HP.SAPMD GSE SOFTWARE LISTINGS (Cont'd.)

(NASA-CR-172119-Pt-2) STAND ALONE PRESSURE MEASUREMENT DEVICE (SAPMD) FOR THE SPACE SHUTTLE ORBITER, PART 2 (Southwest Research Inst.) 78 p

N90-16203

Unclass

0234605
SECTION 2
**DUMP**

Dump routines.

```
#include <supglob.c> /* get globals */

/*
 * Dump a block of SC-1 memory in the specified window in byte format.
 * The address is in dseg|doff.
 */

const window *id;
int i, off;
off=id->topoff;
id->cury=1;
for (i=3;i<=id->lines;i++)
    [dumpln(id,id->daseg,off);
     off+=16;
     id->cury++;
     id->cury=1;]
```

**UBYTES**

Fill the specified window with unassembled instructions.

```
const window *id;
int i;
clear(id);
id->cury=1;
id->daoff=id->topoff;
for (i=3;i<=id->lines;i++)
    [diss(id,id->daseg,id->daoff);
     id->daoff+=ip;
     id->cury+=1;]
id->type=DA;
```

**PADDR**

Print the passed segment and offset.

```
paddr(id,seg,off)
struct window *id;
int seg,
    off;
[hexw(id,seg);
 wchw(id,':');
 hexw(id,off);]```
HEXW

Print the passed word in hex.

hexw(id,x)
struct window *id;
int x;
{hex(id,x>>8);
  hex(id,x);}

HEXC

Convert the passed nibble to hex ascii.

char hexc(x)
int x;
{x=x&0xf;
 return((x<=9)?x+'0':x-10+'A');}

HEX

Print the specified byte at the current cursor position on the specified window.

hex(id,x)
struct window *id;
int x;
{wchw(id,hexc(x)>>4));
  wchw(id,hexc(x));}

DUMPUP

Scroll the specified dump window in response to an up arrow.

dumpup(id)
struct window *id;
{dscroll(id);
  id->topoff-=16;
  id->cury=1;
  dumpln(id,id->daseg,id->topoff);}

DUMPDN

Scroll the specified dump window in response to a down arrow.

dumpdn(id)
```c
/* scroll up */
struct window *id;
{uscroll(id);
id->topoff+=16;
id->cury=id->lines-2;
dumpln(id,id->daseg,id->topoff+(id->lines-3)*16);} /* dump a line */

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

 Scroll the specified dump window in response to a down arrow.
******************************************************************************

undn(id)
struct window *id;
[int i,
j,
k;
uscroll(id);
id->cury=id->lines-2;
diss(id,id->daseg,id->daoff);
id->daoff+=ip;
i=0;
for (j=0;j<4;j++)
{movcurs(id->scry+l,6+j);
k=sch();
k=(k<9)?k-0':k-'A'+10;
i=(i<<4)+k;}
id->topoff=i;

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

 Dump a line of SC-I memory.
******************************************************************************

