```
ERRMSG7  DS    ØF
         DC    AL2(ERRMSG7E-ERRMSG7)
         DC    H'Ø'
         DC    C'CONFLICTING PARSE PARAMETERS - PROGRAM IN ERROR'
ERRMSG7E EQU   *
ERRMSG8  DS    ØF
         DC    AL2(ERRMSG8E-ERRMSG8)
         DC    H'Ø'
         DC    C'TERMINAL HAS BEEN DISCONNECTED - PROGRAM ENDING'
ERRMSG8E EQU   *
ERRMSG9  DS    ØF
         DC    AL2(ERRMSG9E-ERRMSG9)
         DC    H'Ø'
         DC    C'ANSWER ADDRESS FROM PARSE IS ZERO - PROGRAM ENDING'
ERRMSG9E EQU   *
ERRMSGA  DS    ØF
         DC    AL2(ERRMSGAE-ERRMSGA)
         DC    H'Ø'
         DC    C'ERROR CALCULATING JULIAN DAY - PROGRAM ABENDING'
ERRMSGAE EQU   *
ECBADS   DC    F'Ø'
IOPLADS  DC    4F'Ø'
PUTBLOK  PUTLINE MF=L
STCKPCL  IKJPARM
TIMESTMP IKJIDENT 'TIMESTAMP',MAXLNTH=16,FIRST=NONATNUM,                *
               OTHER=NONATNUM,PROMPT='TIMESTAMP TO BE CONVERTED',CHAR
         IKJENDP
         CVT   DSECT=YES
         IKJCPPL
         IKJIOPL
         END
```

David McGeorge
Consultant Systems Programmer (Australia)

A TCB print utility

INTRODUCTION

Every now and then it can be useful to get some idea of what is going on inside an address space. This can be the case if a job or started task is 'hanging', or when a TSO user calls the help desk and you do not know exactly which programs he or she is running.

Some of the more advanced MVS monitors, like OMEGAMON, let you peek inside an address space and give you detailed information. However, it is not always possible or permissible to use that kind of heavy artillery for what might be fairly trivial tasks.

THE PROGRAM

To aid me in reviewing running address spaces, I have written a small TSO command processor called TCBPRT. TCBPRT is designed to give a quick summary of the active tasks and programs, as well as displaying their relationships to each other. A typical TCBPRT output might look like this:

```
TCB structure of job #YCØ6JVT :
IEAVARØØ
  IEESB6Ø5
  +called IEFSDØ6Ø
  | IEFIIC
  |   IKJEFTØ1
  |     IKJEFTØ2
  |     | TCBPRT
  |     IKJEFTØ2
  IEAVTSDT
```

The output contains an appealing overview of the TCB structure, including the RB chain. Indentations mark mother/daughter relationships: tasks connected by a 'l' are sisters. Beware however that the program does not show which program in the RB chain of a task created a certain daughter task: first the whole RB chain is printed, then the daughter tasks.

TCBPRT is a TSO command processor which should be placed in the TSO authorized command table. An IPL or PARMLIB command is necessary to activate the new table.

TCBPRT INTERNALS

The TCBPRT program is written in S/390 Assembler language, using ESA features such as the linkage stack and access register mode. TCBPRT is able to display the TCB/RB structure of all tasks in the system. This information is collected from the appropriate address space by using the MVS/ESA Advanced Address Space Features. The

processing of the TCB/RB control block structures is done recursively – the ESA linkage stack comes in handy here.

To obtain an ALET for the target address space we must have an STOKEN (address space token) and an extended authorization index (EAX). The STOKEN for an address space can be found in the Address Space Secondary Block (ASSB) control block, which is conveniently located in common storage. Since we're not invited to the target address space but plan to 'gate crash' it, we have to resort to supervisor state in order to bypass EAX checking. The ALESERV macro with CHKEAX=NO and the just located STOKEN gives us an ALET for the target address space.

The TCB/RB control block structure invites us to a recursive walk-through of the whole tree. The structure is shown in Figure 1.

