

Share Session 8130

Don Poitras

Sas Institute, Inc.

SHR2MAN.C

```
/*-----+
| SHARE Debugging Example - 2000 Winter conference |
| session 8130 |
+-----*/
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <getmain.h>
#include <osio.h>
#include <dynam.h>
#include "shr2.h"

extern GLOBAL global;          /* Global data */

static void Title(void);
static int ReadFile(void);
static void DynamCall(void);

static __remote prtable (*fp()); /* points to _dynam function */
extern prtable table;          /* allow any linked program to
                               call dynam functions */

extern int count = 0;

/*-----+
| Demo program that demonstrates a few debugging methods |
+-----*/
main()
{
    int rc;

    global.count = 7;
    global.callback = ReadFile;
    Title();          /* print title line */
    ReadFile();      /* call assembly rtn*/
    DynamCall();     /* call routine in other load mod. */
    count = 10;      /* set extern counter */
    printf("Start local count=%i\n", count);
    printf("Start global count=%i\n", global.count);
    rc = (*table->func1>(&global); /* call dynam function */
    rc = (*table->func2)();        /* call dynam function */

    unloadm(fp);                /* cleanup */
}
```

```

printf("End local count=%i\n", count);
printf("End global count=%i\n", global.count);
return (EXIT_SUCCESS);
}

```

```

/*-----+
|
| FUNCTION NAME:      Title
|
| DESCRIPTIVE NAME:  print title line
|
|-----*/

```

```

static void Title(void)
{
    time_t dTime;
    struct tm *pTime;

    time(&dTime);
    pTime = localtime(&dTime); /* parse date */
    printf("%s %s, %s %u, %04u\n",
           "SHARE Debugging Example - ",
           apcDays[pTime->tm_wday],
           apcMonths[pTime->tm_mon],
           pTime->tm_mday,
           pTime->tm_year + 1900);
}

```

```

/*-----+
|
| FUNCTION NAME:      ReadFile
|
| DESCRIPTIVE NAME:  Read a file via BSAM
|
|-----*/

```

```

static int ReadFile(void)
{
    char *buf;
    int len;
    DCB_t *input; /* Input DCB */
    DECB_t input_DECB;
    int err;
    int i;

    count = 20;
    input = osbdcB(0);
    memcpy(input->DCBDDNAM, "INPUT ", 8);
    if( osbopen( input, "input") )
    {
        printf("Input Open Failed\n");
    }
    buf = (char *) GETMAIN_U(input->DCBBLKSI, 0, LOC_BELOW);

    for(;;)
    {
        osread(input_DECB, input, buf,0);
        if ((err = oscheck(input_DECB)) != 0)
        {

```

```

    if( err != -1 )
    {
        printf("Input Error(%d)\n", err);
    }
    else
    {
        printf("eof(%d)\n", err);
    }

    break;
}
else
{
    printf("rec = ");
    i = 0;
    len = input->DCBBLKSI;
    while (len--)
    {
        printf("%02X", buf[i++]);
    }
    printf("\n");
}
}
osbclose(input, "", 1);
return 0;
}
/*-----+
|
| FUNCTION NAME:      DynamCall
|
| DESCRIPTIVE NAME:  Call function in other load module
|                   _dynam function returns table of remote function
|                   pointers.
|
+-----*/
static void DynamCall(void)
{
    SEARCH_P pSearch;

    pSearch = addsrch(MVS_DD, "sub", "shr2");
    loadm("shr2dyn", &fp);
    if (0 == fp)
    {
        printf("Loadm failed\n");
    }
    else
    {
        printf("Loadm succeeded\n");
        table = (*fp)();
    }
}
}

```

SHR2DYN.C

```
/*-----+
| SHARE Debugging Example - 2000 Winter conference |
| session 8130 |
+-----*/
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <dynam.h>
#include "shr2.h"
static int count = 0;

static int func1(PGLOBAL);
static int func2(void);
static void log(char *str);
static rtable dyntab = {&func1, &func2};
static PGLOBAL pGlobal; /* points to global data */

/*-----+
| Initialize function pointer table |
+-----*/
prtable _dynamn(void)
{
    return(&dyntab);
}

/*-----+
| FUNCTION NAME: func1 |
| DESCRIPTIVE NAME: Dynamically loaded function |
+-----*/
static int func1(PGLOBAL pGlobalIn)
{
    pGlobal = pGlobalIn; /* set global data pointer */
    printf("func1() was successfully dynamically called!!!\n");
    printf("Count from SHR2DYN=%i\n", count);
    return(0);
}

/*-----+
| FUNCTION NAME: func2 |
| DESCRIPTIVE NAME: Dynamically loaded function |
+-----*/
static int func2(void)
{
    int result;
    printf("func2() was successfully dynamically called!!!\n");
    count = 5;
    printf("Local count from SHR2DYN=%i\n", count);
    printf("global count from SHR2DYN=%i\n", pGlobal->count);
}
```

```

    pGlobal->count = 8;
    result = (*pGlobal->callback) ();
    log("from func2\n");
    printf("Local count from SHR2DYN=%i\n", count);
    printf("global count from SHR2DYN=%i\n", pGlobal->count);
// abort();
    return(0);
}
static void log(char *str)
{
    time_t dTime;
    struct tm *pTime;

    time(&dTime);
    pTime = localtime(&dTime); /* parse date */
    printf("%s, %s %u, %04u - %s\n",
        apcDays[pTime->tm_wday],
        apcMonths[pTime->tm_mon],
        pTime->tm_mday,
        pTime->tm_year + 1900,
        str);
}

```

SHR2.H

```
/*-----+
|  SHARE Debugging Example - 2000 Winter conference  |
|  session 8130                                     |
+-----*/
typedef struct _GLOBAL      /* global data */
{
    int count;              /* counter that can be seen by all load
modu
    int (*callback)(void); /* function in main module */
} GLOBAL, *PGLOBAL;
typedef struct _rtable /* structure definition for functions */
{
    __remote int (*func1)();
    __remote int (*func2)();
                                /* More functions can go here. */
} rtable, *prtable;
static const char * const apcDays[7] =
{
    "Sunday",
    "Monday",
    "Tuesday",
    "Wednesday",
    "Thursday",
    "Friday",
    "Saturday"
};

static const char * const apcMonths[12] =
{
    "January",
    "February",
    "March",
    "April",
    "May",
    "June",
    "July",
    "August",
    "September",
    "October",
    "November",
    "December"
};
```

