XR    R5,R5
MVCL  R2,R4              CLEAR OBTAINED STORAGE
MVC   SAVEAREA+4(4),=C'F1SA' INDICATE BAKR WAS USED

* CHANGE COMMAND TO 'D GRS,C' AND CLEAR WTO TEXT AND INIT WTO TEXT LEN
MVC   CMDXCMDI(11),=CL11'D GRS,C,L=Z' NEW CMD TO BE PROCESSED
MVC   CMDXCMDL(2),=H'11' NEW LENGTH
OI    CMDXRFL1,CMDXRCMI INDICATE NEW COMMAND
MVC   WTTXTL(2),=AL2(L'WTTXT) LENGTH OF WTO - NEVER CHANGES
MVI   WTTXT,C' '         CLEAR WTO TEXT WITH SMEAR
MVC   WTTXT+1(L'WTTXT-1),WTTXT

* CHECK IF COMMAND WAS ISSUED FROM MCS CONSOLE - IF IT IS NOT, THEN
* IT COMES FROM EITHER INTERNAL (I/ ON SPF) OR INSTREAM FROM JES2 IN
* WHICH CASE IT SHOULD JUST WTO THE MESSAGE WITHOUT SPECIFYING CONSOLE
* NAME
MVC   WTCONSNM(8),CMDXCNNM
TM    CMDXSTU1,CMDXFMCS WAS COMMAND ISSUED FROM MCS CONSOLE?
BO    CNVBEGIN           YES - GO FOR IT
MVC   WTCONSNM(8),=CL8' '  
B     CNVDONE

* SEARCH FOR FIRST AVAILABLE OUT-OF-LINE DISPLAY AREA ON CONSOLE
CNVBEGIN LA    R9,CVPARM
USING CONV,R9
LA    R8,=C' ABCDEFGHIJZ' LIST OF VALID AREAS'
MVI   WTAREAID,C'Z'    PRIME AREAID TO Z - THE DEFAULT
CNVSTRT LA    R8,1(R8) ADVANCE PTR TO NEXT AREAID
CLI   Ø(R8),C'Z'      IS NO OUT-OF-LINE AREA AVAILABLE ?
BE    CNVDONE          NO - USE Z
XC    CONV(CONVGLEN),CONV CLEAR CONVCON PARAMETER AREA
MVC   CONVACRO,=C'CONV' REQUIRED EYECATCHER
MVI   CONVRSN,CONVRID CURRENT VERSION LEVEL
MVI   CONVFLD,C' '    SMEAR CONVFLD WITH BLANKS
MVC CONVFLD+1(L'CONVFLD-1),CONVFLD
MVC CONVFLD(8),CMDXCNNM CONSOLE NAME
LA R2,CONVFLD-1 SEARCH FOR FIRST BLANK
CNVBLP LA R2,1(R2) IN CONVFLD TO
CLI Ø(R2),C' ' INSERT AREAID
BNE CNVBLP
MVI Ø(R2),C'-'
MVC 1(1,R2),Ø(R8) PUT AREAID AFTER C'-' IN CONVFLD
MVC CONVAREA(1),Ø(R8) PUT AREAID IN CONVAREA TO BE SURE
OI CONVFGLS,CONVFPLD FLAG FOR PROCESSING DESIRED

CNVDOIT CONVCON (R9)
LTR R15,R15 WAS RC=Ø ?
BNZ CNVDONE NO - USE AREAID=Z
CLI CONVRSN,X'00' IS AREAID AVAILABLE ?
BE CNVZERO YES - USE IT
CLI CONVRSN,X'10' WAS THERE A SYNTAX TERROR ?
BNE CNVSTRT NO - DO AGAIN
MVC Ø(1,R2),1(R2) REMOVE C'-' FROM CONSOLE NAME BY
MVI 1(R2),C' ' MOVING AREAID OVER IT
B CNVDOIT REISSUE CONVCON

CNVZERO EQU *
MVC WTAREAID(1),CONVAREA PLACE AREAID IN WORKSTOR

CNVDONE EQU *
DROP R9

* ISSUE FIRST WTO TO ORIGINATING CONSOLE AND SET UP CONNECT FOR WTOS
MVC WTLST1(SWTLEN1),SWTLST1 COPY WTO LIST FORM TO WORKSTOR
MVC WTLST1+SWTAID1(1),WTAREAID PLACE AREAID IN LIST FORM
WTO TEXT=((WTFIRL,)),LINKAGE=BRANCH,CONSNAME=WTCONSNM, X
CART=CMDXCART,MF=(E,WTLST1)
ST R1,WTCONNEC
MVC WTLST2(SWTLEN2),SWTLST2 COPY WTO LIST FORM TO WORKSTOR
MVC WTLST2+SWTAID2(1),WTAREAID PLACE AREAID IN LIST FORM

* CLEAR FLAG AND ISSUE GQSCAN
DOITGQSC NI GQFLAG,X'FF'-GQMOFIT
MVC GQSTDYN(GQSTLEN),GQLST
GQSCAN AREA=(RIBSTOR,L'RIBSTOR),SCOPE=ALL,RESERVE=YES, X
TOKEN=GQTOKEN,MF=(E,GQSTDYN)
ST R8,GBKREGØ STORE LENGTHS OF RIB AND RIBE
ST R1,GBKNUMRIB STORE NUMBER OF RIBS RETURNED
LTR R15,R15 ARE THERE OUTSTANDING RESERVES ?
BZ PROCRIBS YES - PROCESS RIBS
CH R15,=H'4' ARE THERE OUTSTANDING RESERVES ?
BE NORESERV NO - ISSUE MESSAGE
CH R15,=H'8' ARE THERE MORE OUTSTANDING RESERVES ?
BE MOREOFIT YES - MORE RESERVES

* UNKNOWN RETURN CODE - WTO IT
MVC WTTXT(18),=CL18'GQSCAN SERVICE RC='
ST R15,WORK
UNPK WTTXT+18(9),WORK(5)
MVZ WTTXT+18(8),ZEROS
TR WTTXT+18(8),TRTAB
MVI WTTXT+26,C' '

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
* ALL OUTSTANDING RESERVES DID NOT FIT INTO RIBSTOR
MOREOFIT IO GQFLAG,GQMOFIT INDICATE MORE CALLS TO GQSCAN
*

