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

Part 2

SECTION 2

```
/*****************************************************************************/
/*                                                                           */
/*                              D U M P                                      */
/*                                                                           */
/*      Dump routines.                                                       */
/*                                                                           */
/**************************************************                          */
                                                /*                           */
#include <supglob.c>                            /* get globals               */
                                                /*                           */
/**************************************************                          */
/*                                                                           */
/*                              D B Y T E S                                  */
/*                                                                           */
/*      Dump a block of SC-1 memory in the specified window in byte format.  */
/* The address is in dseg|doff.                                              */
/*                                                                           */
/**************************************************                          */
                                                /*                           */
dbytes(id)                                      /* dump bytes in window      */
    struct window *id;                          /* window id                 */
    {int i,                                     /* iteration variable        */
         off;                                   /* dump offset               */
     off=id->topoff;                            /* dump offset               */
     id->cury=1;                                /* position cursor of window */
     for (i=3;i<=id->lines;i++)                 /* display data loop         */
        {dumpln(id,id->daseg,off);              /* show a line               */
         off+=16;                               /* next line of data         */
         id->cury++;};                          /* next line                 */
     id->cury=1;}                               /* start of window           */
                                                /*                           */
/**************************************************                          */
/*                                                                           */
/*                              U B Y T E S                                  */
/*                                                                           */
/*      Fill the specified window with unassembled instructions.             */
/*                                                                           */
/**************************************************                          */
                                                /*                           */
ubytes(id)                                      /* unassemble                */
    struct window *id;                          /* window id                 */
    {int i;                                     /* iteration variable        */
     clear(id);                                 /* blank inside window       */
     id->cury=1;                                /* position cursor of window */
     id->daoff=id->topoff;                      /* establish last dis. addr. */
     for (i=3;i<=id->lines;i++)                 /* display data loop         */
        {diss(id,id->daseg,id->daoff);          /* show a line               */
         id->daoff+=ip;                         /* next line of data         */
         id->cury+=1;};                         /* next line                 */
     id->type=DA;}                              /* flag window non-empty     */
                                                /*                           */
/**************************************************                          */
/*                                                                           */
/*                              P A D D R                                    */
/*                                                                           */
/*      Print the passed segment and offset.                                 */
/*                                                                           */
/**************************************************                          */
                                                /*                           */
paddr(id,seg,off)                               /* print address             */
    struct window *id;                          /* window id                 */
    int seg,                                    /* segment address           */
        off;                                    /* offset                    */
    {hexw(id,seg);                              /* write segment high        */
     wchw(id,':');                              /* write colon               */
     hexw(id,off);}                             /* write offset high         */
```

```
                                                 /*                           */
/*************************************************                            */
/*                                                                            */
/*                              H E X W                                       */
/*                                                                            */
/*      Print the passed word in hex.                                         */
/*                                                                            */
/*************************************************                            */
                                                 /*                           */
hexw(id,x)                                       /* display hex word          */
    struct window *id;                           /* window id                 */
    int x;                                       /* data                      */
    {hex(id,x>>8);                               /* display high ...          */
     hex(id,x);}                                 /* ... and low               */
                                                 /*                           */
/*************************************************                            */
/*                                                                            */
/*                              H E X C                                       */
/*                                                                            */
/*      Convert the passed nibble to hex ascii.                               */
/*                                                                            */
/*************************************************                            */
                                                 /*                           */
char hexc(x)                                     /* convert to hex ascii      */
    int x;                                       /* nibble                    */
    {x&=0xf;                                     /* get nibble                */
     return((x<=9)?x+'0':x-10+'A');}             /* convert to ascii          */
                                                 /*                           */
/*************************************************                            */
/*                                                                            */
/*                              H E X                                         */
/*                                                                            */
/*      Print the specified byte at the current cursor position on the        */
/* specified window.                                                          */
/*                                                                            */
/*************************************************                            */
                                                 /*                           */
hex(id,x)                                        /* print byte in hex         */
    struct window *id;                           /* window id                 */
    int x;                                       /* data                      */
    {wchw(id,hexc(x>>4));                        /* print high nibble         */
     wchw(id,hexc(x));}                          /* print low nibble          */
                                                 /*                           */
/*************************************************                            */
/*                                                                            */
/*                              D U M P U P                                   */
/*                                                                            */
/*      Scroll the specified dump window in response to an up arrow.          */
/*                                                                            */
/*************************************************                            */
                                                 /*                           */
dumpup(id)                                       /* scroll up                 */
    struct window *id;                           /* window id                 */