SHR2MANC.JCL

```
//SHR2MANC JOB (,R214),POITRAS,NOTIFY=SASDTP2,TIME=(0,10),CLASS=A,
//*          RESTART=STEP2,
//          MSGCLASS=A
/*JOBPARM FETCH
// JCLLIB ORDER=LSD.CR650.PROCLIB
//*
//* /*-----+
//* | SHARE DEBUGGING EXAMPLE - 2000 WINTER CONFERENCE |
//* | SESSION 8130 |
//* +-----*/
//*
//STEP1 EXEC PGM=ASMA90,PARM='RENT,OBJECT,ADATA'
//SYSPRINT DD SYSOUT=*
//SYSLIB DD DSN=LSD.LC650.MACLIB,DISP=SHR
// DD DSN=SYS1.MACLIB,DISP=SHR
// DD DSN=SYS1.AMODGEN,DISP=SHR
//SYSTEM DD SYSOUT=*
//SYSPUNCH DD DUMMY
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSADATA DD DSN=SASDTP.ADATA,DISP=SHR
//SYSLIN DD DSN=SASDTP.TEST.OBJ(L$UBSAM),DISP=SHR
//SYSIN DD DSN=SASDTP.TEST.ASM(L$UBSAM),DISP=SHR
//STEP2 EXEC PGM=IKJEFT01,DYNAMNBR=100,REGION=4096K,
// PARM='ASMLANGX L$UBSAM (ASM LOUD ERROR'
//SYSTSIN DD DUMMY
//SYSTSPRT DD SYSOUT=*
//LISTING DD SYSOUT=*
//SYSADATA DD DSN=SASDTP.ADATA,DISP=SHR
//ASMLANGX DD DSN=SASDTP.ASMLANGX,DISP=SHR
//STEP3 EXEC LC370CLR,PARM.C='BI(1),EXT,DEB,OMD,SN(SHR2DYN),PFL',
// PARM.LKED='RENT,REFR,PRMAP,GMAP,XREF,MAP,
// ENXREF(SNAME)',
// REGION.C=1024K
//C.SYSDBLIB DD DSN=SASDTP.DBLIB,DISP=SHR
//C.SYSIN DD DSN=SASDTP.TEST.C(SHR2DYN),DISP=SHR
//C.SYSLIN DD DSN=SASDTP.TEST.OBJ(SHR2DYN),DISP=SHR
//C.H DD DSN=SASDTP.TEST.H,DISP=SHR
//LKED.SYSDBLIB DD DSN=SASDTP.DBLIB,DISP=SHR
//LKED.SYSLMOD DD DSN=SASDTP.TEST.LOAD2(SHR2DYN),DISP=SHR
//LKED.OBJLIB DD DSN=SASDTP.TEST.OBJ,DISP=SHR
//LKED.SYSIN DD *
MODE AMODE(31),RMODE(ANY)
INCLUDE OBJLIB(SHR2DYN)
NAME SHR2DYN(R)
/*
//STEP4 EXEC LC370CLR,PARM.C='RENT,BI(1),EXT,DEB,OMD,SN(SHR2MAN)',
// PARM.LKED='RENT,REFR,PRMAP,GMAP,XREF,MAP,
// ENXREF(SNAME)',
// REGION.C=1024K
//C.SYSDBLIB DD DSN=SASDTP.DBLIB,DISP=SHR
//C.SYSIN DD DSN=SASDTP.TEST.C(SHR2MAN),DISP=SHR
//C.SYSLIN DD DSN=SASDTP.TEST.OBJ(SHR2MAN),DISP=SHR
//C.H DD DSN=SASDTP.TEST.H,DISP=SHR
```

```
//LKED.SYSDBLIB DD DSN=SASDTP.DBLIB,DISP=SHR
//LKED.SYSLMOD DD DSN=SASDTP.TEST.LOAD(SHR2MAN),DISP=SHR
//LKED.OBJLIB DD DSN=SASDTP.TEST.OBJ,DISP=SHR
//LKED.SYSIN DD *
MODE AMODE(31),RMODE(ANY)
INCLUDE OBJLIB(SHR2MAN)
INCLUDE OBJLIB(L$UBSAM)
NAME SHR2MAN(R)
/*
//
```

SHR2MAN.JCL

```
//SHR2MAN JOB (,R214),POITRAS,NOTIFY=SASDTP2,TIME=(1,30),CLASS=A,
//          MSGCLASS=A
//*JOBPARM FETCH
//*
//* /*-----+
//* | SHARE DEBUGGING EXAMPLE - 2000 WINTER CONFERENCE |
//* |   SESSION 8130                                     |
//* +-----*/
//*
//TEST1   EXEC PGM=SHR2MAN,REGION=1M,PARM='=V'
//STEPLIB DD DSN=SASDTP.TEST.LOAD,DISP=SHR
//SUB     DD DSN=SASDTP.TEST.LOAD2,DISP=SHR
//CTRANS  DD DSN=LSD.LC650.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTEM  DD SYSOUT=*
//*SYSMDUMP DD DSN=SASDTP.SYSMDUMP,DISP=SHR
//SYSUDUMP DD SYSOUT=*
//INPUT   DD *
TEST1
/*
//
```

SHR2MANC.CLIST

```
PROC 0
ALLOC F(CTRANS) DS('LSD.LC650.LOAD') SHR REUSE
ALLOC F(SUB) DS('SASDTP.TEST.LOAD2') SHR REUSE
ALLOC F(SYSTEM) DS(*) SHR REUSE
ALLOC F(SYSPRINT) DS(*) SHR REUSE
ALLOC F(INPUT) DS('SASDTP.SHR2MAN.TEST') SHR REUSE
TSOEXEC ASMIDF SHR2MAN (/=D
```