* PROCESS RIBS IN RIBSTOR
PROC RIBS L R8,GQNUMRIB GET NUMBER OF RIBS
* SOMETIMES GQSCAN RETURNS RC=Ø ALTHOUGH NO RIBS WERE RETURNED!
LTR R8,R8 SØC4 PROTECTION
BZ CHEERS SØC4 PROTECTION
* SOMETIMES GQSCAN RETURNS RC=Ø ALTHOUGH NO RIBS WERE RETURNED!
LA R7,RIBSTOR
USING RIB,R7
TM GQFLAG,GQBIBTR PREVIOUS GQSCAN RIBES TRUNCATED?
BZ PROCØØØ NO - CONTINUE
NI GQFLAG,X'FF'-GQBIBTR
B PROC3ØØ PROCESS ONLY RIBES REMAINING
PROCØØØ EQU *
MVC WTTXT+1(6),=CL6'QNAME='
MVC WTTXT+7(8),RIBQNAME
LH R6,GQRIBL
AR R6,R7 POINT TO VARIABLE PORTION OF RIB USING RIBVAR,R6
MVC WTTXT+16(6),=CL6'RNAME='
XR R5,R5
IC R5,RIBRNMLN
LTR R5,R5 MOST UNLIKELY TO HAVE RNAME LEN OF Ø
BZ OVEREX1 BUT JUST IN CASE...
CH R5,=H'27'
BNH PROCØ5Ø
LA R5,27
PROCØ5Ø BCTR R5,₀ ADJUST TO MACHINE LENGTH OF RNAME
EX R5,EX1
B OVEREX1
EX1 MVC WTTXT+22(1),RIBRNAME
DROP R6
OVEREX1 MVC WTTXT+50(6),=CL6'SCOPE='
TM RIBSCOPE,RIBSYS
BZ PROC1ØØ
MVC WTTXT+56(6),=CL6'SYSTEM'
PROC1ØØ TM RIBSCOPE,RIBSYSS
BZ PROC11Ø
MVC WTTXT+56(7),=CL7'SYSTEMS'
PROC11Ø TM RIBSCOPE,RIBSTEP
BZ PROC12Ø
MVC WTTXT+56(4),=CL4'STEP'
PROC12Ø MVC WTTXT+64(5),=CL5'LOCAL'
TM RIBSCOPE,RIBGLBL
BZ PROC13Ø
MVC WTTXT+64(6),=CL6'GLOBAL'
PROC13Ø BAS R9,WTOIT
MVC WTTXT+4(13),=CL13'OWNING TASKS='
L R5,RIBNTO
* START PROCESSING THE RIBES HANGING FROM THIS RIB

PROC3ØØ LH R5,RIBVLEN
AR R6,R5
USING RIBE,R6
L R5,RIBNRIBE NUMBER OF RIBES
C R5,RIBTRIBE IS TOTAL AMOUNT OF RIBES RETURNED ?
BE PROC4ØØ
OI GQFLAG,GQRIBTR INDICATE THAT NOT ALL RIBES PRESENT
PROC4ØØ MVC WTTXT+33(3),=CL3'OWN'
MVC WTTXT+33(3),=CL3'SHR'
TM RIBERFLG,RIBETYPE
BO PROC45Ø
PROC45Ø MVC WTTXT+33(3),=CL3'EXC'
TM RIBERFLG,RIBERESC
BZ PROC46Ø
PROC46Ø MVC WTTXT+33(3),=CL3'NO'
TM RIBERFLG,RIBERESC
BZ PROC47Ø
MVC WTTXT+43(3),=CL3'YES'
PROC470 MVC WTTXT+48(2),=CL2'NO'
TM RIBERFLG, RIBEMC
BZ PROC480
MVC WTTXT+48(3),=CL3'YES'
PROC480 MVC WTTXT+52(2),=CL2'NO'
TM RIBERFLG, RIBERESV
BZ PROC490
MVC WTTXT+52(3),=CL3'YES'
PROC490 TM RIBERFLG, RIBESIDV
BZ PROC495
CLC RIBESAID(2),ZEROS
BE PROC495
UNPK WTTXT+66(5),RIBESAID(3) *RELIRES ON WTTXT OVERFLOW BYTE*
MVZ WTTXT+66(4),ZEROS
TR WTTXT+66(4),TRTAB
MVI WTTXT+65,'C'/'
LA R4,Ø ADDRESS OF PSA
USING PSA,R4
L R4,FLCCVT ADDRESS OF CVT
DROP R4
USING CVT,R4
CLC CVTSNAME(8), RIBESYN
BNE PROC495
L R4,CVTASVT ADDRESS OF ASVT
DROP R4
USING ASVT,R4
LH R3,RIBESAID ASID
SLL R3,2 MULTIPLY BY 4 TO GET OFFSET OF ASCB
L R4,ASVFRST(R3) ADDRESS OF ASCB
DROP R4
ST R4,WORK
TM WORK, ASVTAVAL ASID AVAILABLE ?
BO PROC495 YES
USING ASCB,R4
CLC ASCBASID(2), RIBESAID
BNE PROC495
L R4, ASCBASSB ADDRESS OF ASSB
DROP R4
USING ASSB,R4
MVC WTTXT+57(8), ASSBjbni JOBNAME
CLC ASSBjbni(8), ZEROS
BNE PROC495
MVC WTTXT+57(8), ASSBjnbs START/MOUNT/LOGON NAME
DROP R4
PROC495 BAS R9, WTOIT
AH R6, GQRIBEL POINT TO NEXT RIBE
BCT R5, PROC42Ø PROCESS NEXT RIBE
DROP R6
LR R7, R6 POINT TO NEXT RIB
BCT R8, PROC00Ø PROCESS NEXT RIB
TM GQFLAG, GQMOSIT DONE ?
BO DOITGQSC DO ANOTHER GQSCAN
B CHEERS
* NO RESERVES
NORESERV MVC WTTXT(36),=-CL36'NO OUTSTANDING RESERVES AT THIS TIME'
BAS R9,WTOIT
* ISSUE FINAL WTO AND RELEASE WORKING STORAGE
CHEERS MVC WTLSTE(SWTLENE),SWTLSTE COPY WTO LIST FORM TO WORKSTOR
MVC WTLSTE+SWTAIDE(1),WTAREAID PLACE AREAIID IN LIST FORM
WTO TEXT=((WTENDL,)),LINKAGE=BRANCH,CONNECT=WTCONEC,
X CART=CMDXCART,MF=(E,WTLSTE)
LR R1,R13
STORAGE RELEASE,LENGTH=WORKLEN,ADDR=(R1)
EXITNOW XR R15,R15
** WTO SUBROUTINE **
WTOIT WTO TEXT=((WTTXTL,)),LINKAGE=BRANCH,CONNECT=WTCONEC,
X CART=CMDXCART,MF=(E,WTLST2)
MVI WTTXT,C'
CLEAR WTO TEXT WITH SMEAR
MVC WTTXT+1(L'WTTXT-1),WTTXT
BR R9
****************************************************
LTORG
TRTAB DC C'Ø123456789ABCDEF'
GQLST GQSCAN AREA=(Ø,L'RIBSTOR),SCOPE=ALL,RESERVE=YES,
X TOKEN=Ø,MF=L
GQLSTLEN EQU *-GQLST
* WTO LIST FORMAT FOR FIRST WTO OF MULTI LINE WTO
SWTLST1 WTO TEXT=((,L)),LINKAGE=,CONSNAME=,CART=,
AREAID=Z,DESC=(8,9),MF=L
SWTLLEN1 EQU *-SWTLST1
SWTAID1 EQU SWTLLEN1-2 OFFSET FROM SWTLST1 TO AREAID
* WTO LIST FORMAT FOR SUBSEQUENT Wtos
SWTLST2 WTO TEXT=((,D)),LINKAGE=,CONNECT=,CART=,
AREAID=Z,DESC=(8,9),MF=L
SWTLLEN2 EQU *-SWTLST2
SWTAID2 EQU SWTLLEN2-2 OFFSET FROM SWTLST2 TO AREAID
* WTO LIST FORMAT FOR LAST WTO OF MULTI LINE WTO
SWTLSTE WTO TEXT=((,DE)),LINKAGE=,CONNECT=,CART=,
AREAID=Z,DESC=(8,9),MF=L
SWTLLEN EQU *-SWTLSTE
SWTAIDE EQU SWTLLEN-2 OFFSET FROM SWTLSTE TO AREAID
DS OH
WTFLRL DC AL2(L'WTFLRT) MESSAGE LENGTH OF FIRST MESSAGE
WTFLRT DC C'OUTSTANDING RESERVES'
DS OH
WTENDL DC AL2(L'WTENDT) MESSAGE LENGTH OF LAST MESSAGE
WTENDT DC C'END OF DISPLAY'
*************** MAPPING DSECT OF WORKING STORAGE OBTAINED
WORKSTOR DSECT
SAVEAREA DS 18F REGISTER SAVE AREA (FOR GQSCAN)
ZEROS DS XL8 8 BYTES OF ZERO - FOR MVZ
WORK DS XL8 8 BYTES MISC WORKING STORAGE
CVPARM DS XL(CONVGLEN) PARAMETER AREA FOR CONVCON SERVICE
GQTOKEN DS F TOKEN USED BY GQSCAN SERVICE
GQNUMRIB DS F           NUMBER OF RIBS RETURNED BY GQSCAN
GQREG0 DS OF           LENGTH OF RIB AND RIBE IN REG0
GQRIBL DS H            LENGTH OF RIB
GQRIBEL DS H           LENGTH OF RIBE
GQFLAG DS X            A FLAG
GQMODFIT EQU X'80'     ISSUE GQSCAN MORE THAN ONCE
GQRI8TR EQU X'40'      RIB WAS TRUNCATED - RE-ISSUE GQSCAN
WTCONSNM DS CL8        CONSOLE NAME FROM CMDXCNNM
WTCONNEC DS F          CONNECT FIELD FOR SUBSEQUENT WTOS
WTAREAID DS C          AREAID FROM CONVCON MACRO FOR WTOS
WTTXTL DS H            WTO TEXT LENGTH
WTTXT DS CL70          WTO TEXT
                     DS C          WTO TEXT OVERFLOW BYTE FOR UNPK
WTLST1 DS XL(SWTLEN1)  WTO LIST FORM STORAGE
                     ORG WTLST1
WTLST2 DS XL(SWTLEN2)  WTO LIST FORM STORAGE
                     ORG WTLST1
WTLSTE DS XL(SWTLENE)  WTO LIST FORM STORAGE
                     ORG
GQLSTDYN DS XL(GQLSTLEN)DS OF
RIBSTOR DS XL2000      RIB AND RIBE STORAGE
WORKLEN EQU *-WORKSTOR LENGTH OF DYNAMIC OBTAINED AREA
***************
IEZVX101
IEZVGT200
ISGRIB
IHAPSA
CVT DSECT=YES
IHAASVT
IHAASCB
IHAASSB
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
END