The whole tree starts with the Region Control Task TCB (IEAVAR00), a pointer to which can be found in the ASCB.

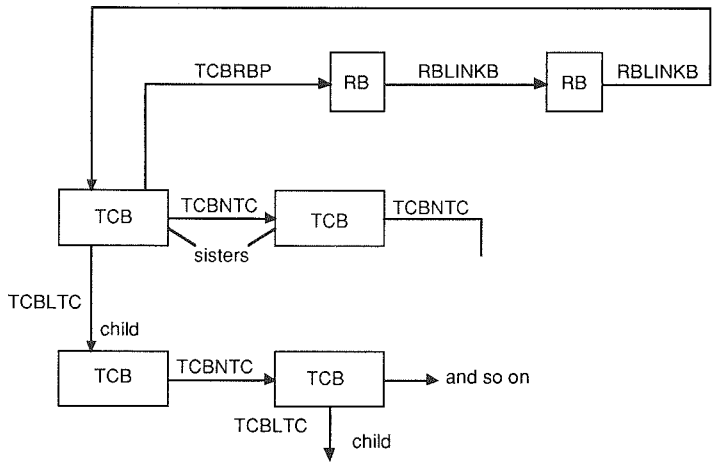

Figure 1: The TCB/RB control block structure

REMARKS, KNOWN PROBLEMS, AND FUTURE EXTENSIONS

The TCBPRT program is extremely small and elegant. The larger part of its code is devoted to processing the command line parameters and locating the requested address space.

A point worth noting at this moment is that TCBPRT can currently only process swapped-in address spaces. If an address space is swapped out, an ALESERV ADD RC=076 will occur. To enhance this aspect, TCBPRT would have to ascertain the swap status of the target address space by scheduling a SYSEVENT TRANSSWAP to it.