SCSTPLIB.REXX

```
/* REXX */
tname = 'scstplib'
tver  = 'v0.2'
/* Name:   scSTPLIB - SAS/C Steplib Lister          */
/*                                               */
/*                                               */
/* FUNCTION: Locate the Steplib for a given tcb and list  */
/*           the datasets in the order they are specified.  */
/*                                               */
/* INPUT:                                           */
/*                                               */
/*   REQUIRED: None                                     */
/*                                               */
/*   OPTIONAL:                                       */
/*                                               */
/*     DEBUG - Issue TRACE A to provide a trace function  */
/*     DEBUGI - Issue TRACE I to provide a trace function  */
/*     INNER - Call was made from another exec          */
/*                                               */
/* Credits:                                         */
/*   Don Poitras is the current owner and EXEC writer  */
/*   Terry Ross is the original author               */
/*******/

ADDRESS IPCS

parse upper arg p1 p2 p3 rest
Numeric digits 10

signal on halt
signal on failure

if (p1='DEBUG') | (p2='DEBUG') | (p3='DEBUG') then
  trace a
else if (p1='DEBUGI') | (p2='DEBUGI') | (p3='DEBUGI') then
  trace i
else
  trace off
if (p1='INNER') | (p2='INNER') | (p3='INNER') then
  inner=1
else
  inner=0
if (p1='ALL') | (p2='ALL') | (p3='ALL') then
  all=1
else
  all=0
if inner=0 then
  'EQU SCSaveX X'
if inner=0 then
  "NOTE '"tname'"('tver)': Please see Don Poitras for support.'" asis

/*-----*/
/* Locate the LLT in the dump.          */
/*-----*/
```

```

'EQU TEMPCVT 4C.? LE(8) POS(0)'
'EVAL TEMPCVT+4DC LE(4) REXX(STORAGE(LLTADDR)) POINTER'
if rc^=0 then signal failure
'EVAL scTCB+0C LE(4) REXX(STORAGE(TIOT)) POINTER'
'evalsym scTCB rexx(address(scTCB))'
'EQU TIOT 'TIOT' LE(10) POS(0)'
'EVAL 'TIOT'.+0 LE(8) REXX(STORAGE(JOBNAME)) CHAR'
if rc^=0 then signal failure
'EVAL 'TIOT'.+8 LE(16) REXX(STORAGE(STEPNAME)) CHAR'
if rc^=0 then signal failure
'EQU TIOTENTRY 'TIOT'.+18 LE(10) POS(0)'

"NOTE ' '"
"NOTE 'Job: "jobname" Step: "stepname"'" asis
"NOTE ' '"
stopit=0
indd=0
inlib=0
hdrprinted=0
tioteod=0
addr = eval(xa(scTCB,0b4)) /* addr(jscb) */
addr = eval(xa(addr,15c)) /* addr(active jscb) */
addr = eval(xa(addr,0f4)) /* addr(qmpa) */
qmat = eval(xa(addr,18)) /* addr(qmat) */
do until (stopit=1)
  'INTEGER 'tioteod' rexx(storage(tioteo))'
  'EVAL tiotentry+'tioteo'+0 LE(1) REXX(STORAGE(tiolngh)) unsigned'
  if rc^=0 then signal failure
  if tiolngh=0 then
    stopit=1
  else
    do
      'EVAL tiotentry+'tioteo'+4 LE(1) REXX(STORAGE(tioDDNM0)) unsigned'
      if rc^=0 then signal failure
      if tioDDNM0^=0 then
        do
          'EVAL tiotentry+'tioteo'+4 LE(8) REXX(STORAGE(tioDDNM)) CHAR'
          if rc^=0 then signal failure
          if tioDDNM^=' ' then
            do; indd=0; inlib=0; lastddnm=tioDDNM; end;
          if (tioDDNM='STEPLIB ') | (tioDDNM='JOBLIB ') | ,
              (tioDDNM='CTRANS ') | (all=1) then
            inlib=1
          if inlib=1 then
            do
              if hdrprinted=0 then
                do
                  hdrprinted=1
                  "NOTE ' DDName DataSetName'"
                  "NOTE ' -----'"
                end
            end
          'EVAL tiotentry+'tioteo'+0C LE(3) REXX(STORAGE(tioJFCB))
pointer'
          if rc^=0 then signal failure
          swa = xa(qmat,tioJFCB) /* addr(swa block address)-1 */
          jfcb = eval(xa(swa,1)) /* addr(swa block) */
          'EVAL 'jfcb'.+10 LE(44) REXX(STORAGE(jfcbDSNM)) char'

```

```

        if rc^=0 then jfcbdsnm = "***NA**"
        "NOTE ' "lastddnm" "jfcbdsnm"'
    end
    end
    tioteod=tioteod+20
    end
end
/*-----*/
/*This is the exit code.*/
/*-----*/
if inner=0 then
    'EQU X SCSaveX'
return 0

failure:
if inner=1 then
"NOTE '"tname"("tver)": Please see Don Poitras for support.'" asis
"NOTE '          Storage is not available.'" asis
"NOTE '          Possible reasons: 1) CSA was not dumped.'" asis
"NOTE '          2) This is not a complete dump.'"
asis
    return 4
/* add 2 hex numbers */
xa: procedure

    numeric digits 10
    x = x2d(arg(1)) + x2d(arg(2))
    if x > 2147483648 then x = x - 4294967296    /* resolve overflow */
    return d2x(x)

/* return value at specified address
*/

/*    */

/* Parms: */
/*    1 - address */
/*    2 - length of value to return (4 is default) */
/*    3 - type of value to return ('C' or 'X') ('X' is default) */
/* */

eval: procedure

    if arg(2) = ''
        then len = 4
        else len = arg(2)

    if arg(3) = ''
        then type = x
        else type = arg(3)

    "evaluate" arg(1)". length("len") " type " rexx(storage(result))"
if rc > 0
    then return -1
    else return result

```

```
halt:  
exit: exit
```

SCDINFO.REXX

```
/* REXX */
tname = 'scdinfo'
tver = 'v0.2'
rexplib = 'sasdtp.rexx'
/* Name:    scDInfo - SAS/C Dump Info */
/* */
/* */
/* FUNCTION: Provide information about the dump. */
/* */
/* INPUT: */
/* */
/*   REQUIRED: None */
/* */
/*   OPTIONAL: */
/* */
/*   DEBUG - Issue TRACE A to provide a trace function */
/*   DEBUGI - Issue TRACE I to provide a trace function */
/*   INNER - Call was made from another exec */
/* */
/* Credits: */
/*   Don Poitras is the current owner and EXEC writer */
/*   Mike Arnold was the original writer */
/*****/
```