dumpln(id,dseg,off)
struct window *id;
int dseg,
off;
{unsigned char dmp[16];
int j,
i;
if (!scdump(dseg,off,17)) return(0);
for (j=0;j<15;j++) dmp[j]=rdsc1();
paddr(id,dseg,off);
wchw(id,' ');
if (id->type==DBT)
{for (j=0;j<15;j++)
{hex(id,dmp[i]);
if (dmp[j]<1 || dmp[j]>=0x7f) dmp[j]='.'; /* for printing */
if (j==7)
wchw(id,'-');
else
wchw(id,' ');}}}
else
for (j=0;j<15;j++)
{hexw(id,(dmp[j]<<8)|dmp[j]);
for (i=j;i<=j+1;i++)
{if (dmp[i]<1 || dmp[i]>=0x7f) dmp[i]='.'; /* for printing */
```
wchw(id,'');} /* space between words */

wchw(id,'*');
for (j=0;j<=15;j++)
  wchw(id,dmp[j]);
wchw(id,'*');
if (rdscl()!=PROMPT) error(BADSCl); /* trailing star */

/********************W************************
DMP REG S
Dump registers to screen window.
**************

dmpregs()
  {int i;
   char r;
   wrscI(DREGS);
   if (!gregs()) error(BADSCl);
   else
     if (!dregs()) error(BADSCl);}

TRACE
Trace execution (single-step).
**************

trace(token)
  int token;
  {wrscl(STEP);
   gregs();
   dregs();
   wchs(CR);
   diss(&screen,sclregs[CS],sclregs[IP]);
   if (token==TU)
     {if (!mkwmd()) return(0);
      activw->daseg=sclregs[CS];
      activw->topoff=sclregs[IP];
      ubytes(activw);}};

GO
Start execution until optional breakpoint.
**************

go(token)
  int token;
  {if (bf) brkpt(gopoop[2],gopoop[3]);
   if (af)
     {lreg(IP,gopoop[1]);
      lreg(CS,gopoop[0]);}
   wrscI(GO);
   gregs();
   if (bf) loadscl(gopoop[2],gopoop[3],1,&bpinst); /* restore instruction */
   dregs();
   wchs(CR);
   diss(&screen,sclregs[CS],sclregs[IP]);
   /* display registers */
   /* new line */
   /* display next instruction */
if (token==GU) {
    if (!mkwnd()) return(0);
    activw->daseg=sclregs[CS];
    activw->topoff=sclregs[IP];
    ubytes(activw);
}

/* trace & unassemble? */
/* try to make one */
/* ... yep, set dump address */
/* ... */
/* disassemble */

().'/**------------------------------------------*/
/* */
/* */
/* */
/* */
/* */
/* */

BRKPT

btrip(seg,off)
int seg,
off;
[static unsigned char trap=0xcc;
if (!scldump(seg,off,2)) return(0);
bpinst=rdscl();
if (rdscl())!=PROMPT)
    {error(BADSC1);
     return(0);};
bpact=1;
loadscl(seg,off,1,&trap);}

set a breakpoint
* address segment
* ... and offset.
* breakpoint instruction
* send dump command
* get replaced instruction
* check for good dump
* send error
* return
* flag active breakpoint
* set breakpoint

слуш

loadscl(seg,off,len,poop)
int seg,
off,
len;
char *poop;
[wrscl(LOAD);
saddr(seg,off,len);
for (;len>0;len--) wrscl(*poop++);
if (rdscl())!=PROMPT) error(BADSC1);]

* load target memory
* address segment ...
* ... and offset ...
* byte count
* data address
* send load memory command
* send address
* load data
* good load?

слуш

GREGS

gregs()
[int i,
    r;
if (rdscl())!=DUMPREG) return(0);
for (i=0;i<14;i++)
    {r=rdscl();
     sclregs[i]=r|(rdscl())<<8};
    if (rdscl())!=PROMPT) return(0);
    return(1);]

* get registers
* iteration variable
* register temp
* return error
* print regs.
* get low reg half
* get register high half
* get prompt
* return success

слуш

DREGS
/* Display registers from SC-1. */
/* */
dregs()
{ /* display registers */
  int i; /* iteration variable */
  static char rtext[]={"AXBXCXDSIDIBPSPCSDSESSSIPFL"}; /* register names */
  wchs(CR); /* scroll */
  for (i=0;i<28;i+=2) /* print regs. */
    {wchs(rtext[i]); /* write reg. name */
     wchs(rtext[i+1]); /* ... */
     wchs('('); /* print register */
     hexw(&screen,sclregs[i>>1]); /* space between registers */
     wchs(' '); /* space between registers */
     wchs(' '); /* new line in middle */
     if (i==12) wchs(CR); /* good register dump */
     }
  return(1);}
/* */
/* */
S C D U M P
/* */
/* Send command to dump a block of SC-1 memory. */
/* */
/* */
scdump(seg,off,len) int seg, off, len;
{wrsc1(DMEM);
  saddr(seg,off,len);
  if (rdsc1()!=DUMPMEM) /* send dump command to SC1 */
   {error(BADSC1);
    return(0);}
  return(1);}
/* */
/* */
S A D D R
/* */
/* Send memory address to SC-1. */
/* */
saddr(seg,off,len) int seg, off, len;
{wrsc1(seg);
  wrsc1(seg>>8);
  wrsc1(off);
  wrsc1(off>>8);
  wrsc1(len);}
/* */
/* */
L R E G
/* */
/* Load an SC-1 register. */
/* */
lreg(reg,x) /* load register */
{int reg, /* register # */
  x; /* data */
wndir() /* window directory */
{static char *title= /* directory title */
["Window Type Addr Lines Window Type Addr Lines"]; /* iteration variable */
int i; /* new line */
wcsh(CR); /* print title */
stype(title); /* scan windows */
for (i=0;i<4;i++) /* new line */
[wchs(CR); /* display window description*/
 wndxex(i); /* space to next column */
 stype(" "); /* next column */
 wndxex(i+4);]}

wndex(i)
{ /* display window poop */
int i; /* window # */
[static char *wtype[]={" " DB "," DW "," DA "}; /* type text */
stype(" "); /* space */
wchs(box[i].wnum); /* print window # */
stype(" "); /* space */
if (box[i].used) /* window in use? */
 stype(wtype[box[i].type]);
 else /* print type */
 stype("Free"); /* window unused */
 stype(" "); /* next column */
 if (box[i].used==0) /* space */
 stype(" "); /* window filled? */
 else /* window filled */
 [if (box[i].type==BL) /* nope blank it */
  stype(" "); /* window filled */
  else /* window filled */
  [hexw(&screen,box[i].daseg); /* print string */
   wchs(;''); /* text there */
   hexw(&screen,box[i].topoff);]
  stype(" "); /* type address */
  hex(&screen,box[i].lines-2);]
}
/* ERROR */
/* Print the specified error message on line 23 of the screen. */

#include <supglob.c> /* locate global data */

error(err) /* print error message */
    int err; /* message id */
    [static char *msgs[]=/* message text */
        "Pressure file recovered", /* 0 */
        "SAPMD communication error", /* 1 */
        "Checksum error", /* 2 */
        "Bad command", /* 3 */
        "Error in hex file", /* 4 */
        "Bad input character", /* 5 */
        "No room for window", /* 6 */
        "Strange SC-1 response", /* 7 */
        "Can't find file", /* 8 */
        "Unexpected end-of-file", /* 9 */
        "No filename specified", /* 10 */
        "Error in calibration file (SAPMD.CAL)", /* 11 */
        "Missing launch simulation file (LA.CMD)", /* 12 */
        "error 13", /* message header */
        "error 14", /* */
        "error 15", /* */
        "error 16", /* */
        "error 17", /* */
        "error 18", /* */
        "error 19", /* */
        "error 20", /* */
        "error 21", /* */
        "error 22", /* */
        "error 23"],
    *arrow="---> "; /* message header */
    wchs(CR); /* write character */
    stype(*arrow); /* print arrow */
    stype(msgs[err]); /* print message */

/* */
/* */
/* */
/* S T Y P E */
/* Print the passed line of text on screen window. */

stype(txt) /* print character string */
    char *txt; /* string pointer */
    [char *ptr; /* iteration pointer */
        for (ptr=txt;*ptr!='\0';ptr++) wchs(*ptr);] /* print arrow */
/* */
/* */
/* */
/* W T Y P E */
/* Print the passed line of text on window. */

wtype(id,txt) /* print character string */
    struct window *id; /* window id */
char *txt; /* string pointer */
char *ptr; /* iteration pointer */
for (ptr=txt; *ptr != '\0'; ptr++) wchw(id, *ptr); /* print arrow */
/**
*******************************/
#define MAIN 1
#include <supglob.c>
main(argc,argv)
int argc;
char *argv[];
{struct window *id,*x;
 int i,j;
erase(activw);
m1=createw(0,11);
m2=createw(1,13);
m3=createw(2,13);
m4=createw(3,1);
m5=createw(4,10);
m6=createw(5,24);
debug(argc,argv);}

/*/ MENU */

Display menus on windows.

#include <supglob.c>

menu(n,e)
int n,
e;
[static char *mltext[]={
"", "",
"1. COMMAND/INTERROGATE SAPMD",
"2. SAPMD SELF-TEST",
"3. RECOVER PRESSURE DATA filename",
"4. DISPLAY PRESSURE DATA filename",
"5. PRINT PRESSURE DATA filename",
"",0};

static char *m2text[]={
"SG ddd/hh:mm:ss",
"SM ddd/hh:mm:ss",
"TM",
"DR xx[yy]",
"DS xx",
"DE xxx[yy]",
"ER xx",
"ES xx",
"EE xxxx",
"P filename[xxxx]",
"MON",
"",
"", "",
"1. HP/SAPMD ACCESS",
"2. HP/SAPMD SELF-TEST",
"3. COMMAND/INTERROGATE SAPMD",
"4. SAPMD SELF-TEST",
"5. RECOVER PRESSURE DATA filename",
"6. DISPLAY PRESSURE DATA filename",
"7. PRINT PRESSURE DATA filename",
"",0};

static char *m3text[]={
"SG ddd/hh:mm:ss",
"SM ddd/hh:mm:ss",
"TM",
"DR xx[yy]",
"DS xx",
"DE xxx[yy]",
"ER xx",
"ES xx",
"EE xxxx",
"P filename[xxxx]",
"MON",
"",
"", "",
"1. HP/SAPMD ACCESS",
"2. HP/SAPMD SELF-TEST",
"3. COMMAND/INTERROGATE SAPMD",
"4. SAPMD SELF-TEST",
"5. RECOVER PRESSURE DATA filename",
"6. DISPLAY PRESSURE DATA filename",
"7. PRINT PRESSURE DATA filename",
"",0];

removw(activw);
if (e) erase(&screen);
switch (n)
{
case 1:
    show(ml);
    pmenu(ml,mltext,1,20);
    activw=ml;
    break;
case 2:
    show(m2);
    pmenu(m2,m2text,1,5);
    activw=m2;
    break;
case 3:
    show(m3);
    pmenu(m3,m3text,1,25);
    activw=m3;
    break;
case 4: /* remove current window */
    /* clear screen */
    /* which menu? */
    /* main menu */
    /* display main menu */
    /* display text */
    /* flag active window */
    /* next */
    /* command/interrogate */
    /* display menu */
    /* display menu text */
    /* no window active */
    /* next */
    /* self-test */
    /* display self-test menu */
    /* print text */
    /* flag active window */
    /* next */
    /* monitor window */
show(m5);
activw=m5;
break;
case 5:
    show(m6);
    activw=m6;
}

/******************

PMENU

Print menu text on screen.

**********

pmenu(id,text,line,col)
    struct window *id;
    char *text[];
    int line,col;
    {int i;
        char *ln;
        id->cury=line;
        for (i=0;(ln=text[i])!=0;i++)
            {id->curx=col;
                while (*ln!='\0') wchw(id,*ln++);
                id->cury++;}}

}
/*******************************************************************************/
/*
*/
/* P R P R E S S
*/
/*
* Print pressure data file.
*/
/*
*******************************************************************************/
#include <supglob.c>
/*
* locate global data
*/
prpress()
{
    int i,
pnum;
    if (!getfile()) return(1);
    freopen("sapmd.lst","w",stdout);
    for (pnum=0;&prsam[pnum]<samptr;pnum+=80)
       ptpage(pnum);
    freopen("prn","w",stdout);
    for (pnum=0;&prsam[pnum]<samptr;pnum+=80)
       ptpage(pnum);
    freopen("con","w",stdout);
    return(0);
}
/*******************************************************************************/
/*
*/
/* P T P A G E
*/
/*
* Print a page of pressure data.
*/
/*******************************************************************************/
ptpage(sm)
{
    int sm;
    {static char *headr=
     "SAMPLE PRESSURE",
     static char *title=
     " SAPMD PRESSURE DATA",
     static char *space=" ",
     int i,
     j,
     k;
    fputc(FF,stdout);
    printf("%s%s%d\n\n\n",iptr,title,prsam[sm].serial,headr);
    for (j=0;j<40;j++)
    {psam(sm);
     printf("%s"," ");
     psam(sm+40);
     printf("\n");
     sm++;}}
}
/*******************************************************************************/
/*
*/
/* P S A M
*/
/*
* Print a sample.
*/
/*******************************************************************************/
psam(sm)
{
    int sm;
    if (&prsam[sm]<samptr)
        printf(" %3d %5.2f",prsam[sm].sample,)
    prsam[sm].press);
    }
/** T E S T **/ 
/*
/* RECOVER
/*
/* Retrieve pressure data.
/*
/******

#include <supglob.c>
#define BUFSIZE 20000

recover()
  [FILE *pfile;
   union [int i;
      unsigned char b[5];] s;