Another possible problem is caused by an error in MVS described by APAR OY44701 (PTF UY67682). The APAR describes a situation where a program ALESERV DELETEs a previously obtained ALET. If the address space to which the ALET points is swapped out at the time of the ALESERV DELETE some bits are not reset. As a result of this, MSGIEF356I will be issued at address space termination time and the address spaces ASID will not be reusable for the life of the IPL.

```
*********************************************************************
* TCBPRT - TCB structure print utility
* This utility prints out the TCB structure of an address space plus
* all active programs and SVCs.
* Syntax:   TCBPRT  jobname
*********************************************************************
TCBPRT   START 0                        - TCBPRT program is AMODE 31
TCBPRT   AMODE 31                       - and has no RMODE restrictions
TCBPRT   RMODE ANY
* Standard MVS/ESA entry
         BAKR  14,0                     - "New" MVS/ESA entry logic,
         LAE   12,0(15,0)               - enhanced to dynamically obtain
         USING TCBPRT,12                - a workarea
         B     CATCH
         DC    C'|TCBPRT|Written by Jos Visser|&SYSDATE|&SYSTIME|'
CATCH    STORAGE OBTAIN,LENGTH=WORKALEN
         LAE   13,0(1,0)
         LR    2,1                      - Fill the just obtained
         L     3,=AL4(WORKALEN)         - workarea with zeroes.
         XR    4,4
         XR    5,5
         MVCL  2,4
         USING WORKAREA,13
         MVC   4(4,13),=C'F1SA'         - Indicate linkage stack usage
         MVC   CATCH2,=C'W*RKAREA'      - Eyecatcher for dump analysis
         EREG  1,5
         USING CPPL,1                   - Copy TSO command processor
```

```
         LA    2,MYIOPL                - parameter list fields to
         USING IOPL,2                  - my own IOPL and PPL
         LA    3,MYPPL
         USING PPL,3
         L     4,CPPLCBUF              - Command buffer address
         ST    4,PPLCBUF               - To PPL
         L     4,CPPLUPT               - User profile table address
         ST    4,IOPLUPT               - To IOPL
         ST    4,PPLUPT                - And PPL
         L     4,CPPLECT               - Environment control table addr
         ST    4,IOPLECT               - To IOPL
         ST    4,PPLECT                - And PPL
         LA    4,MYECB                 - ECB provided by me
         ST    4,IOPLECB               - Store its address in the IOPL
         ST    4,PPLECB                - and in the PPL
         L     2,=V(MYPCL)             - IKJPARS PCL
         ST    2,PPLPCL                - To PPL
         LA    2,MYANS                 - IKJPARS answer area
         ST    2,PPLANS                - To PPL
         DROP  2
         DROP  1
         LA    1,MYPPL                 - Call IKJPARS to try to make
         LINK  EP=IKJPARS              - some sense of the parameters
         LTR   15,15                   - If parameter syntax was ok,
         BZ    PARMOK                  - Continue
*                                      - Else print error message
         PUTLINE PARM=PUTBLOK,OUTPUT=(PARMERR,DATA),MF=(E,MYIOPL)
         B     ERUIT                   - And Finito Benito
PARMOK   L     3,MYANS                 - Move the name of the job
         USING MYDSECT,3               - whose TCB structure must
         LH    4,JOBPARM+4             - be printed from the IKJPARS
         L     3,JOBPARM               - answer area to the workarea.
         DROP  3                       - This involves getting its
         BCTR  4,Ø                     - length and address and then
         EX    4,MOVEJOBP              - the actual move.
         OC    JOBNAME,SPACES          - Jobname to upper case and
*                                      - X'ØØ' to spaces (X'4Ø').
         XR    3,3                     - Obtain the address of the
         USING PSA,3                   - Address Space Vector Table
         L     3,FLCCVT                - (ASVT) from the CVT.
         USING CVT,3
         L     3,CVTASVT
         USING ASVT,3
         L     2,ASVTMAXU              - No of entries in ASVT
         LA    5,ASVTENTY              - Address of first entry
         DROP  3
LOOPØ1   TM    Ø(5),ASVTAVAL           - Is this Address Space entry
         BO    NEXTØ1                  - free? If so continue with next
         L     6,Ø(5)                  - Otherwise load the ASCB ptr
         USING ASCB,6
         L     7,ASCBJBNI              - Is it an initiated task (job)
```



```
         LTR   7,7                     - (then ASCBJBNI field zero)
         BNZ   JOBCMP                  - Yes, compare the job name
         L     7,ASCBJBNS              - No, load the STC/TSU name
JOBCMP   CLC   JOBNAME,Ø(7)            - Is this the job we're looking
         BE    FOUNDJOB                - for? Yes -> look no further
NEXTØ1   LA    5,4(5)                  - Look at the next ASVT entry
         BCT   2,LOOPØ1                - And Do The Loop
*                                      - All ASVT entries searched and
*                                      - no match found? --> Error msg