ADDRESS IPCS

```
/* parse upper arg p1 p2 rest */
parse upper arg crab rest
Numeric digits 10
```

```
signal on halt
signal on failure
```

```
inner=0
```

```
/*
if (p1='DEBUG') | (p2='DEBUG') then
  trace a
else if (p1='DEBUGI') | (p2='DEBUGI') then
  trace i
else
  trace off
if (p1='INNER') | (p2='INNER') then
  inner=1
*/
```

```
if inner=0 then
```

```
  'EQU SCSaveX X'
```

```
if inner=0 then
```

```
"NOTE '"tname"("tver")'" asis
```

```
"NOTE '''
```

```
"NOTE '          *** SAS/C Diagnostic Worksheet ***'" asis
```

```
"NOTE '''
```

```
'EVALDEF REXX(SOURCE(cdsn) PROBLEM(prob) QUALIFICATION(aq))'
```

```

quote='''
parse var cdsn s1 (quote) ccdsn (quote) s2
"NOTE 'Source dataset:          "ccdsn"'" asis
parse var aq  s1 (quote) casid (quote) s2
"NOTE 'Address Space:          ASID('"casid"')'" asis
casidd = x2d(casid)
'EVAL DUMPTIMESTAMP  REXX(STORAGE(DT)) CHAR'
"NOTE 'Dump Time:          "dt"'" asis
'EVAL DUMPREQUESTOR  REXX(STORAGE(DR)) CHAR'
"NOTE 'Dump Requestor:          "dr"'" asis
'EVAL DUMPORIGINALDSNAME REXX(STORAGE(DSN)) CHAR'
"NOTE 'Original Dump Dataset:  "dsn"'" asis
'EVAL DUMPINGPROGRAM  REXX(STORAGE(DP)) CHAR'
"NOTE 'Dumping Program:          "dp"'" asis

If dp='SLIP  ' then
do
'EVAL SLIPTRAP  REXX(STORAGE(ST)) CHAR'
"NOTE 'SLIP command:          "st"'" asis
end
'eval PSW pos(0) len(4) rexx(storage(epsw0)) pointer'
'eval PSW pos(4) len(4) rexx(storage(epsw1)) pointer'
'eval 0r len(4) rexx(storage(reg0)) pointer'
'eval 1r len(4) rexx(storage(reg1)) pointer'
'eval 2r len(4) rexx(storage(reg2)) pointer'
'eval 3r len(4) rexx(storage(reg3)) pointer'
'eval 4r len(4) rexx(storage(reg4)) pointer'
'eval 5r len(4) rexx(storage(reg5)) pointer'
'eval 6r len(4) rexx(storage(reg6)) pointer'
'eval 7r len(4) rexx(storage(reg7)) pointer'
'eval 8r len(4) rexx(storage(reg8)) pointer'
'eval 9r len(4) rexx(storage(reg9)) pointer'
'eval 10r len(4) rexx(storage(reg10)) pointer'
'eval 11r len(4) rexx(storage(reg11)) pointer'
'eval 12r len(4) rexx(storage(reg12)) pointer'
'eval 13r len(4) rexx(storage(reg13)) pointer'
'eval 14r len(4) rexx(storage(reg14)) pointer'
'eval 15r len(4) rexx(storage(reg15)) pointer'

"NOTE '''
"note 'PSW at time of error:  "epsw0" "epsw1"'" asis
"NOTE '''
"note 'Registers at time of error:  '" asis
"Note ' R0:"reg0" R1:"reg1" R2:"reg2" R3:"reg3"'" asis
"Note ' R4:"reg4" R5:"reg5" R6:"reg6" R7:"reg7"'" asis
"Note ' R8:"reg8" R9:"reg9" R10:"reg10" R11:"reg11"'" asis
"Note ' R12:"reg12" R13:"reg13" R14:"reg14" R15:"reg15"'" asis
"archeck header analyze"
'evalsym ascb'casidd' rexx(address(ascbptr))'
"NOTE '''
"NOTE 'TCB Summary for ASID x'"casid"' ASCB:  "ascbptr"'" asis
"NOTE '''
'EVAL ascb'casidd'+6c 'aq' le(4) rexx(storage(asxbptr)) pointer'
if rc^=0 then signal failure
'EVAL 'asxbptr'+04 'aq' le(4) rexx(storage(tcbptr)) pointer'
if rc^=0 then signal failure
abendedtcbs.0 = 0

```

```

abendedcmp.0 = 0
do until tcbptr=0
  'eval 'tcbptr'.+10 'aq' le(4) rexx(storage(cmpf)) pointer'
  if rc^=0 then cmpf='*NA*'
  if cmpf^=0 then
    do
      j=abendedtcbs.0+1
      abendedtcbs.0 = j
      abendedtcbs.j = tcbptr
      abendedcmp.0 = j
      abendedcmp.j = cmpf
    end
  'eval 'tcbptr'.+0 'aq' le(4) rexx(storage(tcbrbp)) pointer'
  if rc^=0 then signal failure
  'eval 'tcbrbp'.+10 le(4) rexx(storage(opsw1)) pointer'
  if rc^=0 then signal failure
  'eval 'tcbrbp'.+14 le(4) rexx(storage(opsw2)) pointer'
  if rc^=0 then signal failure

  rba = tcbrbp
  stcbptr = substr(tcbptr,3)
  'eval 'rba'.+1D le(3) rexx(storage(rblink)) pointer'
  if rc^=0 then rblink=0
  do while rblink^=0 & rblink^=stcbptr & rblink^="000000"
    'eval 'rba'.+1D le(3) rexx(storage(rblink)) pointer'
    if rc^=0 then rblink=0
    if rblink^=0 & rblink^="000000" & rblink^=stcbptr then
      rba=rblink
    else
      rblink=0
  end

  'eval 'rba'.+60 le(8) rexx(storage(ep)) char'
  if rc^=0 then ep='UNKNOWN'
  if cmpf^=0 then
"NOTE '*TCB:  "tcbptr" CMP="cmpf" RB OPSW="opsw1" "opsw2" PRBEP="ep"
  ' " asis
  else
"NOTE ' TCB:  "tcbptr" CMP="cmpf" RB OPSW="opsw1" "opsw2" PRBEP="ep"
  ' " asis
  'eval 'tcbptr'.+74 'aq' le(4) rexx(storage(tcbptr)) pointer'
  if rc^=0 then signal failure
end
do i=1 to abendedtcbs.0
  "note ''
  "note '**Summary for TCB:  "abendedtcbs.i"***' " asis
  tcbptr = abendedtcbs.i
/*'cbf 'tcbptr'. str(tcb)' */
  'equ scTCB 'tcbptr'. LE(4)'
  "exec "rexxlib"(scstplib)' 'inner'"
  'eval 'tcbptr'.+24 'aq' le(4) rexx(storage(llepctr)) pointer'
  if rc^=0 then llepctr=0
  "note '' " asis
  "note 'Load List'" asis
  do while llepctr^=0
    'eval 'llepctr'.+4 'aq' le(4) rexx(storage(cdepctr)) pointer'
    if rc^=0 then cdepctr=0