   unsigned char rch,
       rchbuf[BUFSIZE];

   int i,
   n,
   cks;

   prompt("ENTER FILENAME: ");
i=rdln();
   if (i==HOME || i==LEFT) return(1);
skbl();
   if (*iptr==CR) return(1);
   for (i=0;i<sizeof(line);i++)
      if (line[i]==CR) line[i]=0;
   pfile=fopen(iptr,"wb");
   if (sacmd(DUMPRESS,0,0))
   if (!versg(PRESSFILE))
   [cks=0;
    n = 0;
    while (1)
      [for (i=0;i<2;i++)
        [if ((rch=rdsg())==EOPDATA) goto 11; /* done? */
          if (rch==ABORT)
            [p_error(13);
            fwrite(rchbuf,sizeof(rch),n,pfile); /* write data */
            fclose(pfile);
            return(1);];

          if (n < BUFSIZE)
          [rchbuf[n] = rch;
           n++;
          ]
        ]
      ]
    else
      [p_error(14);
        fwrite(rchbuf,sizeof(rch),n,pfile); /* write data */
        fclose(pfile);
        return(1);
      ];

   /* Next line no longer used */
   [fputc(rch,pfile);
    s.b[i]=rch;]
   s.b[2]='$0';
   cks+=bhex(&s);];

11:
   fwrite(rchbuf,sizeof(rch),n,pfile);/* write data buffer */
   fclose(pfile);
   for (i=0;i<4;i++)
      if ((s.b[i]=rdsg())==ABORT)
        [p_error(15);
         return(1);];
   s.b[4]='$0';
if (cks!=bhex(&s))
    error(BADCHECK);
else
    {wchs(CR);
     stype("Pressure data recovered");} /* print message */
    return(1);} /* complete */
p_error(16);
return(1);
SELF-TEST

Exercise the SAPMD.

#include <supglob.c>  /* locate global data */
#include <process.h>  /* for exit */

selftest() /* command/interrogate SAPMD */
{ static char *cmsgs[] = { "ALL TESTS COMPLETE", /* messages */
                           "EEPROM TEST COMPLETE", "POWER SYSTEM TEST COMPLETE", "A/D CONVERTOR TEST COMPLETE", "PRESSURE TRANSDUCER TEST COMPLETE", "80C31 RAM TEST COMPLETE", "80C31 ROM TEST COMPLETE" };

  static char *emsgs[] = { " - PASSED", /* completion status messages*/
                           " - FAILED" };

  static char *eeperr= [ "EEPROM error at "] ; /* eeprom error message */
  static char *ramerr= [ "80C31 RAM error at "] ; /* ram error message */
  static char *wrote= [ " 'wrote' "] ; /* 'wrote' */
  static char *read= [ " 'read' "] ; /* 'read' */
  static char *c,       /* temporary */
               rch,      /* self test response */
               ech;      /* error character */

  unsigned char c,
               rch,
               ech;

  int i;
  menu(3, i);
  while (i)
  { prompt("SELECT TEST: "); /* prompt for input */
    if ((i=rdln())==LEFT || i==HOME) return(1); /* get input, act. char. */
    if (scan()==NUMBER) /* check for option */
      { c=acc;
        if (scan()==EOL) /* check for number only */
          { switch (c) /* number, process option */
              case 2: /* EEPROM TEST */
              case 6: /* 80C51 RAM TEST */
              case 7: /* 80C51 ROM TEST */
                --c;
                if (!sacmd(SELFTEST,&c,I)) /* issue command */
                  [p_error(BADSAPMD); /* SAPMD broke */
                      continue;] ; /* try again */
              while (1) /* loop until test complete */
                  [switch (rdsg()) /* read response */
                      case TESTCOMP: /* test complete */
                        if ((rch=rdsg())!=ABORT) /* get number */
                          [wchs(CR); /* new line */
                            rch='0'; /* make index */
                            stype(cmsgs[rch]); /* print message */
                            if ((rch=rdsg())!=ABORT) /* get err */
                              [stype(emsgs[rch-'0']); /* error */
                                if (rch==c) break; /* done? */
                                continue;}] ; /* next */
                        p_error(BADSAPMD); /* strange response */
                        break; /* next */
              case EEPERR: /* EEPROM selftest error */
                wchs(CR); /* new line */
                stype(eeperr); /* print message */
                if (!rpbyte()) break; /* print 2 bytes */
                goto ll; /* skip ram poop */
              case RAMERR: /* ram self-test error */
                wchs(CR); /* new line */
                stype(ramerr); /* print message */
                /* EEPROM error entry */
if (!rpbyte()) break; /* print 2 bytes */
stype(wrote); /* print 'wrote' */
if (!rpbyte()) break; /* print 2 bytes */
stype(read); /* print 'read' */
if (!rpbyte()) break; /* print 2 bytes */
continue; /* next */
default: /* else */
p_error(BADSAPMD); /* strange response */
break;};
continue;
case 1:
case 3:
case 4:
case 5:
default:
  error(BADCMD);
  continue;};
if (token==Q)
{scrup(0,0,24,79,0);
i=inp(0x21);
outp(0x21,i|0x10);
exit(0);};
if (token==CMD)
{if (i=excfile()) error(i);} else
  if (token!=EOL) error(BADCMD);}}
status()
    {static char *statxt[]= 
        ["EEPROM-ON","SELF-TEST","GSE","ACQUISITION","COMPLETE","ERROR"];
    int i, 
        j, 
        k, 
        adr, 
        bct;
    unsigned char st;
    st=rdst();
    m4->curx=0;
    m4->cury=0;
    for (i=0;i<6;i++)
        {sat(0x0f);
        if ((st<i)&0x20) sat(0xf8);
        wtype(m4,statxt[i]);
        m4->curx+=4;};
    m4->curx+=6;
    stct=(stct+l)&0x7fff;
    cursor(m4);
    sat(0x0f);
    printf("%5d",stct);
    sat(7);
    if (st&0x40)
        {adr=rdst();
        bct=rdst();
        if (activw==m5)
            {m5->cury=0;
            for (i=bct;i>0;i-=16)
                {wchw(m5,CR);
                hexw(m5,adr);
                wchw(m5,':');
                wchw(' ');
                adr+=16;
                k=0;
                for (j=i>16?16:i;j>0;j--)
                    {wchw(m5,k++==8?'-':' ');
                    hex(m5,rdst());}}}
        else
            for (i=0;i<bct;i++) rdst();}}
/* GLOBAL DECLARATIONS */

#include <stdio.h>
#include <process.h>
#include <stdlib.h>

#if M_I86SM
#pragma message( "Small Model" )
#endif
#if M_I86MM
#pragma message( "Medium Model" )
#endif
#if M_I86CM
#pragma message( "Compact Model" )
#endif
#if M_I86LM
#pragma message( "Large Model" )
#endif
#if M_I86HM
#pragma message( "Huge Model" )
#endif

#define BACKSPACE 8
#define CR 0xd
#define LF 0xa
#define FF 'L'-0x40
#define TAB 9
#define CTRL 3
#define CTRLA 1
#define CTRLR 0x12
#define ESC 0x1b
#define CEOF 0x1a
#define SPL 0
#define DULC 0xc9
#define DURC 0xbb
#define DUMD 0xc7
#define DLMD 0xc8
#define DLLC 0xda
#define DLRNC 0xbc
#define DLNC 0xda
#define DULC 0xc9
#define DURC 0xbb
#define CTRLR 0x12
#define ESC 0x1b
#define CEOF 0x1a
#define SPL 0
#define DULC 0xc9
#define DURC 0xbb
#define DUMD 0xc7
#define DLMD 0xc8
#define DLLC 0xda
#define DLRNC 0xbc
#define DLNC 0xda
#define DULC 0xc9
#define DURC 0xbb
#define CTRLR 0x12
#define ESC 0x1b
#define CEOF 0x1a
#define SPL 0
#define DULC 0xc9
#define DURC 0xbb
#define DUMD 0xc7
#define DLMD 0xc8
#define DLLC 0xda
#define DLRNC 0xbc
#define DLNC 0xda
#define DULC 0xc9
#define DURC 0xbb
#define CTRLR 0x12
#define ESC 0x1b
#define CEOF 0x1a
#define SPL 0
#define DULC 0xc9
#define DURC 0xbb
#define DUMD 0xc7
#define DLMD 0xc8
#define DLLC 0xda
#define DLRNC 0xbc
#define DLNC 0xda
#define DULC 0xc9
#define DURC 0xbb
#define CTRLR 0x12
#define ESC 0x1b
#define CEOF 0x1a
#define SPL 0
#define DULC 0xc9
#define DURC 0xbb
#define DUMD 0xc7
#define DLMD 0xc8
#define DLLC 0xda
#define DLRNC 0xbc
#define DLNC 0xda
#define DULC 0xc9
#define DURC 0xbb
#define CTRLR 0x12
#define ESC 0x1b
#define CEOF 0x1a
#define SPL 0
#define DULC 0xc9
#define DURC 0xbb
#define DUMD 0xc7
#define DLMD 0xc8
#define DLLC 0xda
#define DLRNC 0xbc
#define DLNC 0xda
#define DULC 0xc9
#define DURC 0xbb
#define CTRLR 0x12
#define ESC 0x1b
#define CEOF 0x1a
#define SPL 0
#define DULC 0xc9
#define DURC 0xbb
#define DUMD 0xc7
#define DLMD 0xc8
#define DLLC 0xda
#define DLRNC 0xbc
#define DLNC 0xda
#define DULC 0xc9
#define DURC 0xbb
#define CTRLR 0x12
#define ESC 0x1b
#define CEOF 0x1a
#define SPL 0
#define DULC 0xc9
#define DURC 0xbb
#define DUMD 0xc7
#define DLMD 0xc8
#define DLLC 0xda
#define DLRNC 0xbc
#define DLNC 0xda
#define DULC 0xc9
#define DURC 0xbb
#define CTRLR 0x12
#define ESC 0x1b
#define CEOF 0x1a
#define SPL 0
#define DULC 0xc9
#define DURC 0xbb
#define DUMD 0xc7
#define DLMD 0xc8
#define DLLC 0xda
#define DLRNC 0xbc
#define DLNC 0xda
#define DULC 0xc9
#define DURC 0xbb
#define CTRLR 0x12
#define ESC 0x1b
#define CEOF 0x1a
#define SPL 0
#define DULC 0xc9
#define DURC 0xbb
#define DUMD 0xc7
#define DLMD 0xc8
#define DLLC 0xda
#define DLRNC 0xbc
#define DLNC 0xda
#define DULC 0xc9
#define DURC 0xbb
#define CTRLR 0x12
#define ESC 0x1b
#define CEOF 0x1a
#define SPL 0
#define DULC 0xc9
#define DURC 0xbb
#define DUMD 0xc7
#define DLMD 0xc8
#define DLLC 0xda
#define DLRNC 0xbc
#define DLNC 0xda
#define DULC 0xc9
#define DURC 0xbb
#define CTRLR 0x12
#define ESC 0x1b
#define CEOF 0x1a
#define SPL 0
#define DULC 0xc9
#define DURC 0xbb
#define DUMD 0xc7
#define DLMD 0xc8
#define DLLC 0xda
#define DLRNC 0xbc
#define DLNC 0xda
#define DULC 0xc9
#define DURC 0xbb
#define CTRLR 0x12
#define ESC 0x1b
#define CEOF 0x1a
#define SPL 0
#define DULC 0xc9
#define DURC 0xbb
#define DUMD 0xc7
#define DLMD 0xc8
#define DLLC 0xda
#define DLRNC 0xbc
#define DLNC 0xda
#define DULC 0xc9
#define DURC 0xbb
#define CTRLR 0x12
#define ESC 0x1b
#define CEOF 0x1a
#define SPL 0
#define DULC 0xc9
#define DURC 0xbb
#define DUMD 0xc7
#define DLMD 0xc8
#define DLLC 0xda
#define DLRNC 0xbc
#define DLNC 0xda
#define DULC 0xc9
#define DURC 0xbb
#define CTRLR 0x12
#define ESC 0x1b
#define CEOF 0x1a
#define SPL 0
#define DULC 0xc9
#define DURC 0xbb
#define DUMD 0xc7
#define DLMD 0xc8
#define DLLC 0xda
#define DLRNC 0xbc
#define DLNC 0xda
#define DULC 0xc9
#define DURC 0xb
/* token: control-A */
/* token: control-C */
/* token: control-R */
/* token: control-R */
/* token: comma */
/* token: set-gmt */
/* token: set-met */
/* token: time */
/* token: enter SFR */
/* token: colon */
/* token: dump ram */
/* token: dump code */
/* token: enter ram */
/* token: enter external mem */
/* token: equal sign */
/* token: quit */
/* token: program */
/* token: home */
/* token: end */
/* token: monitor */
/* token: dump SFR */
/* token: '/' */
/* token: load windows */
/* token: input from port */
/* token: output to port */
/* token: enter current type */
/* token: at sign */
/* token: rubber launch */

/* SAPMD command: set-gmt */
/* SAPMD command: set-met */
/* SAPMD command: dump ram */
/* SAPMD command: dump SFR */
/* SAPMD command: dump code */
/* SAPMD command: load ram */
/* SAPMD command: load SFR */
/* SAPMD command: load EEPROM */
/* SAPMD command: self-test */
/* SAPMD command: dump press. */
/* SAPMD command: abort */

/* SAPMD response: ram data */
/* SAPMD response: code data */
/* SAPMD response: test comp. */
/* SAPMD response: SFR data */
/* SAPMD response: press. dta* */
/* SAPMD response: EOD */
/* SAPMD response: EEPROM err*/
/* SAPMD response: ram error */
/* SAPMD response: error */
/* SAPMD response: complete */

/* message: data recovered */
/* error code: SAPMD error */
/* error code: checksum error*/
/* error code: bad command */
/* error code: bad hex file */
/* error code: bad character */
/* error code: file not found*/
/* error code: early EOF */
/* error code: no filename */
/* error code: bad cal. file */
/* error code: no la.cmd */

/* 80C51 address: GMT & MET */
/* window type: blank */
`#define DBT 1`  /* window type: dumped bytes */
`#define DWD 2`  /* window type: dumped words */
`#define DA 3`  /* window type: disassembly */