         PUTLINE PARM=PUTBLOK,OUTPUT=(JOBERR,DATA),MF=(E,MYIOPL)
         B     ERUIT                   - and return to caller
FOUNDJOB L     3,ASCBRCTP              - Get addr of Region Control TCB
         L     6,ASCBASSB              - And of the AS secondary block
         USING ASSB,6
         MODESET KEY=ZERO,MODE=SUP     - Whoops, necessary to skip
*                                      - EAX authorization checking to
*                                      - address space.
*                                      - Get an ALET for the addr space
         ALESERV ADD,STOKEN=ASSBSTKN,ALET=MYALET,CHKEAX=NO,            X
               MF=(E,ALESERV)
         DROP  6
         LR    6,15                    - Remember return code
         MODESET KEY=NZERO,MODE=PROB   - Phew, regaining normality
         LTR   6,6                     - Was ALESERV successful?
         BZ    ALETOK                  - Yep, continue
         MVC   LINE(ALERRLEN),ALERR    - Else format an error message
         CVD   6,WORK8                 - containing the ALESERV return
         UNPK  LINET+57(3),WORK8       - code.
         OI    LINET+59,X'FØ'          - Write out the error msg
         PUTLINE PARM=PUTBLOK,OUTPUT=(LINE,DATA),MF=(E,MYIOPL)
         B     ERUIT                   -
ALETOK   MVC   LINE(TITLELEN),TITLE    - Format the report title
         MVC   LINET+21(8),JOBNAME     - containing the job name
*                                      - And print the title
         PUTLINE PARM=PUTBLOK,OUTPUT=(LINE,DATA),MF=(E,MYIOPL)
         PUTLINE PARM=PUTBLOK,OUTPUT=(BLANKL,DATA),MF=(E,MYIOPL)
         MVC   LINET,SPACES            - Initialize the output line
         LA    2,82                    - buffer for PUTLINE uses
         STH   2,LINE
         LA    2,LINET                 - R2 points to the current pos
         LAM   3,3,MYALET              - Load the ALET for the address
         CPYA  4,3                     - space into AR3 (TCB),
         CPYA  7,3                     - AR4 (RB) and AR7 (CDE).
         SAC   512                     - Enable as access (AR mode)
         BAL   14,DOIT                 - Print the TCB structure
         SAC   Ø                       - And back to primary mode
         MODESET KEY=ZERO,MODE=SUP     - Glub, nobody seen it???
*                                      - Delete access to other AS
         ALESERV DELETE,ALET=MYALET,CHKEAX=NO,MF=(E,ALESERV)
         LR    6,15                    - Remember ALESERV return code
         MODESET KEY=NZERO,MODE=PROB   - And back to safety
```

```
         LTR   6,6                      - Was ALESERV successful?
         BZ    ERUIT                    - Yes, nothing left, return
         MVC   LINE(DELERRL),DELERR     - Format error message
         CVD   6,WORK8                  - containing ALESERV return
         UNPK  LINET+37(3),WORK8        - code
         OI    LINET+39,X'FØ'           - And print the error message
         PUTLINE PARM=PUTBLOK,OUTPUT=(LINE,DATA),MF=(E,MYIOPL)
* STANDARD MVS/ESA EXIT
ERUIT    LAE   1,Ø(13,Ø)                - Release the workarea, load
         LH    4,RETCODE                - the return code and leave
         STORAGE RELEASE,ADDR=(1),LENGTH=WORKALEN
         LR    15,4                     - the program
         PR
         SYSSTATE ASCENV=AR             - Tell MVS macros we're in
*                                       - AR mode
* DOIT - Print the TCB structure (recursive!)
*      : R3 points to the TCB to process
*      : R2 points into the output line
DOIT     BAKR  14,Ø                     - Save the registers
         USING TCB,3
LOOPØ2   LTR   3,3                      - End of sister TCB chain?
         BZ    EXITØ2                   - If so, we're through with it
         L     4,TCBRBP                 - Otherwise load the active RB
         MVI   FIRSTRB,X'Ø1'            - The first printed RB flag
         BAL   14,RBCHAIN               - Print the chain of RBs
         L     5,TCBLTC                 - Does this TCB have any
         LTR   5,5                      - daughters?
         BZ    NEXTSIS                  - If not, process next sister
         CLC   TCBNTC,=F'Ø'             - Otherwise process the
         BE    NOSISTER                 - daughters by sinking one level
         MVC   Ø(2,2),=C'| '            - into the tree. Advance the
NOSISTER LA    2,2(,2)                  - output line pointer and
         LR    6,3                      - include a | if there is a
         LR    3,5                      - sister down the line
         BAL   14,DOIT                  - Process the child TCB chain
         BCTR  2,Ø                      - We're back. Restore the output
         BCTR  2,Ø                      - line and process the sister
         MVC   Ø(2,2),SPACES
         LR    3,6
NEXTSIS  L     3,TCBNTC                 - Get the address of the sister
         B     LOOPØ2                   - and process it
         DROP  3
EXITØ2   PR
* RBCHAIN - Print the RB chain in correct call order
*         : R4 points to the active RB
RBCHAIN  BAKR  14,Ø
         USING RBSECT,4
         TM    RBSTAB2,RBTCBNXT         - Is the next RB a TCB?
         BO    ENDRECUR                 - If so, no need to hop further