```

```

if cdepctr ^= 0 then
  do
    'eval 'cdepctr'.+8 'aq' le(8) rexx(storage(cdename)) char'
    if rc ^= 0 then cdename '***NA***'
    'eval 'cdepctr'.+10 'aq' le(4) rexx(storage(cdeep)) pointer'
    if rc ^= 0 then cdeep = '*NA*'
    'eval 'cdepctr'.+18 'aq' le(2) rexx(storage(cdeuse)) unsigned'
    if rc ^= 0 then cdeuse = 'NA'
    "note ' CDE: "cdepctr" Name="cdename" EP="cdeep" Use
Count="cdeuse"' " asis

    end
    'eval 'lleptr'.+0 'aq' le(4) rexx(storage(lleptr)) pointer'
    if rc ^= 0 then lleptr = 0
  end
  'eval 'tcbptr'.+2c 'aq' le(4) rexx(storage(cdepctr)) pointer'
  if rc ^= 0 then lleptr = 0
  "note ''" asis
  "note 'Job Pack Area'" asis
  do while cdepctr ^= 0
    'eval 'cdepctr'.+8 'aq' le(8) rexx(storage(cdename)) char'
    if rc ^= 0 then cdename '***NA***'
    'eval 'cdepctr'.+10 'aq' le(4) rexx(storage(cdeep)) pointer'
    if rc ^= 0 then cdeep = '*NA*'
    'eval 'cdepctr'.+18 'aq' le(2) rexx(storage(cdeuse)) unsigned'
    if rc ^= 0 then cdeuse = 'NA'
    "note ' CDE: "cdepctr" Name="cdename" EP="cdeep" Use
Count="cdeuse"' " asis
    'eval 'cdepctr'.+0 'aq' le(4) rexx(storage(cdepctr)) pointer'
    if rc ^= 0 then lleptr = 0
  end
end

/* Try to locate the CRAB */
if crab = '' then
  crab = substr(reg12,2)
  'eval 'crab'.+7C 'aq' le(4) rexx(storage(claw)) char'
  if claw ^= "CLAW" then
  do
    "f c'CLAW' nobreak first print noterminal"
    do while rc = 0
      "evalsym x rexx(address(crab))"
      crab = d2x(x2d(crab) - 124)
      if right(crab,3) = "030" then leave
      "f c'CLAW' nobreak print noterminal"
    end
    if rc ^= 0 then
    do
      "note 'Crab not found.'"
      return 0
    end
  end
end

/* print the crab */
"cbf "crab". model(crabm)"
if rc ^= 0 then
do

```

```

    "note 'Crab not found.'"
    return 0
end
"note '"
"eq crab "crab"."
"eval crab+c "aq" le(4) rexx(storage(prv)) pointer"
prv = d2x(x2d(prv) - 56)
curprv = prv

/* print transient level */
"eval crab+b4c "aq" le(4) rexx(storage(ioep)) pointer"
"l "ioep" print noterminal"
"f c'LSCX056' print noterminal"
if rc = 0 then
do
    "evalsym x rexx(address(tranptr))"
    tranptr = d2x(x2d(tranptr) - 8)
    "eval "tranptr". "aq" le(8) rexx(storage(tranlevel)) char"
    "note 'Transient level - "tranlevel"'"
end

/* print the current prv */
"cbf "prv". model(prvm)"
if rc = 0 then "eq prv "prv"."
else
do
    "note '"
    "note 'PRV bad.'"
    "note '"
    return 0
end
"eval prv+10 "aq" le(4) rexx(storage(prvsize)) pointer"
"note '"
"note 'Current PRV Storage'"
"note '"
"l prv+38 l("x2d(prvsize)-56")"

/* print other prv headers */
"note '"
"note 'Shared PRVs'"
"eval crab+b3c "aq" le(4) rexx(storage(prv)) pointer"
do while x2d(prv) ^= 0
    if x2d(prv) ^= x2d(curprv) then "cbf "prv". model(prvm)"
    "eval "prv".+18 "aq" le(4) rexx(storage(prv)) pointer"
end
"note '"
"note 'Private PRVs'"
"eval crab+b40 "aq" le(4) rexx(storage(prv)) pointer"
do while x2d(prv) ^= 0
    if x2d(prv) ^= x2d(curprv) then "cbf "prv". model(prvm)"
    "eval "prv".+18 "aq" le(4) rexx(storage(prv)) pointer"
end

/* print stack info - CRAB+X'108' = CRABEAST */
"note '"
"note 'Stack'"
"eval crab+108 "aq" le(4) rexx(storage(att)) pointer"

```

```

call formatatt(att)

if inner=0 then
  'EQU X SCSaveX'
return 0

formatatt: procedure expose aq
arg att
"eval "att". "aq" le(4) rexx(storage(ath)) pointer"
"cbf "ath". model(athm)"
"eval "ath".+c "aq" le(4) rexx(storage(ash)) pointer"
do while x2d(ash) ^= 0
  "cbf "ash". model(ashm)"
  "eval "ash". "aq" le(4) rexx(storage(peast)) pointer"
  if x2d(peast) ^= 0 then call formatatt(peast)
  "eval "ash".+4 "aq" le(4) rexx(storage(ash)) pointer"
end
return

failure:
if inner=1 then
"NOTE '"tname'"("tver"): Please see Don Poitras for support.'" asis
"NOTE '          Storage is not available.'" asis
"NOTE '          Possible reasons: 1) CSA was not dumped.'" asis
"NOTE '          2) This is not a complete dump.'"
asis
return 4

halt:
exit: exit
  if ltcbptr^=0 then
    do
      backup = backup+1
      tcbptr = ltcbptr
    end
  else if backup=0 then
    do
      'eval 'tcbptr'+80 'aq' le(4) rexx(storage(ntcbptr)) pointer'
      if rc^=0 then signal failure
    end
  else
    do
      otcbptr=tcbptr
      tcbptr=0
      do until tcbptr^=0 | backup<=0
        backup = backup-1
        if otcbptr^=0 then
          do
            'eval 'otcbptr'+84 'aq' le(4) rexx(storage(otcbptr)) pointer'
            if rc^=0 then signal failure
          end
        if otcbptr^=0 then
          do
            'eval 'otcbptr'+80 'aq' le(4) rexx(storage(ntcbptr)) pointer'
            if rc^=0 then signal failure
            if ntcbptr^=0 then
              tcbptr=ntcbptr

```

end
end
end

PRVM.ASM

```
PRVM      CSECT
CBMODEL  BLSQMDEF BASELBL=PRVH,CBLEN=PRVHLEN,PREFIX=3,
X
          ACRONYM=PRV,ACROLBL=PRVPRV+1,HEADER=PRV
BLSQMFLD NAME=PRVPRV,DTYPE=EBCDIC
BLSQMFLD NAME=PRVCOG
BLSQMFLD NAME=PRVMODNM,DTYPE=EBCDIC
BLSQMFLD NAME=PRVSIZE
BLSQMFLD NAME=PRVFWD
BLSQMFLD NAME=PRVBKWD
BLSQMFLD NAME=PRVFLAGS
BLSQMFLD NAME=PRVUSERS
BLSQMFLD NAME=PRVFPTR
BLSQMFLD NAME=PRVSY,NEWLINE
BLSQMFLD NAME=PRVLKEP,NEWLINE
BLSQMFLD NAME=PRVDESTR
BLSQMDEF END
USING PRVH,12
COPY PRVH
END
```