Systems Programmer (UK) © Xephon 1999
**Inter-address space access program**

**INTRODUCTION**

The following program allows you to extract information from address spaces using inter-address space communication. This sample code extracts only the program names executing in JES initiators, but you can enhance it to collect as much information as you need. Using this method, you could even modify data in another address space.

The next main activities performed to achieve inter-address space communication are:

1. TESTAUTH SVC – verify whether the APF bit is set indicating authority to switch to supervisor mode and key=0.
2. MODESET SVC – switch to supervisor mode and key=0.
3. GETMAIN SVC – obtain a storage area in the fixed common SQA for a monitor routine, an SRB, and a working area for passing data between address spaces.
4. Copy a monitor routine from our private area into the SQA making it executable for all address spaces.
5. STIMER SVC – set a timer with a 10-second interval in order to assume control if the object address space does not respond to our request (time-out).
6. Schedule an SRB to activate the monitor routine as an independent task in the object address space.
7. WAIT SVC – wait for completion of the monitoring task.

After some time, the SRB task is activated in the object address space. This task collects the data requested and stores it in the SQA work area. On completion of this operation, a cross-memory post operation is scheduled to awaken our address space, which continues with:

1. TTIMER SVC – cancel the timer for the 10-second time-out.
2. Copy the collected data from the SQA work area into our private address space.
3  FREEMAIN SVC – free the areas for the monitor routine, SRB, and work area in the SQA.

4  MODESET SVC – return to problem-program mode and key=8.

PROGRAM SOURCE

TITLE 'X> X> GETS PROGRAM NAMES FROM JES INITIATORS X> X>,'
PRINT NOGEN
* *************************************************************************
* CFOCPGM  - GETS THE PROGRAM NAMES CURRENTLY RUNNING IN JES
* INITIATORS.
* *************************************************************************
CFOCPGM  CSECT
* ** PROLOG CODE - ESTABLISH ADDRESSABILITY
STM   R14,R12,12(R13)
LR    R12,R15
USING CFOCPGM,R12
ST    R13,SAVEAREA+4
LR    R2,R13
LA    R13,SAVEAREA
ST    R13,8(R2)
* ** ASK IF PROGRAM IS APF-AUTHORIZED
TESTAUTH FCTN=1
LTR   R15,R15
BZ    MODESET
WTO   '** PROGRAM IS NOT AUTHORIZED **'
B     RETURN
* ** SWITCH TO SUPERVISOR MODE, KEY Ø
MODESET  MODESET MODE=SUP,KEY=ZERO
* ** MAKE THE CURRENT ADDRESS SPACE NON-SWAPPABLE
XR    R1,R1
SYSEVENT TRANSWAP
* ** GET STORAGE AREA IN FIXED COMMON SQA (SUBPOOL 245)
GETMAIN RU,LV=WRKALEN,SP=245
LR    R6,R1
USING WRKAREA,R6
MVI   WRKAREA,X'ØØ'
MVC   WRKAREA+1(WRKALEN-1),WRKAREA
* ** GET THE CURRENT ASCBID
L     R5,CVTPTR  R5 -> CVT
L     R5,Ø(.R5)  R5 -> PSATNEW
L     R5,12(.R5)  R5 -> CURRENT ASCB
ST    R5,XMASCB  STORE THE ADDRESS (IT WILL
*     BE USED BY XMPOST)
* ** COPY THE SRB ROUTINE INTO COMMON STORAGE
LA    R2,SRBCSTG
LA    R3,ENDSRBTN-SRBRTN
LA    R4,SRBRTN
LR   R5,R3
MVCL R2,R4