         LR    5,4                      - Otherwise print the RB chain
         ICM   4,B'Ø111',RBLINKB        - starting at the next RB of
```



```
         BAL   14,RBCHAIN               - this RB
         LR    4,5                      - We're back, now only print
ENDRECUR XR    7,7                      - this RB and we're through
         ICM   7,B'Ø111',RBCDE1         - Load the CDE of this RB
         LTR   7,7                      - If no CDE, then probably an
         BZ    NOCDE                    - SVRB of a type 2 SVC (nucleus)
         USING CDENTRY,7                - If a CDE is present, then
         MVC   ENTRNAME,CDNAME          - move the load module name to
         DROP  7                        - a work field and go on
         B     FORMATL
NOCDE    MVC   ENTRNAME,=C'SVC XXX '    - Otherwise pick up the
         PUSH  USING                    - interrupt code from the RB
         SH    4,=H'64'                 - prefix and format an SVC
         USING RBPREFIX,4               - message into the workfield
         LH    11,RBINTCOD
         LA    4,64(4)
         POP   USING
         CVD   11,WORK8
         UNPK  ENTRNAME+4(3),WORK8
         OI    ENTRNAME+6,X'FØ'
FORMATL  CLI   FIRSTRB,X'Ø1'           - If this is the first RB of the
         BNE   NOFIRST                  - TCB just move its "name"
         MVC   Ø(8,2),ENTRNAME          - (workfield) to the output line
         B     PRTLINET
NOFIRST  MVC   Ø(8,2),=C'+called '      - Otherwise first move in a
         MVC   8(8,2),ENTRNAME          - "called" phase
PRTLINET SAC   Ø                        - PUTLINE can't take AR mode too
         SYSSTATE ASCENV=P              - well, so switch to prim mode
*                                       - Print the output line
         PUTLINE PARM=PUTBLOK,OUTPUT=(LINE,DATA),MF=(E,MYIOPL)
         SAC   512                      - And back to real life,
         SYSSTATE ASCENV=AR             - adventure and really wild
*                                       - things
         MVC   Ø(2Ø,2),SPACES           - Clear part of the output line
         MVI   FIRSTRB,X'ØØ'            - First RB has been printed
         DROP  4
         PR                             - And return.....
         SYSSTATE ASCENV=P
* Constants
MOVEJOBP MVC   JOBNAME(Ø),Ø(3)
SPACES   DC    CL8Ø' '
TITLE    DC    AL2(L'TITLET+4),AL2(Ø)
TITLET   DC    C'TCB structure of job xxxxxxxx :'
TITLELEN EQU   *-TITLE
BLANKL   DC    AL2(5),AL2(Ø),C' '
PARMERR  DC    AL2(L'PARMERRT+4),AL2(Ø)
PARMERRT DC    C'Parameter error'
ALERR    DC    AL2(L'ALERRT+4),AL2(Ø)
ALERRT   DC    C'Could not establish access to required address space, X
               RC=XXX'
ALERRLEN EQU   *-ALERR
```

```
JOBERR   DC    AL2(L'JOBERRT+4),AL2(Ø)
JOBERRT  DC    C'Specified job not found'
DELERR   DC    AL2(L'DELERRT+4),AL2(Ø)
DELERRT  DC    C'Error deleting cross memory link, RC=XXX'
DELERRL  EQU   *-DELERR
* Command processor parameter definitions
MYPCL    IKJPARM DSECT=MYDSECT
JOBPARM  IKJIDENT 'JOBNAME',MAXLNTH=8,FIRST=ALPHANUM,OTHER=ALPHANUM,   X
               PROMPT='NAME OF JOB TO PRINT TCB STRUCTURE OF',CHAR
         IKJENDP
* Dynamically obtainable work area
WORKAREA DSECT
SAVEAREA DS    18F                    - Save area for pre-ESA progs
CATCH2   DS    CL8                    - Eye catcher for dump analysis
RETCODE  DS    H                      - Program return code
         DS    ØD
WORK8    DS    PL8                    - For use in CVD/UNPK sequences
MYECB    DS    F                      - An ECB for PUTLINE and IKJPARS
MYANS    DS    F                      - An answer area for IKJPARS
MYALET   DS    F                      - ALET for target address space
MYIOPL   DS    4F                     - An IOPL for PUTLINE
MYPPL    DS    8F                     - A PPL for IKJPARS
LINE     DS    2H                     - Output line prefix (len/ofs)
LINET    DS    CL8Ø                   - The output line text
ENTRNAME DS    CL8                    - Workfield for output line fmt
JOBNAME  DS    CL8                    - Name of job to process
FIRSTRB  DS    XL1                    - Flag to determine if 1st RB
*                                     - of the TCB has been printed
PUTBLOK  PUTLINE MF=L                 - And again... for PUTLINE
ALESERV  ALESERV MF=L                 - List form ALESERV
WORKALEN EQU   *-WORKAREA             - Length of the workarea
         DROP  13                     - R13 addresses the workarea
* Mapping macros
         IHAPSA DSECT=YES             - Prefixed Storage Area
         IHARB                        - Request Block
         IHACDE                       - Contents Directory Entry
         IKJTCB DSECT=YES             - Task Control Block
         IKJCPPL                      - Command Proc Param List
         IKJIOPL                      - Input Output Param List
         IHAASCB                      - Address Space Control Block
         IKJPPL                       - Parse Parameter List
         CVT     DSECT=YES            - Communications Vector Table
         IHAASVT DSECT=YES            - Address Space Vector Table
         IHAASSB                      - Address Space Secondary Block
         END
```