PRVM.JCL

```
//ASM      EXEC,PGM=ASMA90,PARM='OBJECT,TEST',
//SYSPRINT DD SYSOUT=*
//SYSLIB   DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.AMODGEN,DISP=SHR
//         DD DSN=LSD.LC700.MACLIB,DISP=SHR
//SYSTEM   DD SYSOUT=*
//SYSPUNCH DD DUMMY
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLIN   DD DSN=SASDTP.TEST.OBJ(PRVM),DISP=SHR
//SYSIN    DD DSN=SASDTP.IPCS.SOURCE(PRVM),DISP=SHR
//LINK     EXEC PGM=IEWL,PARM='LIST,MAP,XREF,LET,RENT,REFR,TEST'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD  DD DSN=SASDTP.TEST.LOAD,DISP=SHR
//SYSLIN   DD DSN=SASDTP.TEST.OBJ(PRVM),DISP=SHR
//         DD DDNAME=SYSIN
//SYSIN    DD *
          MODE AMODE(24),RMODE(24)
          NAME PRVM(R)
/*
```

IPCS CLIST SCDINFO Output

scdinfo(v0.2):

*** SAS/C Diagnostic Worksheet ***

Source dataset: SASDTP.SYSMDUMP
Address Space: ASID('02D7')
Dump Time: 03/03/2000 09:06:30.853958
Dump Requestor: SYSMDUMP
Original Dump Dataset: SASDTP.SYSMDUMP
Dumping Program: SYSMDUMP

PSW at time of error: 078D3000 8F00F260

Registers at time of error:

R0:0F037FA0	R1:8F02E600	R2:00000008	R3:0001DFF8
R4:0F033470	R5:0F033218	R6:0F00F260	R7:0F00F258
R8:0F0334D8	R9:0F002548	R10:00019FC8	R11:00017BF8
R12:80006030	R13:0001D0B8	R14:8F0332EE	R15:0F00F260

IEA11015I The requested ALETs are zero.

TCB Summary for ASID x'02D7' ASCB: 00F4C280

TCB:	007FE1D8	CMP=00000000	RB	OPSW=070C1000	81478E90	PRBEP=IEAVAR00
TCB:	007FF1D8	CMP=00000000	RB	OPSW=070C2000	85BEEC9C	PRBEP=IEAVTSDT
TCB:	007FDE48	CMP=00000000	RB	OPSW=070C1000	80DBD624	PRBEP=IEEPRWI2
TCB:	007EE620	CMP=00000000	RB	OPSW=070C1000	80DBD624	PRBEP=IEFIIC
*TCB:	007EE140	CMP=840C4000	RB	OPSW=070C1000	85C18494	PRBEP=SHR2MAN

Summary for TCB: 007EE140

Job: SHR2MAN Step: TEST1

DDNAME	DATASETNAME
-----	-----
STEPLIB	SASDTP.TEST.LOAD
CTRANS	LSD.LC650.LOAD

Load List

CDE:	007EE070	Name=SHR2DYN	EP=8F03381C	Use	Count=1
CDE:	00B5DC80	Name=IGG019DK	EP=00DE41B0	Use	Count=0
CDE:	007EE0C0	Name=LSCDIND	EP=00024EFC	Use	Count=1
CDE:	007EE100	Name=LSCOSEQ	EP=8F032C3C	Use	Count=1
CDE:	007F1778	Name=LSCABTRX	EP=00019FC8	Use	Count=1
CDE:	007F17E0	Name=LSCDMGR	EP=00014FF0	Use	Count=1
CDE:	007F1830	Name=LSCOTERM	EP=0000EE4C	Use	Count=1
CDE:	007FE110	Name=LSCSIO	EP=8F022A74	Use	Count=1
CDE:	00B4E168	Name=IGG019BB	EP=00CBB970	Use	Count=0
CDE:	00B62D48	Name=IGG019BA	EP=00B79C80	Use	Count=0
CDE:	00B5D4B0	Name=IGG0193B	EP=00D20C08	Use	Count=0

Job Pack Area

CDE:	007EE070	Name=SHR2DYN	EP=8F03381C	Use	Count=1
CDE:	007EE0C0	Name=LSCDIND	EP=00024EFC	Use	Count=1

CDE: 007EE100 Name=LSCOSEQ EP=8F032C3C Use Count=1
 CDE: 007F1778 Name=LSCABTRX EP=00019FC8 Use Count=1
 CDE: 007F17E0 Name=LSCDMGR EP=00014FF0 Use Count=1
 CDE: 007F1830 Name=LSCOTERM EP=0000EE4C Use Count=1
 CDE: 007FE110 Name=LSCSIO EP=8F022A74 Use Count=1
 CDE: 007FF0A0 Name=SHR2MAN EP=8F002090 Use Count=1

CRAB: 00006030

```
+000C PRV..... 0F037B98
+0038 DWK..... 47F0E000 00008F00
+004C FLGC..... 00 FLGA..... 00 FLGM..... 00
+0060 SVCA..... 8F01AF7E LVER..... 98061C6E CDSA..... 0001D0B8
+0104 STOP..... 0001D148 EAST..... 0001DFF8 MDSA..... 00007028
+0124 FLG0..... 34 FLG1..... 00 FLG2..... 20
+0127 FLG3..... 00 CXXG..... 00000000 FKCT..... 0000
+01DC INTH..... 00006034 PIQH..... 00000000 PIQT..... 00000000
+01F8 ENBL..... FEF24200 FC000000
+0200 DFLH..... FFFFFFFF FFFFFFFF
+0214 FLG4..... 00 CRIT..... 0000 SGNO..... 00
+0219 SGNM..... DMCB..... 00017BF8 FLG5..... 00
+0253 FLG6..... 00 BIQH..... 00000000 BIQT..... 00006288
+0260 BIQN..... 00000000 PSEV..... 00000000 FLG7..... 80
+0273 FLG8..... 80 SGEP..... 0F008368 SGPR..... 00000000
+0294 MSEG..... 00007000 NEED..... 00000000 COMX..... 0001
+02BE COPN..... 0001 SRGS..... 00000000 SPND..... 00
+02DC GIGN..... 010DBDFF 03FFFFFF
+02E4 GDFL..... FFFFFFFF FFFFFFFF
+02ED FLG9..... 00 ASAC..... 0000 DOMN..... 00000000
+02FC NEST..... 0000 AXE..... 00000000 FAXE..... 00000000
+0340 FLGO..... 00 FLGU..... 00
```

```
---00--- ---01--- ---02--- ---03--- ---04--- ---05---
FKBT FKBT FKBT FKBT FKBT FKBT
-----
000 0F03AE60 0F03AE40 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
001 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 0F03A9A8
002 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
003 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
004 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
005 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 0F03A000 00000000
```

```
---00--- ---01--- ---02--- ---03--- ---04--- ---05---
FHBT FHBT FHBT FHBT FHBT FHBT
-----
000 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
001 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
002 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
003 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
004 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
005 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
006 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
007 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
008 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
009 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
00A FFFFFFFF 00000000 00000000 00000000 00000000 00000000
```