```c
struct window {
    int scry;
    int curx;
    int cury;
    int lines;
    int daseg;
    int daoff;
    int topoff;
    char wnum;
    char disp;
    char type;
    char ovr;
    char used;
};
```

```c
struct sam {
    int sample;
    int serial;
    float press;
};
```

```c
struct cal {
    int serial;
    int offset;
    float coef;
} sapmd[100];
```

```c
struct window screen
#endif
```

```c
#define MAIN
```

```c
struct window *scline[26]
#endif
```

```c
char ch,
    line[128];
char *iptr
```

```c
#define MAIN
```

```c
int token tacc;
```

```c
/* window context block */
/* cursor position */
/* number of lines in window */
/* address of displayed data */
/* ... offset */
/* addr. of top instr. (DA) */
/* window number */
/* displayed flag */
/* window contents flag */
/* segment override flag */
/* in-use flag */
/* sample number */
/* SAPMD serial # */
/* pre-processed samples */
/* last processed sample */
/* SAPMD serial # */
/* SAPMD transducer adjust. */
/* SAPMD calibration coefs. */
/* underlying screen */
/* fool worthless compiler */
/* compiler fooled */
/* dump windows */
/* fool compiler */
/* compiler fooled */
/* all null */
/* compiler fooled */
/* currently active window */
/* fool worthless compiler */
/* start with screen */
/* character temp */
/* keyboard command line */
/* command input index */
/* fool worthless compiler */
/* initial index */
/* compiler fooled */
/* token id of lexical unit */
/* accumulated number */
int op0,
c, cflag, echo
#ifdef MAIN
    = 1
#endif
int stct
#ifdef MAIN
    = 0
#endif
int cmdfile
#ifdef MAIN
    = 0
#endif
struct window *m1, *m2, *m3, *m4, *m5, *m6;
FILE *cfile;
extern struct window *createw();

/* disassembly opcode byte 0 */
/* characters in byte */
/* byte changed flag */
/* echo kb input flag */
/* fool compiler */
/* default to echo */
/* compiler fooled */
/* status transmission count */
/* fool compiler */
/* clear */
/* compiler fooled */
/* command file flag */
/* fool compiler */
/* ... */
/* */
/* menu windows */
/* command file */
/* func: create dump window */
/*****************************/
The routines in this collection implement the dump windows. They do the right thing at the right time when it is time to draw a box on the screen and keep track of it for the purpose of displaying memory contents of the target computer.

`#include <supglob.c>`
/* locate global data */

`CREATEW`
/* Create a dump window on the screen so it can be dumped all over. */

```c
struct window *createw(id, linect)
/* create a window */
    int id,
    linect;
    if (!box[id].used)
        [if (!box[id].used) /* free slot? */
            box[id].used=l;
            box[id].lines=linect;
            box[id].curx=1;
            box[id].cury=1;
            box[id].disp=0;
            box[id].type=BL;
        return(&box[id]);}
    return(0);]
/* can't create window */
```

`SHOW`
/* Attempt to place the specified window on the screen. Return success */
/* or failure indicator. */

```c
show(id)
/* present window */
    struct window *id;
    [int i,
        j;
    if (id->disp) return(1);
    for (i=0;i<24;i++)
        if (!scline[i])
            [if (25-i<id->lines) break;
                for (j=1;j<id->lines;j++) scline[j]=id; /* mark line in use */
            id->scry=1;
            screen.scry=j;
            screen.lines-=id->lines;
            screen.curx-=id->lines;
            erase(id);
            frame(id);
            id->disp=1;
            return(1);]
    return(0);
/* can't create window */
```

`REMOVW`
/* Delete the specified dump window and erase it from the screen. */
/* Squish any windows that may be under it and update the # lines remaining */
/* on the screen. */

removw(id)
struct window *id;
[int i, j, wbot;]
if (!id) return(l);
if (!id->used) return(0);
if (!id->disp) return(l);
wbot=id->scry+id->lines;
for (i=wbot;scline[i];i+=scline[i]->lines)
  [scline[i]->scry=id->lines;]
  for (j=scline[i]->scry;j<scline[i]->scry+scline[i]->lines;j++) /* for new window */
    scline[j]=scline[i];];
scrup(id->scry,0,i-1,79,id->lines);
for (j=id->lines;j>0;j--) scline[i-j]=0;
screen.scry=id->lines;
screen.cury=id->lines;
screen.lines+=id->lines;
id->disp=0;
if (id==activw) actv(scline[0]);
return(l);]

wchw(id,ch)
struct window *id;
char ch; /* character */
[if (!id->used) return(0);
if (ch==CR) {id->curx=l;
  if (id->cury==id->lines-2)
    uscroll(id);
  else
    id->cury++;
  cursor(id);}]
else
  [cursor(id);
    wch(ch);
    if (id->curx<80) id->curx++;
  return(1);]
[if (ch==CR)                         /* check for new line */
  [screen.curx=0;                 /* cursor to start of line */
    if (screen.cury==screen.lines-1) /* check for bottom */
      scrup(screen.scry,0,screen.scry+screen.lines-1,79,1); /* scroll */
    else                              /* not at bottom line */
      screen.cury++;                   /* next line */
      cursor(&screen);}               /* move cursor */
  else                               /* not cr or lf */
    {cursor(&screen);                /* position cursor */
      wch(ch);                        /* write character */
      if (screen.curx<80)             /* at end of line? */
        return(l);}                   /* return success */
}

/*****************************/
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*****************************/

CURSOR

/* Position the cursor to the screen position of the cursor for the */
/* the specified window. */
/*
/*****************************/
cursor(id)                          /* position cursor */
  struct window *id;               /* window id */
  {movcurs(id->scry+id->cury,id->curx);} /* move cursor */

/*****************************/
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*****************************/

USCROLL

/* Scroll the specified window up. Blank bottom line. */
/*
/*****************************/
uscroll(id)                          /* scroll window 1 line */
  struct window *id;               /* window id */
  {scrup(id->scry+1,1,id->scry+id->lines-2,78,1);} /* scroll */

/*****************************/
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*****************************/

DSCROLL

/* Scroll the specified window down. Blank top line. */
/*
/*****************************/
dscroll(id)                          /* scroll down 1 line */
  struct window *id;               /* window id */
  {scrdn(id->scry+1,1,id->scry+id->lines-2,78,1);} /* scroll */

/*****************************/
/*
/*
/*
/*
/*
/*
/*
/*****************************/

FRAME

/* Draw a box around a dump window. */
/*
/*****************************/
frame(id)                            /* frame a window. */
  struct window *id;                 /* window id */
  [int i;                            /* iteration variable */
    if (id->lines<3) return(0);      /* room for frame? */
    movcurs(id->scry,0);             /* move to upper left corner */
    wch(DULC);                       /* draw upper left corner */
    for (i=1;i<79;i++)               /* draw line */
      wch(DLN);                      /* draw border */
wch(DURC);
for (i=id->scry+1;i<id->scry+id->lines-1;i++)
    [movcurs(i,0);
     wch(DVB);
     movcurs(i,79);
     wch(DVB);};
movcurs(i,0);
wch(DLLC);
for (i=i;i<79;i++)
wch(DLRC);
]
wch(DLN);

/******************************************************************************
/ *                       E R A S E                                      *
/ *                       Blank the specified window.                    *
/ */
/******************************************************************************
erase(id)
    /* blank window */
    struct window *id;
    [scrup(id->scry,0,id->scry+id->lines-1,79,0);] /* clear window */

/******************************************************************************
/ *                       C L E A R                                      *
/ *                       Blank the specified window inside border.      *
/ */
/******************************************************************************
clear(id)
    /* blank window */
    struct window *id;
    [scrup(id->scry+1,1,id->scry+id->lines-2,78,0);] /* clear window */

/******************************************************************************
/ *                       A C T V                                       *
/ *                       Change active window.                         *
/ */
/******************************************************************************
actv(id)
    struct window *id;
    [activw=id;] /* change active window */
    /* new active window */
    /* change activw window */

/******************************************************************************/
CONIO

This collection performs console I/O in "don't help me" mode (direct console I/O). These are C callable functions.

Local symbols:

.8086
PUBLIC _RDC,_WRCH,_TICK

_TEXT SEGMENT BYTE PUBLIC 'CODE'
ASSUME CS:_TEXT


WRCH;
; Write a character to the screen. Character is passed as parameter.

_WRCH PROC NEAR
PUSH BP
MOV BP,SP
MOV mx, 4[BP]
MOV AH,6
INT 21H
POP BP
RET
ENDP

RDCH;
; Check for a keyboard input character. Return 0FFH if no character available, character otherwise.

_RDCH PROC NEAR
; Read from console
MOV AH,6
MOV DL,OFFH
INT 21H
JNZ RDI
MOV AL,0FFH
RDI:
XOR AH,AH
RET
ENDP

TICK;
; Check for an interval timeout. Timeout flag is set by interval timer interrupt.

_TICK PROC NEAR
MOV AX,SEG TIMER
; check for timeout
; point to timer flag
MOV ES, AX
MOV AX, ES: TIMER
RET
_TICK ENDP

; ; TIMER INTERRUPT HANDLER
;
;
;::::::::::::::::::::::::::::::::::::::::;
; timer interrupt handler data
TIMER DW 0
TIMPOOP ENDS
; interrupt flag
; end-of-TIMPOOP

_TEXT ENDS
; end-of-_TEXT

END
; end-of-console I/O routines
F I O

This collection of routines performs SAPMD I/O.

.8086 ; select instruction set
PUBLIC _COMIO, _WRSG, _RDS, SERDATA, _RDST, _POLST, _POLSG, _PURGE

_TEXT SEGMENT BYTE PUBLIC 'CODE' ; code segment
ASSUME CS: _TEXT, DS: SERDATA

_PUBLIC

_COMIO PROC NEAR
  ; COM: I/O
  ; perform C procedure entry protocol
  ; point to parameters
  ; get command
  ; point at RS232 card
  ; get init command
  ; initialize card
  ; delay
  ; point to baud rate
  ; ... 
  ; get high baud rate
  ; set high baud rate count
  ; back up address
  ; delay
  ; get low divisor (18=6400 baud)
  ; set low divisor
  ; new address
  ; get parameter
  ; get character specs
  ; set character poop
  ; save
  ; lock
  ; point to serial port
  ; read away any garbage
  ; delay
  ; read 8259 mask register
  ; clear serial mask
  ; set mask
  ; get 0
  ; point to vectors
  ; initialize serial vector
  ; point to interrupt mask on serial cd.
  ; enable receive interrupts
  ; ... 
  ; unlock
  ; restore
  ; ... 
  ; --> return
  end-of- _COMIO

_COMIO ENDP
; Write a character to the SAPMD.