* ** SCAN BATCH ADDRESS SPACES
L   R10,CVTPTTR   R10 -> CVT  
L   R10,CVTASVT-CVTMAP(.R10) R10 -> ASVT  
USING ASVT,R10  
XR   R9,R9

NEXTASID L   R8,ASVTENTY(R9) R8 -> ASCB  
USING ASCB,R8  
C   R8,XMASCB  
BE   NXTASVT  
CLM   R8,B'1000',=X'08' ASVT ENTRY IN USE?  
BNE   EXASCB  
CLM   R8,B'0111',=XL3'0' END OF ASVT?  
BE   ENDOFSCN

NXTASVT LA   R9,4(R9) ASVT ENTRY UNUSED.  
B   NXTASID  

EXASCB L   R7,ASCBJBNI JES2 INITIATED?  
LTR   R7,R7  
BTC   NXTASVT NO. GET ANOTHER ASVT ENTRY.  
MVC   JOBNAME(8),Ø(R7) SAVE JOB NAME.

* ** INITIALIZE THE SRB
LA   R7,SRBAREA  
USING SRB,R7  
MVC   SRBID,=CL4'SRB'  
ST   R8,SRBASC1B  
MVC   TCBADDR,ASCBTNEW  
MVC   SRBPASID,ASCBASEID  
LA   R1,SRBCSTG  
ST   R1,SRBEP  
NI   SRBEP,X'7F'  
LA   R1,SRBCLEAN-SRBRTN(.R1)
ST   R1,SRBRMTR  
NI   SRBRMTR,X'7F'  
LA   R1,PARMLIST  
ST   R1,SRBPARM  
LA   R1,WAITEC1B  
ST   R1,HELPEC1B

* ** SET A TIMER WITH A 10-SECONDS INTERVAL, IN ORDER TO ASSUME CONTROL  
* ** CONTROL IF OBJECT ADDRESS SPACE DOES NOT RESPOND.  
STIMER REAL,TIMEOUT,DINTVL=TENSECS

* ** SCHEDULE THE SRB
SCHEDULE SRB=(R7),SCOPE=LOCAL,LLOCK=YES

* ** WAIT FOR COMPLETION OF THE MONITORING TASK
WAIT ECB=WAITEC1B

* ** CANCEL PREVIOUS STIMER IF A TIME-OUT DID NOT EXIST
TTIMER T
LTR   R0,R0  
BTC   MOVEPGM  
TTIMER CANCEL
** MOVE THE PROGRAM NAME TO OUTPUT AREA AND GO TO GET THE**

** NEXT ASVT ENTRY.**

MOVEPGM MVC PROGNAME,PGMNAME
WTO MF=(E,WTOMSG)
XC WAITECB,WAITECB
XC SRBAREA,SRBAREA
B NXTASVT GET NEXT ASVT ENTRY

** FREEMAIN THE WORKAREAS, SWITCH TO PROBLEM PROGRAM MODE AND**

** RETURN TO CALLER.**

ENDOFSCN FREEMAIN RU,LV=WRKALEN,A=(6),SP=245
MODESET MODE=PROB,KEY=NZERO
RETURN L R13,SAVEAREA+4
RETURN (14,12),RC=Ø

** TIME-OUT ROUTINE**

TIMEOUT SAVE (14,12)
USING TIMEOUT,R15
L R2,HELPECB
POST (2)
RETURN (14,12)

** WORK AREAS**

SAVEAREA DC 18F'Ø' SAVE AREA
HELPECB DC F'Ø'
TENSECS DS D
ORG TENSECS
DC CL8'ØØØØ1ØØØ' TEN SECOND INTERVAL
WTOMSG WTO '** JOBNAME=XXXXXXXX — PGMNAME=XXXXXXXX',MF=L
JOBNAME EQU WTOMSG+15,8
PROGNAME EQU WTOMSG+35,8
LTORG *

*—————————————————————————————————————————————————————————————————————*
*                     S R B    R O U T I N E                          *
*—————————————————————————————————————————————————————————————————————*

SRBRTN LR R12,R15 LOAD BASE REGISTER
USING SRBRTN,R12
LR R4,R1 LOAD PARMLIST ADDRESS
USING PARMSRB,R4
LR R5,R14 SAVE RETURN ADDRESS
L R3,TCBSRB R3 -> OBJECT TCB
L R3,TCBJSCB-TCB(R3) R3 -> OBJECT JSCB
MVC PGMSRB,JSCBPGMN-IEZJSCB(R3) MOVE OBJECT PROG NAME
L R6,ASCURB
POST ECBSRB,ASCB=(R6),ECBKEY=Ø, CROSS MEMORY POST X
LINKAGE=SYSTEM,ERRET=POSTERR
SETLOCK RELEASE,TYPE=LOCAL
BR R5 ENDS SRB ROUTINE
POSTERR BR R14
SRBCLEAN XC Ø(SRBSIZE,R1),Ø(R1)
BR R14
ENDSRTN EQU *

** SRB WORK AREA
WRKAREA DSECT
SRBAREA DS CL(SRBSIZE) SRB STORAGE
PARMLIST DS 0XL2Ø PARM LIST TO SRB ROUTINE
TCBADDR DS F TCB ADDRESS
WAITECB DS F WAIT ECB
XMASCB DS F ASCB ADDRESS FOR XMPOST
PGMNAME DS CL8 PROGRAM NAME
SRBCSTG DS CL(ENDSRTN-SRBRTN)
WRKALEN EQU *-WRKAREA

* ** PARMLIST DSECT
PARMSRB DSECT SAME AS PARMLIST IN WRKAREA
TCBSRB DS F
ECBSRB DS F
ASCSRB DS F
PGMSRB DS CL8

* ** CONTROL BLOCKS DSECTS
PRINT OFF
CVT DSECT=YES CONTROL VECTOR TABLE
IHAASVT ADDRESS SPACE VECTOR TABLE
IHAASCB ADDRESS SPACE CONTROL BLOCK
IHAASXB ADDRESS SPACE EXTENSION BLOCK
IKJTCB DSECT=YES,LIST=NO TASK CONTROL BLOCK
IEZJSCB JOB STEP CONTROL BLOCK
IHASRB SRB DSECT
IHAPSA PSA DSECT
REGEQU
END

Enrique Garcia
Banco de Credito del Peru (Peru)  © Xephon 1999
An IPL subsystem (part 4)