+0B04	MPRV.....	0F00F1F8	ASP1.....	00000000	AST1.....	00000000
+0B10	FST1.....	00000000	AHWM.....	00002000	ALTM.....	00000000
+0B1C	HHAM.....	00001000	HHUM.....	00000740	CHAM.....	00001000
+0B28	CHUM.....	00000678	HAT.....	0F00F0C0	HPSZ.....	00000000
+0B3C	PRVS.....	0F037B60	PRVP.....	0F00F1C0	IOEP.....	0F018C60
+0B50	IOPR.....	0F02A6F0	CNTX.....	0F03AC58	AMDS.....	80023040
+0B6C	BAIT.....	00000000	JORG.....	00000000	HGCT.....	00000002
+0B9C	HFCT.....	00000000	IHAT.....	00000000	IHPS.....	00000000
+0BA8	IHPE.....	00000000	TIPA.....	00000000	AOWA.....	0F02C388
+0BF0	SIMP.....	ZOEM.....	00000000	USOH.....	00008F00
+0BFC	LSCP.....	...	EVNM.....	MVS/SP	EVVN.....	06
+0F09	EVRN.....	00	EVML.....	06	EVSM.....	40
+0F0C	SENM.....	SERN.....	00	SEML.....	00
+0F17	SESM.....	00	EVNO.....	01	EVSF.....	C0
+0F1A	EVXA.....	C1	EVTS.....	00	EVS1.....	FC
+0F20	OEVN.....	03	EDBV.....	00	PSEQ.....	00000000
+0F34	PNM.....				
+0F44	PVN.....	0000	PRN.....	0000	PML.....	0000

TRANSIENT LEVEL - 98061C6E

PRV: 0F037B60

+0000	PRV.....	PRV	COG.....	00000000	MODNM....	SHR2DYN
+0010	SIZE.....	00000499	FWD.....	0F02A190	BKWD.....	00006B6C
+0020	FLAGS....	00	USERS....	0001	FPTR.....	0F0334C8
+0028	SYS.....	00000000		00000000		
+0030	LKEP.....	0F03381C	DESTR....	00000000		

CURRENT PRV STORAGE

```

0F037B98.                0F00F440 00000000 |           ..4 .... |
0F037BA0 LENGTH(256)==>All bytes contain X'00'
0F037CA0. 0F00F8C8 00000000 00000000 00000000 | ..8H..... |
0F037CB0. 000061F0 00000000 000061F4 00000000 | .. /0..... /4.... |
0F037CC0. 000072A0 00000000 000070E4 00000000 | .....U.... |
0F037CD0 LENGTH(800)==>All bytes contain X'00'
0F037FF0. 00000000 00000000 00                | ..... |

```

SHARED PRVS

PRV: 0F02A190

+0000	PRV.....	PRV	COG.....	00000000	MODNM....	LSCDIND
+0010	SIZE.....	00000118	FWD.....	0F02A2A8	BKWD.....	0F037B78
+0020	FLAGS....	00	USERS....	0001	FPTR.....	0F02A2A0
+0028	SYS.....	00000000		8F01327A		
+0030	LKEP.....	00024EFC	DESTR....	00000000		

PRV: 0F02A2A8

+0000	PRV.....	PRV	COG.....	00000000	MODNM....	LSCOSEQ
+0010	SIZE.....	0000017C	FWD.....	0F00F080	BKWD.....	0F02A1A8
+0020	FLAGS....	00	USERS....	0001	FPTR.....	0F02A318
+0028	SYS.....	00000000		00000000		
+0030	LKEP.....	0F032C3C	DESTR....	00000000		

PRV: 0F00F080

+0000 PRV..... PRV COG..... 00000000 MODNM.... LSCDMGR
+0010 SIZE..... 00000040 FWD..... 0F02A428 BKWD..... 0F02A2C0
+0020 FLAGS.... 00 USERS.... 0001 FPTR..... 0F00F0B8
+0028 SYS..... 4BF5F0C3 404D9985
+0030 LKEP..... 00014FF0 DESTR.... 00000000

PRV: 0F02A428

+0000 PRV..... PRV COG..... 00000000 MODNM.... LSCOTERM
+0010 SIZE..... 00000290 FWD..... 0F02A6B8 BKWD..... 0F00F098
+0020 FLAGS.... 00 USERS.... 0001 FPTR..... 0F02A6B0
+0028 SYS..... 00000000 00000000
+0030 LKEP..... 0000EE4C DESTR.... 00000000

PRV: 0F02A6B8

+0000 PRV..... PRV COG..... 00000000 MODNM.... LSCSIO
+0010 SIZE..... 00003941 FWD..... 00000000 BKWD..... 0F02A440
+0020 FLAGS.... 00 USERS.... 0001 FPTR..... 0F02DD68
+0028 SYS..... 00000000 00000000
+0030 LKEP..... 0F022A74 DESTR.... 00000000

PRIVATE PRVS

PRV: 0F00F1C0

+0000 PRV..... PRV COG..... 00000000 MODNM....
+0010 SIZE..... 00000E39 FWD..... 00000000 BKWD..... 00006B70
+0020 FLAGS.... 80 USERS.... 0001 FPTR..... 00000000
+0028 SYS..... 00000000 00000000
+0030 LKEP..... 00000000 DESTR.... 00000000

STACK

ATH: 0001D000

+0000 NEXT..... 00000000 SIZE..... 00000FE0 LEFT..... 00000FE0
+000C LSEG..... 0001D018 FLAGS.... 48

ASH: 0001D018

+0000 PEAST.... 00007FF8 PSEG..... 00000000 FLAGS.... 00

ATH: 00007000

+0000 NEXT..... 00000000 SIZE..... 00000FE0 LEFT..... 00000D40
+000C LSEG..... 00007018 FLAGS.... 28

ASH: 00007018

+0000 PEAST.... 00000000 PSEG..... 00000000 FLAGS.... 00

Traceback

LSCX056 SAS/C library release 6.50C (resident), release 6.50C
(transient).