_WRSG PROC NEAR
        ; write SAPMD
        ; perform C procedure entry protocol
        ; point to parameters
        ; get status port address
WRS1:
        IN    AL,DX
        TEST   AL,40H
        JZ        WRS1
        MOV    DX,3FEH
        IN    AL,DX
        TEST  AL,10H
        JZ            WRS2
        MOV        AL,4[BP]
        MOV    DX,03F8H
        OUT   AL,DX
        POP    BP
        --- > return
_ENDP

; Read a response character from SAPMD.

_RDS PROC NEAR
        ; read SAPMD response byte
        ; save
        ; point to input buffers
        ; ...
        ; get output pointer
        ; check buffer level
        ; compare with input pointer
        ; any data in buffer?
        ; get character
        ; clear upper word
        ; move to next position
        ; mod 256
        ; save
        ; ...
        ; --- > return
_ENDP

; Read a status character from SAPMD.

_RDST PROC NEAR
        ; read SAPMD status byte
        ; save
        ; point to input buffers
        ; ...
_ENDP
RDSL:

MOV BX, DBOUTP ; get output pointer
CMP BX, DBINP ; check buffer level
JE RDSL ; compare with input pointer
MOV AL, DBUF[BX] ; any data in buffer?
MOV AH, 0 ; get character
INC BX ; clear upper word
AND BX, 0FFH ; move to next position
MOV DBOUTP, BX ; mod 256
POP DS ; save
RET ; ---> return

_ENDP

; get output pointer
; check buffer level
; compare with input pointer
; any data in buffer?
; get character
; clear upper word
; move to next position
; mod 256
; save
; ---> return
; end-of-_RDST

POLST:

Check for status character from SAPMD.

POLST PROC NEAR
PUSH DS ; save
MOV AX, SEG SERDATA ; point to input buffers
MOV DS, AX
XOR AX, AX ; ...
MOV BX, DBOUTP ; get output pointer
CMP BX, DBINP ; compare with input pointer
JNE POLS1 ; any data in buffer?
POLS2:
DEC AX ; flag no data
POLS1:
POP DS ; restore and return
... ; ---> return
RET ; ---> return
_ENDP

Poll a response character from SAPMD.

POLSG PROC NEAR
read SAPMD response byte
PUSH DS ; save
MOV AX, SEG SERDATA ; point to input buffers
MOV DS, AX
XOR AX, AX ; flag data
MOV BX, RBOUTP ; get output pointer
CMP BX, RBINP ; compare with input pointer
JNE POLS1 ; any data in buffer?
JMP POLS2 ; no data
_ENDP

Empty SAPMD response buffer.

PURGE PROC NEAR
delete responses
PUSH DS ; save
MOV AX, SEG RBINP ; point to response buffer pointers

_ENDP
; PURGE
; MOV DS, AX
; CLI
; MOV RBINP, 0
; MOV RBOUTP, 0
; STI
; POP DS
; RET
; -->> return
; _PURGE ENDP
; end-of-PURGE

; SERINT
; Serial interrupt handler.
;

SERINT:
; serial interrupt
PUSH AX
PUSH DX
PUSH BX
PUSH SI
PUSH ES
PUSH DS
STI
MOV AX, SEG SERDATA
MOV DS, AX
MOV DX, 3F8H
IN AL, DX
CMP DMFLG, 0
JNE SERIN4
TEST AL, 80H
JNZ SERINI
MOV SI, RBINP
MOV RBUF[SI], AL
INC SI
AND SI, 0FFH
CMP SI, RBOUTP
JNE SERIN2
MOV SI, RBINP
MOV AL, 20H
OUT 20H, AL
POP DS
POP ES
POP SI
POP BX
POP DX
POP AX
IRET
; -->> resume

SERIN2:
MOV RBINP, SI
; restore and return

SERIN3:
MOV AL, 20H
; get EOI code
OUT 20H, AL
POP DS
POP ES
POP SI
POP BX
POP DX
POP AX
; ...

SERIN1:
TEST AL, 40H
JZ SERIN5
; check for status or dump
; status?

SERIN6:
INC DMFLG
; ... nope, dump. Flag it.

SERIN5:
MOV SI, DBINP
MOV DBUF[SI], AL
INC SI
AND SI, 0FFH
CMP SI, DBOUTP
JNE SERIN8
MOV SI, DBINP
; restore pointer

SERIN8:
; restore and return
MOV DBINP,SI ; save input pointer
JMP SERIN3 ; go return

SERIN4: ; dump in progress
CMP DMFLG,2 ; check for byte count time
JL SERIN6 ; go count again
JG SERIN7 ; byte count gone?
MOV BCNT,AL ; plant byte count

SERIN7: ; byte count gone.
DEC BCNT ; count byte
JNL SERIN6 ; more to come?
MOV DMFLG,0 ; last byte
JMP SERIN5 ; save it

_TEXT ENDS

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Local data
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

SERDATA SEGMENT PARA PUBLIC ; serial I/O data
RBINP DW 0 ; response buffer input pointer
RBOUTP DW 0 ; response buffer output pointer
DBINP DW 0 ; dump buffer input pointer
DBOUTP DW 0 ; dump buffer output pointer
DMFLG DW 0 ; dump progress counter
DBUF DB 256 DUP (?) ; dump buffer address
RBUF DB 256 DUP (?) ; response buffer address
BCNT DB 0 ; dump byte count
SERDATA ENDS

END ; end-of-ground I/O routines
These routines support the C routines in WINDOW.C. The perform direct screen control using the ROM BIOS.

Local symbols:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8086</td>
<td>select instruction set</td>
</tr>
<tr>
<td>PUBLIC</td>
<td>_WCH, _SCRUP, _SCRDN, _MOVCURS, _SETVPAGE, _SCH, _SAT</td>
</tr>
<tr>
<td>PUBLIC</td>
<td>_PSHCURS, _SETCURS, _POPCURS</td>
</tr>
</tbody>
</table>

```
_TEXT SEGMENT BYTE PUBLIC 'CODE' ; code segment
ASSUME CS:_TEXT

_WCH PROC NEAR
; WCH
; Write a character to the screen. Character is passed as parameter.
; Character is written at current cursor position using current attributes.

_WRITE PROC NEAR ; Write to console
; perform C procedure entry
; ... point to window data
; ... get character
; get video page number
; get character count
; get BIOS function code
; get current attributes
INT 10H ; write character
MOV AH,3 ; get command
INT 10H ; read cursor position
CMP DL,79 ; check column
JGE WCHI ; end-of-line?
INC DL ; nope, next column
MOV AH,2 ; get command
INT 10H ; set cursor position
WCHI:
POP BP ; don't move cursor
RET ; ---> return
_WCH ENDP ; end-of-_WCH

_MOVCURS PROC NEAR ; move cursor
; save
; point to parameters
; get window data pointer
; ... get video page number
; ...
MOV DL,6[BP]; ...
```
MOV    AH,2 ; get command code
INT    10H ; position cursor
POP    BP ; ...  
RET    ; ---> return
_MOVCURS ENDP ; end-of-_MOVCURS

PSHCURS PROC NEAR ; save cursor
PUSH   BP ; save
MOV    BP,SP ; point to parameters
MOV    AX,SEG WINDPOOP ; get window data pointer
MOV    ES,AX ; ...
MOV    BH,ES:VPAGE ; get video page number
MOV    AH,3 ; get function code
INT    10H ; read cursor poop
MOV    BX,ES:CURPTR ; get stack pointer
MOV    ES:RCOL[BX],DX ; plant position ...
MOV    ES:CTYPE[BX],CX ; ... and attributes
ADD    ES:CURPTR,2 ; bump stack
POP    BP ; ...
RET    ; ---> return
PSHCURS ENDP ; end-of-_PSHCURS

POPCURS PROC NEAR ; restore cursor
PUSH   BP ; save
MOV    BP,SP ; point to parameters
PUSH   SI ; save SI
MOV    AX,SEG WINDPOOP ; get window data pointer
MOV    ES,AX ; ...
MOV    SI,ES:CURPTR ; get stack pointer
TEST   SI,SI ; anything on stack?
JZ     POP1 ; ...
DEC    SI ; pop stack
DEC    SI ; ...
MOV    ES:CURPTR,SI ; ...
MOV    BH,ES:VPAGE ; get video page number
MOV    DX,ES:RCOL[SI] ; get position
MOV    AH,2 ; get function code
INT    10H ; position cursor
MOV    CX,ES:CTYPE[SI] ; get type
MOV    AH,1 ; get function code
INT    10H ; turn on cursor
POP1:   POP    SI ; ... restore
POP    BP ; ...
RET    ; ---> return
POPCURS ENDP ; end-of-_POPCURS

SETCURS
; Set the current cursor attributes.
...

; SETCURS PROC NEAR
PUSH BP
MOV BP, SP
MOV CX, 4[BP]
MOV AH, 1
INT 10H
POP BP
RET

SETCURS ENDP

; set cursor, save, point to parameters, get attributes, get function code, read cursor poop, ---> return

; SCRUP;
; Scroll window up. Bottom line is blanked.

; SCRUP PROC NEAR
MOV AH, 6
SCRUP1:
PUSH BP
MOV BP, SP
push ax
MOV ax, SEG WINDPOOPO
pop ax
MOV BH, ES: ATTRIB
MOV CH, 4[BP]
MOV CL, 6[BP]
MOV DH, 8[BP]
MOV DL, 10[BP]
MOV AL, 12[BP]
INT 10H
POP BP
RET

SCRUP ENDP

; Scroll function code, SCRUP entry, save, point to parameters, point to window poop, ...; get attributes; get upper left row; get upper left column; get lower right row; get lower right column; get # lines; scroll; restore; ---> return; end-of-scroll up

; SCRDN;
; Scroll active window down. Top line is blanked.

; SCRDN PROC NEAR
MOV AH, 7
JMP SCRUP1

SCRDN ENDP

; scroll window down; get function code; go scroll; end-of-SCRDN

; SETVPAGE;
; Set video page number displayed on screen.