This month we continue our look at the Initial Program Load Subsystem which reduces the errors inherent in the manual typing and entering of system commands required to activate on-line systems.

```assembly
WTORWTOR DS (PPGLENWR)C
WTORANS DS C
   DS 0F
WTORSVC DS H,H
WTORCMG DS CL20
   SPACE
PPGDOUBL DS D
   SPACE 1
WORKLEN EQU */-WORK
   SPACE 1
PATASID EQU COMMNDWK+ENTLEN-4
   SPACE 1
PATVTAB EQU COMMNDWK+ENTLEN-8
   SPACE 1
PATVCNT EQU COMMNDWK+ENTLEN-12
   TITLE 'ID''S OF SUPPORTED SYSTEMS + MANDATORY NCP CHANNEL ADR'
*******************************************************************
*        CHANNELS REQUIRED TO BE ON-LINE BEFORE NET IS STARTED    *
*******************************************************************
   SPACE 1
DCIPLS CSECT ,
   DS 0F
PPGSYSTM DC CL4'VSØ1',A(PPGVSØ1)
PPGSYSIZ EQU (*-PPGSYSTM)
   DC CL4'VSØ2',A(PPGVSØ2)
   DC CL4'VSØ3',A(PPGVSØ3)
   DC CL4'VSØ4',A(PPGVSØ4)
   DC CL4'VSØ5',A(PPGVSØ5)
PPGSYSNT EQU (*/-PPGSYSTM)/PPGSYSIZ
   SPACE 1
PPGVØ1 DC XL2'0602'
   DC C'N'
   DC XL2'0605'
   DC C'N'
   DC XL2'0606'
   DC C'N'
   DC C'T' 
   SPACE 1
PPGVØ2 DC XL2'0602'
   DC C'N'
   DC XL2'0606'
   DC C'N'
```

DC C' '
SPACE 1
PPGVSO3 DC XL2'0602'
DC C'N'
DC XL2'0606'
DC C'N'
DC C' '
SPACE 1
PPGVSO4 DC XL2'0602'
DC C'N'
DC XL2'0603'
DC C'N'
DC XL2'0605'
DC C'N'
DC XL2'0606'
DC C'N'
DC C' '
SPACE 1
PPGVSO5 DC XL2'0602'
DC C'N'
PPGUCBLN EQU *-PPGVSO5
DC XL2'0603'
DC C'N'
DC XL2'0605'
DC C'N'
DC XL2'0606'
DC C'N'
DC C' '
SPACE 1
DS 0F
PPGWTOR WTOR 'DCIP01A DEVICE PPHG IS NOT AVAILABLE; REPLY ''Y'' TO PROCEED WITH OR ''N'' TO TERMINATE ACTIVATION PROCESS X ',,...,MF=L
PPGLENWR EQU *-PPGWTOR
SPACE 1
DS 0F
PPGWTORD WTOR 'DCIP07A 37X5'S WERE OFFLINE WHEN NET WAS STARTED; REPLY ''Y'' TO PROCEED WITH OR ''N'' TO TERMINATE ACTIVATION PROCESS',,...,MF=L
SPACE 1
PPGLENW D EQU *-PPGWTORD
SPACE 1
DS 0F
SVCPARM DC H'18',H'0'
CLAIRE DC CL20'V PPHG,ONLINE '
SPACE 1
PPGSVCL EQU *-SVCPARM
TITLE 'ESA CONTROL BLOCKS'
***********************************************************************
* GENERATE REQUIRED OS CONTROL BLOCKS
***********************************************************************
SPACE 1
PUSH PRINT
PRINT NOGEN
SPACE 1
IKJTCB TASK CONTROL BLOCK
SPACE 1
IHAXTLST EXTENT LIST
SPACE 1
IHARB REQUEST BLOCK
SPACE 1
IHACDE CONTENTS DIRECTORY ENTRY
SPACE 2
YREGS
SPACE 2
PRINT NOGEN
SPACE 1
IHAPSA ESA PREFIX SAVE AREA
SPACE 1
DCBD DSORG=PS
SPACE 1
CVT DSECT=YES COMMUNICATIONS VECTOR TABLE
SPACE 1
IEFJESCT JES CONTROL TABLE
SPACE 1
IEFJSCVT SUBSYSTEM COMMUNICATIONS VECTOR TABLE
SPACE 1
IHASDWA SYSTEM DIAGNOSTIC WORK AREA
SPACE 1
CSCB IEECHAIN COMMAND SCHEDULER CONTROL BLOCK
SPACE 1
IRAOUCB SRM USER CONTROL BLOCK
SPACE 1
IHAASCB ADDRESS SPACE CONTROL BLOCK
SPACE 1
IHAASVT ADDRESS SPACE VECTOR TABLE
SPACE 1
IEESMCA
SPACE 1
IEECUCM
SPACE 1
IHAORE
SPACE 1
IHAMQE
SPACE 1
IHADVCT
SPACE 1
IEFUCBOB
EJECT
IEFJSSVT SUBSYSTEM COMMUNICATION VECTOR TABLE
SPACE 1
POP PRINT
SPACE 2
***********************************************************************
*        SUPPORTED FUNCTIONS                                          *
***********************************************************************
SPACE 1
FUNCTION EQU SSVTFCOD-1  
ORG FUNCTION+10  
SSVTCMDS DS X  
COMMAND BROADCAST FUNCTION FIELD  
ORG,  
SPACE 2
***********************************************************************
*        SSVT USER EXTENSION                                          *
***********************************************************************
SPACE 1
SSVTANKR DS F  
COMMAND TABLE POINTER  
SSVTECB DS F  
SUBSYSTEM ADDRESS SPACE ECB  
SSVTASCB DS F  
SUBSYSTEM ADDRESS SPACE ASCB POINTER  
SSVTCMDQ DS C  
SUBSYSTEM COMMAND IDENTIFIER  
DS C  
DS ØD  
SSVTLEN EQU *-SSVTBEGN  
TITLE ' NOTES REGARDING THE COMMAND TABLE.'
***********************************************************************
*   THE ANKR POINTS TO THE FREE, ALLOCATED, AND SPIN WORDS  
*   LOCATED AT THE BEGINING OF THE TABLE.  
*   
*   -  THE FREE AND ALLOCATED WORDS POINT TO A CHAIN OF ENTRIES  
*      WITHIN THE TABLE. EACH ENTRY POINTS TO THE NEXT TABLE  
*      ENTRY CONTAINED ON THAT PARTICULAR QUEUE. THE END OF  
*      THE QUEUE IS DENOTED BY A ZERO CHAIN POINTER.  
*   
*   -  THE SPIN WORD IS USED TO MAINTAIN TABLE INTEGRITY IN THE  
*      EVENT THAT A DP, MP, OR AP IS BEING USED.  
*   
*   -  NEITHER THE ACTUAL NUMBER OF TABLE ENTRIES NOR THE ACTUAL  
*      TABLE ENTRY LENGTHS ARE SHOWN IN THE ABOVE EXAMPLE.  
*      BOTH THE ALLOCATED AND FREE QUEUES ARE PUSH DOWN (LIFO)  
*      STACKS.  
***********************************************************************
EJECT
***********************************************************************
*   ANKR  *————*  
*         * TABLE *  
*         * POINTER*——>  
*         * X'1000'*  
*         *——*—*  
*                          |  
*          X'1000' TABLE *———*———*———*  
*                         *FREE *ALLOC *SPIN  
*                         *X'2000'*X'2300' *Ø*  
*                         *———*———*———*  
*                         |  