LSCX041 **** ERROR ****

ABEND occurred in line 56 of FUNC2(SHR2DYN),offset 0000D4

Program terminated by operating system. ABEND code = S0C4

Calling trace:

Function	Line	Offset	Context
FUNC2(SHR2DYN)	56	0000D4	
MAIN(SHR2MAN)	43	000158	

SYSLOG

J E S 2 J O B L O G -- S Y S T E M P R O D -- N O D E S D C M V S

```
15.46.25 JOB14604 ---- THURSDAY, 02 MAR 2000 ----
15.46.25 JOB14604 IRR010I USERID SASDTP IS ASSIGNED TO THIS JOB.
15.47.03 JOB14604 ICH70001I SASDTP LAST ACCESS AT 15:18:01 ON THURSDAY, MARCH 2, 2000
15.47.03 JOB14604 $HASP373 SHR2MAN STARTED - INIT 9 - CLASS T - SYS PROD
15.47.03 JOB14604 IEF403I SHR2MAN - STARTED
15.47.10 JOB14604 +LSCX064 ABEND diagnostic messages directed to ddname SYSTEM
15.47.15 JOB14604 $HASP375 SHR2MAN ESTIMATED LINES EXCEEDED
15.47.15 JOB14604 IEA995I SYMPTOM DUMP OUTPUT
SYSTEM COMPLETION CODE=0C4 REASON CODE=00000004
TIME=15.47.10 SEQ=41909 CPU=0000 ASID=02D7
PSW AT TIME OF ERROR 078D3000 8F00F260 ILC 0 INTC 04
NO ACTIVE MODULE FOUND
NAME=UNKNOWN
DATA AT PSW 0F00F25A - 00080F00 F2600F00 0A180F00
GPR 0-3 0F037FA0 8F02E600 00000008 0001DFF8
GPR 4-7 0F033470 0F033218 0F00F260 0F00F258
GPR 8-11 0F0334D8 0F002548 00019FC8 00017BF8
GPR 12-15 80006030 0001D0B8 8F0332EE 0F00F260
END OF SYMPTOM DUMP
15.47.15 JOB14604 IEF450I SHR2MAN TEST1 - ABEND=S0C4 U0000 REASON=00000004
15.47.15 JOB14604 IEFACTRT005I: STEP TEST1 ENDED -----
RETURN CODE S0C4
15.47.15 JOB14604 IEF404I SHR2MAN - ENDED
15.47.15 JOB14604 $HASP395 SHR2MAN ENDED
----- JES2 JOB STATISTICS -----
02 MAR 2000 JOB EXECUTION DATE
20 CARDS READ
13,000 SYSOUT PRINT RECORDS
0 SYSOUT PUNCH RECORDS
1,496 SYSOUT SPOOL KBYTES
0.21 MINUTES EXECUTION TIME
```

How to setup and use ASMIDF on MVS.

ASMIDF can also be used in CMS. There are some differences in setup, but once running they're almost identical.

Part 1. - setup

First create some new datasets.

userid.ASMIDF.REXX - allocated to DDNAME=ASM. This PDS contains REXX execs that can be used while running the debugger. The member PROFILE will be run at startup. My version of PROFILE does things like making HEX the default for display and entry and orders the registers in a more normal looking layout than the default.

userid.ASMLANGX - allocate to DDNAME=ASMLANGX. This PDS contains the output of the ASMLANGX program. LANGX stands for language extraction. These files are where the source statements are stored for use in source level debugging. When debugging C code, you won't be using anything from here. The process to create the members that are in here is described below.

userid.ADATA - allocate to DDNAME=SYSADATA. This is an intermediate file produced by the assembler and is input to the ASMLANGX program. This is a flat file that will be overwritten every time you run the assembler, so be sure to run ASMLANGX immediately after assembling if you want to be able to use the source in a debug session.

Next, you need to add some HLASM datasets to your ISPF allocations. Either update your logon clist, or logon jcl or however you do this kind of thing.

DDNAME	Add this dataset
--------	------------------

ISPMLIB	SYS1.SBLSMSG0
ISPPLIB	SYS1.SBLSPNL0
ISPSLIB	SYS1.SBLSKEL0
ISPTLIB	SYS1.SBLSTBL0
STEPLIB	ASMT.SASMMOD2

This last DD is problematic in that you can't run the debugger using SVC 97 unless it's there. See:

<http://www.s390.ibm.com:80/bookmgr-cgi/bookmgr.cmd/BOOKS/ASMTIU02/2%2e3%2e2?SHELF=ASMSH006>

A way around this problem is to use the TSOLIB command. In my TERMPROF CLIST, I have the following:

```
TSOLIB ACT FILE(DONDBG)
```

DONDBG is a DD that I had allocated in my logon JCL proc, but could have

just as easily been done in the TERMPROF. The ASMT... dataset and the program you're debugging need to be accessed in this fashion. I'd recommend doing the following in your TERMPROF:

```
ALLOC F(xxxDBG) DS('ASMT.SASMMOD2' 'userid.TEST.LOAD') SHR REUSE
TSOLIB ACT FILE(xxxDBG)
```

Part 2. - creating debuggable input

When assembling a module that you'd like to debug at the source level use the ADATA assembler option and add an SYSADATA DD to your job. Like so:

```
//ASM      EXEC PGM=ASMA90,PARM='OBJECT,ADATA'
...
//SYSADATA DD DISP=SHR,DSN=userid.ADATA
```

After assembly run the ASMLANGX program to extract the source level information from the ADATA file produced by the assembler. For example, if I had a csect HELLO that I wanted to debug, I would issue

```
ASMLANGX HELLO (ASM LOUD ERROR
```

You don't need the extra parms, they just give some info to tell you that the process worked. The minimum required is:

```
ASMLANGX csectname
```

This creates a member in the userid.ASMLANX dataset called "csectname".

When running the debugger, either "statement step" or "lan load csectname" will read this file to show source statements.

Part 2. - Running the debugger

at a READY prompt, from ISPF option 6 or as an ISPF TSO command:

```
TSOEXEC ASMIDF PROGRAMNAME (asmidfoptions/programoptions
```

The "PROGRAMNAME" must be able to run under TSO. If the program needs DD's then run the debugger from a clist, allocating the needed DD's before invoking ASMIDF.

ASMIDF Rexx Monitor Program

```
/* REXX - written by George Wilder */
Parse Arg address count rest
if address='' then
do
  'SET MSG ADDRESS not provided'
  exit
end
if count= '' then
do
  'SET MSG MONITOR not provided'
  exit
end
'EXTRACT location 4 X''address''''
if RC ^= 0 then
do
  'SET MSG Extract failed ' RC
  'SET ALARM'
  exit
end
memstr=c2x(MEMAREA)
'SET OPTIONS ON TRACEALL'
do while 1
  if memstr = count then
  do
    'MSTEP'
    'EXTRACT EVENT'
    'EXTRACT location 4 X''address''''
    if RC ^= 0 then
    do
      'SET MSG Extract failed ' RC
      'SET ALARM'
      exit
    end
    memstr=c2x(MEMAREA)
  end
else
  do
    say 'MEMORY='memstr':MONITOR='count
    'EXTRACT REGS'
    BPOINT=SUBSTR(OPSW,9)
    say 'SET BREAK ON 'BPOINT
    'set break ON 'BPOINT
    BPOINT=SUBSTR(PSW,9)
    say 'SET BREAK ON 'BPOINT
    'set break ON 'BPOINT
    exit
  end
end
end
```