; SETVPAGE PROC NEAR
PUSH BP
MOV BP, SP
POP BP
RET

SETVPAGE ENDP

; set video page number, save
MOV BP,SP
MOV AX,SEG VPAGE
MOV ES,AX
MOV AL,4[BP]
MOV ES:VPAGE,AL
MOV AH,5
INT 10H
POP BP
RET

_SETVPAGE ENDP

; point to parameter
; point at page #
; ...
; get page #
; plant page #
; get BIOS function code
; change page
; restore
; ---> return
; end-of__SETVPAGE

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; SCH
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

_SCH PROC NEAR
MOV AX,SEG VPAGE
MOV ES,AX
MOV BH,ES:VPAGE
MOV AH,8
INT 10H
XOR AH,AH
RET

_SCH ENDP

; read character from screen
; point at page #
; ...
; get video page #
; get command code
; read character
; clear attributes
; ---> return
; end-of__SCH

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Set character attributes

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

_SAT PROC NEAR
MOV AX,SEG VPAGE
MOV ES,AX
MOV AX,4[BP]
MOV ES:ATTRIB,AL
POP BP
RET

_SAT ENDP

; set character attributes
; save
; point to stack
; point to attributes
; ...
; get attributes
; plant new attributes
; restore
; ---> return
; end-of__SAT

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; LOCAL DATA

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

WINDEPOOP SEGMENT PARA PUBLIC
ATTRIB DB 07h
VPAGE DB 0
CURPTR DW 0
RCOL DW 0,0,0
CTYPE DW 0,0,0
WINDEPOOP ENDS

__TEXT ENDS

END

; window data
; character attributes
; current video page
; cursor stack pointer
; cursor row and column
; cursor type
; end-of-local data
; end-of__TEXT
; end-of-console I/O routines
Notes on the modified "Shuttle Gauge Access" software, HPSGA:

1. The program is invoked by typing HPSGA<return> at the DOS prompt.

2. Options 3, 4, and 5 have been removed from the self test menu. Option 1, "all tests" has also been removed.

3. Display and print pressure data functions do not include GMT. Only the sample number and the pressure is printed.

4. Display and print pressure data functions do not handle missing or incorrect sample numbers. If this happens, the displayed results may not be reliable.

HPSGA Generation

The program consists of several C and assembly modules. These modules have the same names as the original SGA software.

The C modules were compiled using the Microsoft C version 5.1 compiler. These modules are NOT GUARANTEED to compile without errors on earlier versions of the Microsoft compiler or any other compiler. It may be necessary to make minor modifications to compile with anything other than Microsoft C 5.1.

The assembly modules were compiled using the Microsoft Macro Assembler version 5.1 assembler. These modules are NOT GUARANTEED to assemble without errors on earlier versions of the Microsoft assembler or any other assembler. It may be necessary to make minor modifications to assemble with anything other than Microsoft Macro Assembler 5.1. However, it is expected that earlier versions would assemble without errors.

Two command files are supplied to aid in generating HPSGA:

hpsga           - make file for HPSGA
hpsgalnk.lnk    - link file for HPSGA

The Microsoft MAKE ( ver 4.07 ) utility is used to start the generation by typing the following at the DOS prompt:

make hpsga<return>

This will start the MAKE utility (not supplied). The file "hpsga" contains rules that MAKE uses to decide which files need to be recompiled. After all necessary compilation have been performed, the linker file "hpsgalnk.lnk" is used to link the modules together. This is handled automatically by the MAKE utility. The make file "hpsga" assumes that all of the source files are in a directory named "SAPMD", and that the programming environment follows the conventions suggested by Microsoft. (Specifically, that C include files are in the "\INCLUDE" directory.)

Note that it is not necessary to use MAKE, or the command files
supplied. If these are not used, then each of the files will need to be compiled manually and the programmer needs to keep track of the source files needing to be recompiled or reassembled.

All of the source files were edited using the Microsoft Editor. Any editor that produces ASCII files may be used.

jlk 1/3/89
SAPMD ACCEPTANCE TEST PROCEDURE
An Acceptance Test Procedure
for the
Stand Alone Pressure Monitor

Prepared by: __________________________ Date: 4/7/88

Approved by: __________________________ Date: 4/7/88
Project Manager

Approved by: __________________________ Date: 4/7/88
Quality Assurance Engineer

SOUTHWEST RESEARCH INSTITUTE
P.O. DRAWER 28510
SAN ANTONIO, TEXAS 78284
Change Log Here
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Scope</td>
<td>1</td>
</tr>
<tr>
<td>2. Applicable Documents and Specifications</td>
<td>2</td>
</tr>
<tr>
<td>3. General Test Guidelines</td>
<td>3</td>
</tr>
<tr>
<td>3.1. Test Documentation Practices</td>
<td>3</td>
</tr>
<tr>
<td>3.2. Test Equipment Calibration</td>
<td>3</td>
</tr>
<tr>
<td>3.3. Test Safety</td>
<td>3</td>
</tr>
<tr>
<td>3.4. Test Cleanliness</td>
<td>3</td>
</tr>
<tr>
<td>3.5. Test Rules</td>
<td>3</td>
</tr>
<tr>
<td>4. Initial Electrical Performance Test</td>
<td>5</td>
</tr>
<tr>
<td>4.1. Initial Performance Test Configuration</td>
<td>5</td>
</tr>
<tr>
<td>4.2. Initial Performance Test Measurement Tolerance</td>
<td>5</td>
</tr>
<tr>
<td>4.3. Initial Performance Test Measurement Equipment</td>
<td>5</td>
</tr>
<tr>
<td>5. Thermal Performance Test</td>
<td>9</td>
</tr>
<tr>
<td>5.1. Thermal Performance Test Configuration</td>
<td>9</td>
</tr>
<tr>
<td>5.2. Thermal Performance Test Measurement Tolerances</td>
<td>9</td>
</tr>
<tr>
<td>5.3. Thermal Performance Test Measurement Equipment</td>
<td>9</td>
</tr>
<tr>
<td>5.4. Thermal Performance Tests</td>
<td>11</td>
</tr>
<tr>
<td>6. X Axis Vibration Test</td>
<td>13</td>
</tr>
<tr>
<td>6.1. X Axis Vibration Test Configuration</td>
<td>13</td>
</tr>
<tr>
<td>6.2. X Axis Vibration Test Measurement Tolerances</td>
<td>13</td>
</tr>
<tr>
<td>6.3. X Axis Vibration Test Measurement Equipment</td>
<td>13</td>
</tr>
<tr>
<td>6.4. Vibration Test X Axis</td>
<td>15</td>
</tr>
<tr>
<td>7. Y Axis Vibration Test</td>
<td>17</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1.</td>
<td>Y Axis Test Configuration</td>
<td>17</td>
</tr>
<tr>
<td>7.2.</td>
<td>Y Axis Vibration Test Measurement Tolerances</td>
<td>17</td>
</tr>
<tr>
<td>7.3.</td>
<td>Y Axis Test Measurement Equipment</td>
<td>17</td>
</tr>
<tr>
<td>8.</td>
<td>Z Axis Vibration Test</td>
<td>19</td>
</tr>
<tr>
<td>8.1.</td>
<td>Z Axis Test Configuration</td>
<td>19</td>
</tr>
<tr>
<td>8.2.</td>
<td>Z Axis Measurement Tolerance</td>
<td>19</td>
</tr>
<tr>
<td>8.3.</td>
<td>Z Axis Measurement Equipment</td>
<td>21</td>
</tr>
<tr>
<td>8.4.</td>
<td>Z Axis vibration Tests</td>
<td>21</td>
</tr>
<tr>
<td>9.</td>
<td>Shock Test Procedure</td>
<td>23</td>
</tr>
<tr>
<td>9.1.</td>
<td>Shock Test Configuration</td>
<td>23</td>
</tr>
<tr>
<td>9.2.</td>
<td>Shock Test Measurement Tolerances</td>
<td>23</td>
</tr>
<tr>
<td>9.3.</td>
<td>Shock Test Measurement Equipment</td>
<td>23</td>
</tr>
<tr>
<td>9.4.</td>
<td>Shock Test Specification</td>
<td>25</td>
</tr>
<tr>
<td>9.5.</td>
<td>Shock Tests</td>
<td>25</td>
</tr>
<tr>
<td>10.</td>
<td>Thermal Burn In Procedure</td>
<td>27</td>
</tr>
<tr>
<td>10.1.</td>
<td>Thermal Burn In Test Configuration</td>
<td>27</td>
</tr>
<tr>
<td>10.2.</td>
<td>Thermal Burn In Test Measurement Tolerances</td>
<td>27</td>
</tr>
<tr>
<td>10.3.</td>
<td>Thermal Burn In Measurement Equipment</td>
<td>27</td>
</tr>
<tr>
<td>10.4.</td>
<td>Thermal Burn In Tests</td>
<td>27</td>
</tr>
<tr>
<td>11.</td>
<td>Test Closeout</td>
<td>30</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure Number</th>
<th>Figure Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 4.1</td>
<td>Test Configuration, Initial Electrical</td>
<td>6</td>
</tr>
<tr>
<td>Figure 5.1-1</td>
<td>Test Configuration, Thermal Performance</td>
<td>10</td>
</tr>
<tr>
<td>Figure 6.1-1</td>
<td>Test Configuration, X &amp; Y Axis Vibration</td>
<td>14</td>
</tr>
<tr>
<td>Figure 8.1-1</td>
<td>Test Configuration, Z Axis Vibration</td>
<td>20</td>
</tr>
<tr>
<td>Figure 9.1-1</td>
<td>Test Configuration, Shock Test</td>
<td>24</td>
</tr>
</tbody>
</table>
1. Scope

This document contains the acceptance test procedure for the flight model Stand Alone Pressure Monitor (SAPMD), P.N.15-1062-900, serial numbers 1 to 4. Work on this project was funded by the NASA Johnson Space Center under contract NAS9-17601.
2. Applicable Documents and Specifications

The following documents and specifications are applicable to this procedure to the extent called out in the body of this procedure. In the event of a conflict between the contents of this document and one or more of the documents or specifications listed below this document shall take precedence.

**NASA Documents and Specifications**

- JSC-SP-T-0023B Specifications for Environmental Acceptance Tests
- NAS9-17601 Contract for the Stand Alone Pressure Monitor device (SAPMD)

**DOD Documents and Specifications**

**SwRI Documents and Specifications**

- 1062-CEI-01 Configuration End Item Specification for the Stand Alone Pressure Monitor Device
- DSS-7 Control of Electrostatic Discharge
- XX-AG-103 Instrument Calibration and Instrument Repair Procedure
3. General Test Guidelines

3.1. Test Documentation Practices

Unless otherwise specified all test results are to be recorded directly on this procedure in the spaces provided. When required, Quality Assurance shall affix their stamp to the procedure at points specified in the procedure.

3.2. Test Equipment Calibration

When measurements are made during the execution of this procedure the equipment used to make the measurements shall be calibrated in accordance with SwRI document XX-AG-103. A calibration sticker shall be visible on the test instrument showing the calibration due date.