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
DCIPLSRB

TITLE 'SERVICE REQUEST ROUTINE (SRB)'

SPACE 1

DCIPLSRB CSECT ,

DCIPLSRB AMODE 24

DCIPLSRB RMODE 24

SPACE 1

***********************************************************************

SPACE

END

***********************************************************************

DCIPLSRB

TITLE 'SERVICE REQUEST ROUTINE (SRB)'

SPACE 1

DCIPLSRB CSECT ,

DCIPLSRB AMODE 24

DCIPLSRB RMODE 24

SPACE 1

***********************************************************************

* REGISTER USAGE: *

* *

* RØ - SRB POINTER UPON ENTRY *

* R1 - POINT TO USER FIELD
* R3 - CSECT BASE REGISTER
* R9 - RETURN ADDRESS (SET FROM ORIGINAL CONTENTS OF R14)
* R14 - RETURN ADDRESS UPON ENTRY
* R15 - ENTRY POINT ADDRESS UPON ENTRY


***********************************************************************
SPACE 1
LR    R9,R14          PRESERVE R14 FOR RETURN TO ESA
SPACE 1
LR    R3,R15          PRIME SRB BASE REGISTER
SPACE 1
USING DCIPLSRB,R3    ESTABLISH DCIPLSRB ADDRESSABILITY
SPACE 1
LM    R5,R8,Ø(R1)    RETRIEVE SRB PARAMETERS
EJECT

***********************************************************************
* POST TWO ECBS:
* POST THE SUBSYSTEM ADDRESS SPACE ROUTINE,
* IT RESIDES IN THE CURRENT ADDRESS SPACE.
* CROSS MEMORY POST THE SUBSYSTEM FUNCTION ROUTINE.
* OBTAIN THE LOCAL LOCK FOR THE CROSS MEMORY POST.
* REGISTERS 5 THROUGH 8 CONTAIN THE NECESSARY ECB AND ASCB POINTERS AS FOLLOWS:
* R5 - SUBSYSTEM ADDRESS SPACE ROUTINE'S ECB POINTER
* R6 - SUBSYSTEM ADDRESS SPACE ROUTINE'S ASCB POINTER
* R7 - SUBSYSTEM FUNCTION ROUTINE'S ECB POINTER
* R8 - SUBSYSTEM FUNCTION ROUTINE'S ASCB POINTER

***********************************************************************
SPACE 1
SETLOCK OBTAIN,TYPE=LOCAL,MODE=UNCOND,REGS=USE,RELATED=POST
SPACE 1
POST (5),Ø,LINKAGE=BRANCH NUDGE SUBSYSTEM ADDR SPACE ROUTINE
EJECT
POST (7),Ø,ASCB=(8),ERRET=ERRET,LINKAGE=BRANCH & FUNCTION RTN
SPACE 1
SETLOCK RELEASE,TYPE=LOCAL,REGS=USE,RELATED=POST
SPACE 1
LR    R14,R9          RELOAD R14 FOR RETURN TO ESA
BR    R14             BACK TO DUST
SPACE 1
ERRET    BR    R14     POST ERROR ROUTINE
TITLE 'DCIPLSRB - ESA CONTROL BLOCKS AND EQUATES'
YREGS
SPACE 2
PRINT NOGEN
SPACE 1
SYS1.COMMANDS and SYS1.CURLIST have the following traits:

RECFM=FB, LRECL=80, BLKSIZE=3120, DSORG=PO.

MEMBERS IN SYS1.COMMANDS

An operator will enter \(?CONNECT\) in response to the DCIPL01A message after an IPL of the PRO domain. \(?CONNECT\) is the third and last command in a series of commands that an operator directs to DCIPLS after he IPLs a mainframe – the first being \(?PROUP\), the second \(?VERIFY\) – to ensure that required applications are active. The following members in SYS1.COMMANDS are accessed by DCIPLS.

**CONNEPRO**

CONNEPRO contains modify commands for on-line applications and HSM.

```
F I1IM1S,EXEC EMVTAM01
F I2IM2S,EXEC EMVTAM02
F CICPTOR1,CEMT SET VTAM OPEN
F CICRTO1,CEMT SET VTAM OPEN
F CICPAOD1,CEMT SET VTAM OPEN
F CICPAOD2,CEMT SET VTAM OPEN
F CICPAOD2,CEMT SET CON(SSA1) ACQ
F HSMC,RELEASE ALL
```

**CONWTORS**

CONWTORS contains a response to IMS that notifies it that VTAM is once again active and restablishes communications with it.

```
DFS996I /STA DC
DFS996I /STA DC
DFS996I /STA DC
```

**CONWTORP**

CONWTORP contains a response that enables IMS1 to reestablish its link with IMS6.

```
DFS996I /RST LINK 2
```
CURLIST

CURLIST contains the current operand for the LIST= option that is used to initiate network activity. DCIPLS, in this instance, would issue a S NET,,,(LIST=A1) command. This must be properly maintained!

A1 CURRENT LIST FOR DCIPLES 09/04/98

DCIPWARN

DCIPWARN, a member in SYS1.COMMANDS, contains the modify command that DCIPLS uses to broadcast a warning message to active users of a ROSCOE system. A command is generated internally that issues the same warning message to TSO users. It is referenced whenever DCIPLS encounters a ?WARN message... command. message... is affixed to the tail-end of the modify command that is issued.

ROSCO F ROSCO,SEND ALL,
ROSCOEB F ROSCOEB,SEND ALL,
ROSCOEC F ROSCOEC,SEND ALL,
ROSCOED F ROSCOED,SEND ALL,
TROSCO F TROSCO,SEND ALL,

The following members in SYS1.COMMANDS are accessed by DCIPLS when an operator enters ?PRODOWN. This readies PRO for a reload of the NCP.

NETWTOR1

NETWTOR1 contains responses for WTORs that will cause normal cessation of an application’s activities by either terminating them or severing their communications link with VTAM.

CHF0011ACLOSE CANCEL
DSI802A CLOSE IMMED
DSI803A CLOSE IMMED
DSI804I TERMINATE
DSI805I TERMINATE
U11D-600F

NETWTOR2

NETWTOR2 has an identical purpose in life.

IKT010D FSTOP
U11D-600F
PRODWNET

PRODWNET contains operator commands that will terminate applications that cannot tolerate an outage of VTAM, or must be informed that it will be out-of-service for awhile. ?VERIFYDW is used afterwards to confirm that these applications are indeed dormant.