Jos Visser
Systems Specialist (The Netherlands)

VM console under TSO

The following program uses the diagnose command to pass commands to VM. It then displays the response on the user's screen in full screen format just like a VM console (see Figure 1).

This program can be called under native TSO or ISPF. Use PF3 to terminate it.

```
R; T=Ø.Ø1/Ø.Ø1 13.35.56
q name
IPO3     - ØF4C, IPO2     - ØØF2, OPERATOR - DISC, EREP     - DISC
PVM      - DISC, RSCS     - DISC, TUBES    - DISC, VMUTIL   - DISC
ISPVM    - DISC, VPAC     - DISC, JAS      - DISC, DISKACNT - DISC
R; T=Ø.Ø1/Ø.Ø1 13.36.Ø3
q tape
TAPE ØC8Ø ATTACHED TO IPO3     ØC8Ø R/W
TAPE ØC81 ATTACHED TO IPO2     ØC81 R/W
TAPE ØC82 ATTACHED TO IPO2     ØC82 R/W
R; T=Ø.Ø1/Ø.Ø1 13.36.14
q dasd
DASD Ø28Ø ATTACHED TO IPO2     Ø28Ø R/W JEC2ØØ
DASD Ø281 ATTACHED TO IPO2     Ø281 R/W JEC2Ø1
DASD Ø282 ATTACHED TO IPO2     Ø282 R/W JBBSYS
DASD Ø283 ATTACHED TO IPO2     Ø283 R/W SPL2Ø1
DASD Ø284 ATTACHED TO IPO2     Ø284 R/W JEC229
DASD Ø285 ATTACHED TO IPO2     Ø285 R/W JEC2Ø7
DASD Ø286 ATTACHED TO IPO2     Ø286 R/W JBBPPS
DASD Ø287 ATTACHED TO IPO2     Ø287 R/W JBXCAT
DASD Ø288 ATTACHED TO IPO2     Ø288 R/W JEC231
DASD Ø289 ATTACHED TO IPO2     Ø289 R/W JEC232

                                          MORE... > VM/XA SF
```

Figure 1: Sample display