3.3. Test Safety

The SwRI Project manager shall be responsible for the safe execution of this procedure. He shall take every reasonable precaution to prevent damage to the flight SAPMDs and to reduce the possibility of accident to test personnel. Personnel having reason to handle the SAPMDs will be reminded of the ESD sensitivity of the device and will be directed to review the contents of SwRI document DSS-7 for instructions on ESD damage prevention.

3.4. Test Cleanliness

The SwRI SAPMD project manager shall take reasonable precautions to protest the SAPMDs from becoming seriously contaminated with oils, corrosive materials, radioactive materials, toxic materials or any other material hazardous to test personnel or the flight articles.

3.5. Test Rules

During the execution of this procedure the following test rules shall not be violated.

3.5.1. Test Personnel

The SAPMD shall be operated by the following personnel only;

A) Benny Piepgrass
B) Bill Gibson
C) Gill Harmon (NASA JSC Employee)

3.5.2. Test Facility Failure

If any of the facilities used for the execution of this procedure experience a major failure during the SAPMD testing, the test shall be stopped and shall not be restarted until the SAPMD SwRI Project Manager has determined that the facility has been repaired in such a way as to present no danger of damage to the test articles.
3.5.3. Injury or Illness of Test Personnel

If any of the personnel listed in paragraph 3.5.1 become incapacitated during the execution of this procedure it shall be the responsibility of the SwRI Project manager to determine whether the test can continue. If the SwRI Project Manager is unable to take part in the test Mr. Don Shirley, Manager of Spacecraft Computer Development, shall make the determination.

3.5.4. Conformance

In the event of a test failure the nonconformance shall be dispositioned according to the provisions of the contract (NAS9-17601), DRL T-2049.
4. Initial Electrical Performance Test

This section describes a procedure for verifying the electrical performance of SAPMDs.

4.1. Initial Performance Test Configuration

The initial electrical test configuration shall be as shown in figure 4.1-1.

4.2. Initial Performance Test Measurement Tolerance

For the initial electrical performance test the following tolerances shall be used.

A) Time + - 2 Second  
B) Pressure + - 0.1 PISA

4.3. Initial Performance Test Measurement Equipment

Document below the measurement equipment used for the initial performance test.

<table>
<thead>
<tr>
<th>ITEM</th>
<th>MODEL</th>
<th>MANUFACTURER</th>
<th>S/N</th>
<th>CAL DUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FIGURE 4.1 INITIAL ELECTRICAL TEST CONFIGURATION
4.4. Initial Aliveness Tests

Q.A. Stamp

___ A) Turn on the SAPMD GSE, load the SGA software.

___ B) Attach the GSE cable to SAPMD, SN 01.

___ C) Verify that the tick counter in the upper right corner of the GSE begins to increment after attachment of the SAPMD.

___ D) Following the instructions in section 5.2 of the SGA Software User's Guide, initialize the GMT counter in the SAPMD.

___ E) Following the instructions is section 5.3 of the SGA Software User's User's Guide, read back the contents of the SAPMD's GMT value and verify that the reading is within 1 second of the time initialized.

___ F) Following the instructions in section 6.0 of the SGA User's Guide, execute the "All Self Test" command and verify that the message "Passed" appears on the GSE display.

___ G) Disconnect the GSE cable from SAPMD S/N 1, store the SAPMD in its conductive container.

___ H) Connect SAPMD S/N 2 to the GSE.

___ I) Repeat steps A to F for S/N 2. If the SAPMD does not pass all steps record below the step on which the unit failed.

Failed Step =

___ J) Disconnect the GSE cable from S/N 2 and store the unit safely in its conductive bag.

___ K) Connect SAPMD S/N 3 to the GSE.

___ L) Repeat steps A to F for S/N 3. If the SAPMD does not pass all steps record below the step on which the unit failed.

Failed Step =

___ M) Disconnect the GSE from S/N 3 and store the unit in its conductive bag.

___ N) Connect SAPMD S/N 4 to the GSE.

___ O) Repeat steps A to F for S/N 4. If the SAPMD fails to pass all steps record below the step on which the unit fails.

Failed Step =
P) Disconnect S/N 4 from the GSE and store it safely in its conductive bag.

Q) Turn off the GSE and store all cables in their proper positions in the GSE housing.
5. Thermal Performance Test

The purpose of this test is to verify the ability of the SAPMDs to operate over their specified temperature range.

5.1. Thermal Performance Test Configuration

The configuration for the thermal test shall be as shown in figure 7.1-1.

5.2. Thermal Performance Test Measurement Tolerances

Tolerances for the thermal performance test shall be as follows;

A) Pressure + - 0.1 PISA
B) Time + - 2 Minutes
C) Temperature + - 2 Deg. F

5.3. Thermal Performance Test Measurement Equipment

Document below the measurement equipment used for the temperature performance test.

<table>
<thead>
<tr>
<th>ITEM</th>
<th>MODEL</th>
<th>MANUFACTURER</th>
<th>S/N</th>
<th>CAL DUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.4. Thermal Performance Tests

Q.A. Stamp

___ A) Install SAPMD S/N 1 in the test chamber as shown in figure 7.1-1.

___ B) Attach the SAPMD GSE and verify that the tick counter in the upper right of the GSE display is incrementing.

___ C) Turn on the thermal chamber and set the temperature control for +185 deg.F.

___ D) When the chamber temperature arrives at 185 deg.F, verify that the tick counter is still incrementing.

___ E) Following the instructions in paragraph 6.0 of the SGA Software User's Guide, perform an "All Self Test" on the SAPMD S/N 1.

___ F) Allow the SAPMD to soak at 185 deg.F for approximately 30 minutes.

___ G) Perform a second "All Self Test" on the SAPMD following the instructions in par. 6.0 of the SGA Software User's Guide.

___ H) Set the temperature chamber controller for -30 deg.F.

___ I) When the temperature chamber reaches -30 deg.F, perform a self test on the SAPMD per the instructions in par. 6.0 of the SGA Software User's Guide.

___ J) Allow the SAPMD S/N 1 to soak at -30 deg.F for approximately 30 minutes.

___ K) Perform a second low temperature self test.

___ L) Set the temperature controller for 72 deg.F and allow the chamber to return to room temperature.

___ M) When the chamber temperature has returned to 72 deg.F, disconnect the SAPMD from the GSE and store it safely in its conductive carrier.

___ N) Install SAPMD S/N 2 in the temperature chamber and repeat steps B through L.

___ O) When the chamber temperature reaches 72 deg.F, remove SAPMD S/N 2 from the chamber and store it safely away in its conductive carrier.

___ P) Install SAPMD S/N 3 in the temperature chamber and repeat steps B through L.

___ Q) When the chamber temperature reaches 72 deg. F, remove the SAPMD from the chamber and store it safely in its conductive carrier.

___ R) Install SAPMD S/N 4 in the temperature chamber and repeat steps B through L.
When the chamber temperature reaches 72 deg. F. remove the SAPMD from the chamber and store it safely in its conductive carrier.
6. X Axis Vibration Test

The purpose of this test is to verify the ability of the SAPMDs to withstand the contractually specified X axis vibration environment.

6.1. X Axis Vibration Test Configuration

The vibration test configuration shall be per figure 6.1-1. The required level for the X axis test is 2Gs over a frequency range of 0 to 20000 Hz.

6.2. X Axis Vibration Test Measurement Tolerances

Tolerances for the vibration test shall be as follows;

A) Pressure  + - 0.1 PISA
B) Time  + - 10 Seconds
C) Temperature  + - 2 Deg. F
D) Acceleration  + - .1 Gs

6.3. X Axis Vibration Test Measurement Equipment

Document below the measurement equipment used for the vibration test.

<table>
<thead>
<tr>
<th>ITEM</th>
<th>MODEL</th>
<th>MANUFACTURER</th>
<th>S/N</th>
<th>CAL DUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6.4. **Vibration Test X Axis**

**Q.A. Stamp**

___ A) With SAPMD S/N 1 attached to the vibration test stand turn on the GSE, load and start the SGA software.

___ B) Connect the GSE cable to the SAPMD.

___ C) Verify that the tick counter is incrementing on the GSE display.

___ D) Following the instructions is para.5.2 of the SGA Software User's Guide set the SAPMD GMT time with local time.

___ E) Following the instructions is par.5.3 of the SGA User's Guide read back GMT from the SAPMD and verify that time is being kept properly. The SAPMD time should be within 1 second of local time.

___ F) Following the instructions in par. 3.4 of the SGA User's Guide run the "Set001.cmd" batch file.

___ G) When the setup command has run to completion disconnect the GSE cable from the SAPMD.

___ H) Start the X-Axis vibration and continue the vibration for 1 minute.

___ I) After the 1 minute of vibration reconnect the GSE cable to the SAPMD.

___ J) Verify that the tick counter is still running.

___ K) If the SAPMD acquired pressure data during the vibration test, follow the instructions in par.4.3 of the SGA Software User's Guide, to acquire the pressure data from the SAPMD and store it on a floppy disc under the title "001XAXIS.DTA".

___ L) Following the instructions in par.5.2 of the SGA Software User's Guide, inspect the SAPMD GMT clock value and verify that it is within 5 second of local time.

___ M) Disconnect SAPMD S/N 1 from the test stand and install SAPMD S/N 2.

___ N) Repeat steps B through L using file name "002XAXIS.DTA" for data and "SET002.cmd" for setup of S/N 2.

___ O) Remove S/N 2 from the test stand and install S/N 3.

___ P) Repeat steps B through L using file name "003XAXIS.DTA" for data and "SET003.CMD" for setup of S/N 3.

___ Q) Remove S/N 3 from the test stand and install S/N 4.
R) Repeat steps B through L using file name "004XAXIS.DTA" for data and "SET004.CMD" for setup of S/N 4.

S) Remove S/N 4 from the test stand.

T) Reconfigure the test stand for y axis testing.
7. Y Axis Vibration Test

The purpose of this test is to verify the ability of the SAPMD to withstand the Y axis vibration loads specified by the contract.

7.1. Y Axis Test Configuration

The Y axis test configuration shall be the same as the x axis configuration.

7.2. Y Axis Vibration Test Measurement Tolerances

The Y axis measurement tolerances are the same as those for the x axis. The Y axis vibration level is 2Gs from 0 to 2000Hz.

7.3. Y Axis Test Measurement Equipment

The Y axis measurement equipment list is the same as that for the x axis.