F SAMS,STOP
C TELE005A
C HOAVS05
V NET,INACT,ID=HCFAPPL,F
V NET,INACT,ID=INFOAPPL,F
V NET,INACT,ID=INFOAPL1,F
V NET,INACT,ID=INFOAPL4,F
C APPC
C ASCH
P OMRTA2
P EDAPRDS1
P EDAPRDS2
P PHOENIX
S DRAINRMT
P TMONVS05
P TMVSDL53
P TMVSMST3
P CICSMONP
P CICSDLSP
P DB2MONP
P DB2DLSP
P TSOC
P PHOENIX
P NETNPM
P NDMTN
F JCLARCH,STOP
P OMRTA
F RO SCOEC, SHUTDOWN NOW
F XCOM, STOP, IMMED
F EDM, SHUTDOWN
P PHOENIX
-STOP DDF

The following members in SYS1.COMMANDS are accessed by DCIPLS when an operator enters ?PAP. This readies a system for an IPL of it.

Members of SYS1.COMMANDS have been listed and described in alphabetical order however, DCIPLS processes them as follows: PAPRESOR, PAPWTOR1, PAPMODIF, PAPPCANC, PAPWTOR2. Afterwards, beginning at the statement labelled PROBLEMS, other applications such as HSM and ThruPut Manager are terminated. This
is also the area where a check for tasks executing in performance group seventeen is made.

**PAPMODIF1**

PAPMODIF contains modify commands relevant to the orderly termination of an application’s activities. Columns 1 - 8 contain a task’s identifier, columns 9 - 39 contain an operator command that will gracefully end it. Task identifiers must be alphabetized.

<table>
<thead>
<tr>
<th>Task</th>
<th>Operator Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C NOTHING (PAPMODIF AND PAPPCANC MUST BE KEPT IN A-ORDER)</td>
</tr>
<tr>
<td>BMCP</td>
<td>BMCP SHUTDOWN</td>
</tr>
<tr>
<td>BULL</td>
<td>V NET,INACT,ID=JANUS,F</td>
</tr>
<tr>
<td>BULL2</td>
<td>V NET,INACT,ID=JANUS2,F</td>
</tr>
<tr>
<td>CAL7P</td>
<td>CA11 SHUTDOWN ALL</td>
</tr>
<tr>
<td>CICATOR1F</td>
<td>CICATOR1,CEMT PER SHUT</td>
</tr>
<tr>
<td>CICPAOR1F</td>
<td>CICPAOR1,CEMT PER SHUT</td>
</tr>
<tr>
<td>CICPAOR2F</td>
<td>CICPAOR2,CEMT PER SHUT</td>
</tr>
<tr>
<td>CICPAOR3F</td>
<td>CICPAOR3,CEMT PER SHUT</td>
</tr>
<tr>
<td>CICPAOR4F</td>
<td>CICPAOR4,CEMT PER SHUT</td>
</tr>
<tr>
<td>CICPAOR5F</td>
<td>CICPAOR5,CEMT PER SHUT</td>
</tr>
<tr>
<td>CICPTOR1F</td>
<td>CICPTOR1,CEMT PER SHUT</td>
</tr>
<tr>
<td>CICRAOR1F</td>
<td>CICRAOR1,CEMT PER SHUT</td>
</tr>
<tr>
<td>CICRTO1F</td>
<td>CICRTO1,CEMT PER SHUT</td>
</tr>
<tr>
<td>CICSAOR1F</td>
<td>CICSAOR1,CEMT PER SHUT</td>
</tr>
<tr>
<td>CITCTOR1F</td>
<td>CITCTOR1,CEMT PER SHUT</td>
</tr>
<tr>
<td>CITCTOR1F</td>
<td>CITCTOR1,CEMT PER SHUT</td>
</tr>
<tr>
<td>CITCTOR2F</td>
<td>CITCTOR2,CEMT PER SHUT</td>
</tr>
<tr>
<td>CITCTOR1F</td>
<td>CITCTOR1,CEMT PER SHUT</td>
</tr>
<tr>
<td>DBUSS</td>
<td>Z DBUSS</td>
</tr>
<tr>
<td>DB2MSTR</td>
<td>/STOP DB2,MODE(FORCE)</td>
</tr>
<tr>
<td>DB2PMSTR</td>
<td>/STOP DB2,MODE(FORCE)</td>
</tr>
<tr>
<td>DB2SMSTR</td>
<td>/STOP DB2,MODE(FORCE)</td>
</tr>
<tr>
<td>DB2TMSTR</td>
<td>/STOP DB2,MODE(FORCE)</td>
</tr>
<tr>
<td>DEVAAFX</td>
<td>F DEVAAFX,SHUT 2Ø</td>
</tr>
<tr>
<td>EDM</td>
<td>F EDM,SHUTDOWN</td>
</tr>
<tr>
<td>HCF</td>
<td>F HCF,CLOSE IMMED</td>
</tr>
<tr>
<td>HSMA</td>
<td>F HSMA,STOP</td>
</tr>
<tr>
<td>HSMC</td>
<td>F HSMC,STOP</td>
</tr>
<tr>
<td>HSMC</td>
<td>F HSMC,STOP</td>
</tr>
<tr>
<td>ICOM</td>
<td>F ICOM,STOP</td>
</tr>
<tr>
<td>INFOPAC</td>
<td>V NET,INACT,ID=INFOAPPL,F</td>
</tr>
<tr>
<td>INFOPAC1V</td>
<td>NET,INACT,ID=INFOAPPL1,F</td>
</tr>
<tr>
<td>INFOPAC4V</td>
<td>NET,INACT,ID=INFOAPPL4,F</td>
</tr>
<tr>
<td>JCLARCH</td>
<td>F JCLARCH,STOP,IMM</td>
</tr>
<tr>
<td>PRODAAFX</td>
<td>PRODAAFX,SHUT 2Ø</td>
</tr>
<tr>
<td>RAC2</td>
<td>RAC2 STOP</td>
</tr>
<tr>
<td>ROSCOE</td>
<td>F ROSCOE,SHUTDOWN,NOW</td>
</tr>
<tr>
<td>ROSCOE</td>
<td>F ROSCOE,SHUTDOWN,NOW</td>
</tr>
<tr>
<td>ROSCOEC</td>
<td>F ROSCOEC,SHUTDOWN,NOW</td>
</tr>
</tbody>
</table>
ROSCOE F ROSCOED, SHUTDOWN, NOW
ROSCOE F ROSCOEE, SHUTDOWN, NOW
SPMS2ØØØ@DOWN CLEAR
TROSCOE F TROSCOE, SHUTDOWN, NOW
ULTRAOPTZ BMCU
VAM F VAM, REMOVE
XCOM F XCOM, STOP, IMMED
XCOMTST F XCOMTST, STOP, IMMED
XOSFØØ F XD13, SYSTEM, SHUTDOWN
ZZZZZZZ C NOTHING (PAPMODIF AND PAPPCANC MUST BE KEPT IN A-ORDER)

PAPPCANC

PAPPCANC takes a more brutal approach – it issues a CANCEL command for many program products, a STOP for others. Column 1 contains an appropriate operator command, the name of a task begins in column 3. Tasknames are alphabetized, of course.