7.4. Y Axis Vibration Tests

Q.A. Stamp

_____ A) Attach SAPMD S/N 1 to the vibration test fixture.

_____ B) Attach the GSE connector to the SAPMD and verify that the tick counter is still incrementing.

_____ C) Following the instructions is par.5.3 of the SGA User's Guide read back GMT from the SAPMD and verify that time is being kept properly. The SAPMD time should be within 20 seconds of local time.

_____ D) Following the instructions in par.3.4 of the SGA Software User's Guide, execute the "001Set.cmd" batch file.

_____ E) After completion of the "001Set.cmd" batch file disconnect the SAPMD from the GSE cable.

_____ F) Start the Y axis random vibration test and continue vibrating the SAPMD for 1 minute.

_____ G) At the end of vibration reconnect the SAPMD and verify that the tick counter is still incrementing.

_____ H) Following the instructions in par.5.2 of the SGA Software User's Guide, inspect the SAPMD GMT clock value and verify that it is within 30 seconds of local time.

_____ I) If the SAPMD acquired data during the vibration test, follow the instructions in par.-4.3 of the SGA Software User's Guide, to transfer the pressure data from the SAPMD to the GSE floppy disc with the file labeled "001YAXIS.DTA".
J) Remove SAPMD S/N 1 from the test stand and install S/N 2.

K) Connect the GSE cable to S/N 2 and verify that the tick counter is incrementing.

L) Repeat steps C through H using file "Set002.cmd" for setup and "002YAXIS.DTA" for the pressure data file for S/N 2.

M) Remove S/N 2 from the test stand, install S/N 3.

N) Connect the GSE cable to S/N 3 and repeat steps C through H for S/N 3 using file "Set003.cmd" for setup and "003YAXIS.DTA" for pressure data.

O) Remove S/N 2 from the test stand, install S/N 4.

P) Connect the GSE cable to S/N 4 and repeat steps C through H for S/N 4 using file name "Set004.cmd" for setup and "004YAXIS.DTA" for pressure data.

Q) Remove S/N 4 from the test stand.

R) Reconfigure the test stand for z axis testing.
8. Z Axis Vibration Test

The purpose of this test is to verify that the SAPMD will withstand the z axis vibration loads as specified in the contract.

8.1. Z Axis Test Configuration

The Z axis test configuration shall be as shown in figure 10.1-1.

8.2. Z Axis Measurement Tolerance

The Z axis measurement tolerances shall be the same as those used for the x and y axes. The vibration level for the Z axis shall be 2Gs from 0 to 2000Hz.
8.3. Z Axis Measurement Equipment

List below the test equipment used for the z axis vibration test.

<table>
<thead>
<tr>
<th>ITEM</th>
<th>MODEL</th>
<th>MANUFACTURER</th>
<th>S/N</th>
<th>CAL DUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8.4. Z Axis vibration Tests

Q.A. Stamp

___ A) Attach SAPMD S/N 1 to the test fixture.

___ B) Connect the GSE cable to the SAPMD.

___ C) Verify that the tick counter is incrementing.

___ D) Following the instructions in par. 5.2 of the SGA Software User's Guide, verify that the SAPMD's GMT time is within 40 seconds of local time.

___ E) Following the instructions in par. 3.4 of the SGA Software User's Guide, execute the "Set001.cmd" batch file.

___ F) Disconnect the GSE cable and start the z axis vibration.

___ G) After 1 minute discontinue the vibration and reconnect the GSE cable.

___ H) Verify that the tick counter is still running.

___ I) Following the instructions in par. 5.2 of the SGA Software User's Guide, read the GMT time and verify that it is within 50 seconds of local time.

___ J) Note whether the SAPMD acquired pressure data. If data was acquired transfer the acquired data to the GSE and store it in a floppy disc file labeled "001ZAXIS.DTA".

___ K) Disconnect the GSE cable and store S/N 1 in its conductive carrier.

___ L) Install SAPMD S/N 2 on the vibration test stand and connect the GSE cable.
M) Repeat steps C through J on S/N 2 using file "SET002.CMD" for setup and "002ZAXIS.DTA" for pressure data.

N) Disconnect the GSE cable from S/N 2, remove it from the test stand and store it in its conductive carrier.

O) Install SAPMD S/N 3 on the test stand and connect the GSE cable.

P) Repeat steps C through J on S/N 3 using file "Set003.CMD" for setup and "003ZAXIS.DTA" for pressure data.

Q) Disconnect the GSE cable and store S/N 3 in its conductive carrier.

R) Install SAPMD S/N 4 on the test stand and connect the GSE cable.

S) Repeat steps C through J on S/N 4 using file "Set004.cmd" for setup and "004ZAXIS.DTA" for pressure data.

T) Disconnect the GSE cable from S/N 4, remove it from the test stand and store it in its conductive carrier.

U) Turn off the GSE.
9. Shock Test Procedure

The purpose of this test is to verify that the SAPMD can withstand the shock specified in the contract and continue to operate correctly.

9.1. Shock Test Configuration

The shock test configuration shall be as shown in figure 9.1-1.

9.2. Shock Test Measurement Tolerances

Tolerances for the shock test shall be as follows;

A) Time + 3 milliseconds
B) Acceleration +5Gs

9.3. Shock Test Measurement Equipment

Document below the measurement equipment used for the shock test.

<table>
<thead>
<tr>
<th>ITEM</th>
<th>MODEL</th>
<th>MANUFACTURER</th>
<th>S/N</th>
<th>CAL DUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FIGURE 9.1-1  SHOCK TEST CONFIGURATION
Figure 9.1-1  Test Configuration, Shock Test
9.4. Shock Test Specification

The SAPMD is to be shocked to 78Gs for 11 milliseconds with a half sine waveform.

It is assumed that the shock test is executed within 2 days of completion of the vibration tests described in section 10. If this is not the case all 4 of the SAPMDs must have their GMT counters reinitialized per the instructions in par. 5.2 of the SGA Software User's Guide.

9.5. Shock Tests

Q.A. Stamp

____ A) Attach SAPMD S/N 1 to the shock test fixture.

____ B) Shock S/N 1 to the specified level.

____ C) Turn on the GSE, load and start the SGA software.

____ D) Attach the GSE to the SAPMD and verify that the tick counter is incrementing.

____ E) Following the instructions in par. 5.2 of the SGA Software User's Guide, read the SAPMD GMT and verify that it is within 15 seconds of local time.

____ F) Disconnect the GSE from S/N 1.

____ G) Remove S/N 1 from the test stand and attach S/N 2.

____ H) Shock S/N 2 to the specified level.

____ I) Attach the GSE to S/N 2 and verify the tick counter is running.

____ J) Repeat step E.

____ K) Disconnect the GSE from S/N 2.

____ L) Remove S/N 2 and attach S/N 3.

____ M) Shock S/N 3 to the specified levels.

____ N) Repeat step E.

____ O) Disconnect the GSE from S/N 3.

____ P) Remove S/N 3 from the test stand and attach S/N 4.

____ Q) Shock S/N 4 to the specified levels.
R) Repeat step E.
S) Disconnect the GSE from S/N 4.
T) Remove S/N 4 from the test stand.
U) Turn off the GSE and pack all GSE cables and documentation in the GSE carrying case.
V) Attach the hardcopies of the shock data to the end of this procedure.
10. Thermal Burn In Procedure

The purpose of this test is to subject the SAPMDs to an extended period of operation at elevated temperature. Latent manufacturing problems, if present, should be detected with this test.

10.1. Thermal Burn In Test Configuration

For the thermal burn in test the SAPMDs shall be placed in a thermal chamber and their temperature monitored. The GSE shall not be connected to the SAPMDs until the end of the test at which time the units will be taken out of the temperature chamber.

10.2. Thermal Burn In Test Measurement Tolerances

Tolerances for the vibration test shall be as follows;

B) Time + - 30 Minutes

C) Temperature + - 2 Deg. F

10.3. Thermal Burn In Measurement Equipment

Document below the measurement equipment used for the burn in test.

<table>
<thead>
<tr>
<th>ITEM</th>
<th>MODEL</th>
<th>MANUFACTURER</th>
<th>S/N</th>
<th>CAL DUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10.4. Thermal Burn In Tests

Q.A. Stamp

___ A) Turn on the SAPMD GSE and load the SGA software.

___ B) Remove SAPMD S/N 1 from its protective carrier and place it on a safe working surface for attachment to the GSE.

___ C) Attach the GSE to SAPMD S/N 1.
Following the instructions in par. 5.2 of the SGA Software User's Guide, set the GMT value in the SAPMD to local time.

Following the instructions in par. 5.3 of the SGA Software User's Guide, read back the GMT and verify correct time to within 1 second.

Disconnect the GSE cable from the SAPMD and place S/N 1 in the temperature chamber.

Repeat steps B through F on SAPMD S/N 2.

Repeat steps B through F on SAPMD S/N 3.

Repeat steps B through F on SAPMD S/N 4.

Turn on the temperature chamber and set the temperature controls for 110 deg. F.

Leave the SAPMDs in the temperature chamber for 48 hours.

Turn off the temperature chamber and allow the interior temperature to return to approximately 72 deg. F.

Remove SAPMD S/N 1 from the chamber and place it on a safe, ESD controlled, working surface for attachment to the GSE.

With the GSE already on attach S/N 1 to the GSE.

Following the instructions in par. 5.3 of the SGA Software User's Guide, read the GMT value from the SAPMD and verify that the time seen is within 5 minutes of local time.

Disconnect the GSE from S/N 1 and store the SAPMD in its conductive carrier.

Repeat steps M through P for SAPMD S/N 2.

Repeat steps M through P for SAPMD S/N 3.

Repeat steps M through P for SAPMD S/N 4.

Turn off the GSE and store all cables and supplies in their appropriate places in the GSE enclosure.
11. Test Closeout

Before completing this procedure verify that copies of all vibration and shock data plots are attached to the back of this procedure. Also verify that all attached plots are labeled with the appropriate information (i.e. date, time, axis, SAPMD S/N).

___ A) Remove SAPMD SN 1 from its conductive carrier and place it on a clean, ESD controlled working surface.

___ B) Carefully remove the battery pack and store it in a clean conductive carrier.

___ C) Replace the SAPMD in its conductive carrier.

___ D) Repeat step A with SAPMD S/N 2.

___ E) Repeat step A with SAPMD S/N 3.

___ F) Repeat step A with SAPMD S/N 4.