A
C NOTHING (PAPMODIF AND PAPPCANC MUST BE KEPT IN A-ORDER)
C ALLCCICS
C ALLCIMS1
C ALLCIMS2
C ALLCIMS4
C ALLCIMS6
C ALLCPROD
P AOF1ISSI
C APAF
P APAF4
C APPC
C ASCH
P BMCLINK
P CICSDLSD
P CICSDLSM
P CICSDLSP
P CICSMOND
P CICSMONM
P CICSMONP
P CNMPSSI
P DB2AMV
P DB2DLSD
P DB2DLSP
C DB2MOND
C DB2MONP
C DMG
P EDADEVSR
P EDADEVS2
P EDAPRD51
P EDAPRDS2
P HGLINK
C HOAVSØ5
P XMON
C XPROC
C ZZZZZZZZ (PAPMODIF AND PAPPCANC MUST BE KEPT IN A-ORDER)

PAPRESOR
PAPRESOR contains imperious commands that are unequivocally issued.

$PI1-68
$PPRT1,PRT2
$PPRT3,PRT4
$PPRT5,PRT6
$PPRT7,PRT8
$PPRT13,PRT14,PRT16
S DRAINRMT

PAPWTOR1
PAPWTOR1 contains responses to outstanding WTORs that will initiate termination of IMS and NetView and software products from Computer Associates.

DF996I /CHE DUMPQ
DS1Ø02A CLOSE IMMED
DS1Ø03A CLOSE IMMED
DS1Ø04A TERMINATE
DS1Ø05A TERMINATE
U11D-Ø45Y
U11D-6Ø0F
U11D-7Ø0F

PAPWTOR2
PAPWTOR2 contains responses to WTORs that are used to terminate other applications.

ARCØ055ACANCEL
ARCØ055ACANCEL
ARCØ381ACANCEL
BMC1693 STOP
BMC247Ø1N
DFS996I /CHE DUMPQ
DS1Ø04A TERMINATE
DS1Ø05A TERMINATE
IKTØ1ØD FSTOP
PROUPNET

The following members in SYS1.COMMANDS are accessed by DCIPLS when an operator enters ?PROUP in response to a DCIPL01A message. If he responds with ?VERIFYUP or ?VERIFYDW, then DCIPLS uses only PROUPNET. PROUPNET contains the names of applications that are to be activated immediately after an IPL (or NCP reload) as well as the operator commands that are used to activate them. Columns 1-29 contain operator commands, column 30 contains the identifier of a monitored task that must be active, column 38 contains the character ‘P’ if the command is for a software product that must be inactive before the command to start it is issued, otherwise the command is unequivocally issued, and column 39 contains the character ‘Q’ if it is to be inactivated before the NCP is reloaded.

S APPC, SUB=MSTR, APPC=ØØ APPC PQ
S ASCH, SUB=MSTR, ASCH=ØØ ASCH PQ
S ULTRAOPT ULTRAOPTP
V NET, ACT, ID=HCFAPPL
V NET, ACT, ID=INFOAPPL
V NET, ACT, ID=INFOAPL1
V NET, ACT, ID=INFOAPL4
S CNMPSSI CNMPSSI P
S SYNCDSM1 SYNCDSM1P
S EDAPRDS1 EDAPRDS1PQ
S EDAPRDS2 EDAPRDS2PQ
S PRODAAFX PRODAAFXP
S CICSMONP CICSMONPPQ
S HOAVSØ5 HOAVSØ5 PQ
S CNMPNVØ5 CNMPNVØ5PQ
S VAM VAM P
S IRLM4 IRLM4 P
S DB2DLSP DB2DLSP P
S DB2MONP DB2MONP P
S RMTSNA
S INFOPAC4, TIME=1440 INFOPAC4PQ
S INFOPACV, TIME=1440 INFOPACVPQ
S NDMTN NDMTN PQ
S OMRTA OMRTA PQ
S OMRTA2 OMRTA2 PQ
S TSOC TSOC PQ
S TRDR, U=ROSCOEC ROSCOEC PQ
S IOFSLAM IOFSLAM P
S HCF HCF PQ
S XCOM XCOM PQ
S TELEØØ5A TELEØØ5APQ
S JCLARCH JCLARCH PQ
S IMS2 IMSVSTP2P
NETWTOR3

NETWTOR3 contains responses that are required for action messages from applications that have been just been activated. A pregnant pause is allowed between the time those applications are activated and the time outstanding WTORs are sought in order to give those applications time to reach that point in their processing. Slower processors may require a longer delay. Columns 1 - 8 contain the first eight characters of a message identifier; the response to it follows, beginning in column nine.

ERB3Ø6D GO
IST183A NO
U11D-531R
InCert Software Corporation has announced the availability of TraceBack, used for tracing back statement sequences to identify root causes of application failure. TraceBack uses binary instrumentation, monitors application execution and traces the exact sequence of statements leading up to application failure. By reviewing the trail, statement by statement, from the point of failure, users can analyze how and why the failure occurred. TraceBack will eliminate the need to replicate application failures in test environments for standard debuggers to be deployed to provide such sequencing information. TraceBack provides this capability with no measurable performance overhead.

InCert has also announced the availability of Examiner 2.1, an enhanced version of its MVS COBOL test analysis tool. Examiner allows developers to check software applications to see precisely how much of a new or already deployed application has been tested. Examiner, through binary instrumentation techniques, is also able to determine exactly which code has been exercised with both test and production environments. Examiner 2.1 further enhances quality efforts by ensuring that notification occurs when changed code is being put into production without being tested.

For further information contact:
InCert Software Corporation, 201 Broadway, Fifth Floor, Cambridge, MA 02139, USA.
Tel: (617) 621 8080
Fax: (617) 621 8081
www.incert.com

Walser Software+Support AG has announced the release of a new language for OS/390 called CAO. CAO has taken 10 years to develop and contains only four language constructions, but it can support multiple tasks, 3270 datastreams, and host graphic support. The implemented functions are for VTAM PLU (Primary Logical Unit) and SLU (Secondary Logical Unit). The CAO compiler translates the CAO statements directly into S/390 machine code for extremely rapid running.

For further information contact:
Walser Software+Support AG, Im Dorfli 29, 8953 Zurich-Dietikon, Switzerland.
Tel: 1 1774 15 88
Fax: 1 1774 15 89

Forte Software has announced the availability of the Forte Transaction Adapter for OS/390. The transaction adapter enables users to integrate application programs running under CICS, IMS and other MVS subsystems into extended and composite applications. It works in conjunction with the Forte Application Server.

For further information contact:
Forte Software Inc, 1800 Harrison Street, Oakland, CA 94612, USA.
Tel: (510) 869 3400
Fax: (510) 834 1508
Forte Software, St James House, Oldbury, Bracknell, Berkshire, RG12 8SA, UK.
Tel: (01344) 482100
Fax: (01344) 420905
www.forte.com