    {dscroll(id);                                /* blank top line            */
     id->topoff-=16;                             /* adjust address in window  */
     id->cury=1;                                 /* adjust cursor             */
     dumpln(id,id->daseg,id->topoff);}           /* dump a line               */
                                                 /*                           */
/*************************************************                            */
/*                                                                            */
/*                              D U M P D N                                   */
/*                                                                            */
/*      Scroll the specified dump window in response to a down arrow.         */
/*                                                                            */
/*************************************************                            */
                                                 /*                           */
```

```
dumpdh(id)                                      /* scroll up                */
    struct window *id;                          /* window id                */
    {uscroll(id);                               /* blank top line           */
     id->topoff+=16;                            /* adjust address in window */
     id->cury=id->lines-2;                      /* adjust cursor            */
     dumpln(id,id->daseg,id->topoff+(id->lines-3)*16);} /* dump a line      */
                                                /*                          */
/*************************************************                          */
/*                                                                          */
/*                              U N D N                                     */
/*                                                                          */
/*      Scroll the specified dump window in response to a down arrow.       */
/*                                                                          */
/*************************************************                          */
                                                /*                          */
undn(id)                                        /* scroll up                */
    struct window *id;                          /* window id                */
    {int i,                                     /* top address accumulator  */
         j,                                     /* iteration variable       */
         k;                                     /* screen character         */
     uscroll(id);                               /* blank top line           */
     id->cury=id->lines-2;                      /* adjust cursor            */
     diss(id,id->daseg,id->daoff);              /* disassemble a line       */
     id->daoff+=ip;                             /* next line of data        */
     i=0;                                       /* clear accumulator        */
     for (j=0;j<4;j++)                          /* decode address loop      */
        {movcurs(id->scry+1,6+j);               /* position to offset       */
         k=sch();                               /* get screen character     */
         k=(k<='9')?k-'0':k-'A'+10;             /* convert to binary        */
         i=(i<<4)+k;};                          /* accumulate address       */
     id->topoff=i;}                             /* set new top address      */
                                                /*                          */
/*************************************************                          */
/*                                                                          */
/*                              D U M P L N                                 */
/*                                                                          */
/*      Dump a line of SC-1 memory.                                         */
/*                                                                          */
/*************************************************                          */
                                                /*                          */
dumpln(id,dseg,off)                             /* dump data                */
    struct window *id;                          /* window id                */
    int dseg,                                   /* segment address          */
        off;                                    /* offset                   */
    {unsigned char dmp[16];                     /* dumped data              */
     int j,                                     /* iteration variable       */
         i;                                     /* iteration variable       */
     id->curx=1;                                /* move to start of line    */
     if (!scdump(dseg,off,17)) return(0);       /* send dump command        */
     for (j=0;j<=15;j++) dmp[j]=rdsc1();        /* read data                */
     paddr(id,dseg,off);                        /* print memory address     */
     wchw(id,' ');                              /* space ...                */
     wchw(id,' ');                              /* ...                      */
     if (id->type==DBT)                         /* dump bytes or words?     */
        {for (j=0;j<=15;j++)                    /* dump loop                */
            {hex(id,dmp[j]);                    /* write byte in hex        */
             if (dmp[j]<' ' || dmp[j]>=0x7f) dmp[j]='.'; /* for printing    */
             if (j==7)                          /* middle?                  */
                wchw(id,'-');                   /* dash in middle           */
             else                               /* otherwise, space         */
                wchw(id,' ');}}                 /* ...                      */
     else                                       /* words ...                */
        for (j=0;j<=15;j+=2)                    /* dump loop 2              */
            {hexw(id,(dmp[j+1]<<8)|dmp[j]);     /* dump a word              */
             for (i=j;i<=j+1;i++)               /* convert to ascii loop    */
                {if (dmp[i]<' ' || dmp[i]>=0x7f) dmp[i]='.'; /* for printing */
```

```
                 wchw(id,' ');}};               /* space between words       */
     wchw(id,' ');                              /* another space             */
     wchw(id,'*');                              /* print star                */
     for (j=0;j<=15;j++)                        /* encode and print data     */
        wchw(id,dmp[j]);                        /* print it                  */
     wchw(id,'*');                              /* trailing star             */
     if (rdscl()!=PROMPT) error(BADSC1);}       /* SC-1 in synch?            */
                                                /*                           */
/*************************************************                           */
/*                                                                           */
/*                              D M P R E G S                                */
/*                                                                           */
/*      Dump registers to screen window.                                     */
/*                                                                           */
/*************************************************                           */
                                                /*                           */
dmpregs()                                       /* dump registers            */
    {int i;                                     /* iteration variable        */
     char r;                                    /* register half temp        */
     wrscl(DREGS);                              /* send dump command         */
     if (!gregs())                              /* check response code       */
        error(BADSC1);                          /* send error                */
     else                                       /* here come the registers   */
        if (!dregs()) error(BADSC1);}           /* check completion          */
                                                /*                           */
/*************************************************                           */
/*                                                                           */
/*                              T R A C E                                    */
/*                                                                           */
/*      Trace execution (single-step).                                       */
/*                                                                           */
/*************************************************                           */
                                                /*                           */
trace(token)                                    /* trace execution           */
    int token;                                  /* command code              */
    {wrscl(STEP);                               /* send step command         */
     gregs();                                   /* get registers             */
     dregs();                                   /* display registers         */
     wchs(CR);                                  /* new line                  */
     diss(&screen,sclregs[CS],sclregs[IP]);     /* display next instruction  */
     if (token==TU)                             /* trace & unassemble?       */
        {if (!mkwnd()) return(0);               /* try to make one           */
         activw->daseg=sclregs[CS];             /* ... yep, set dump address */
         activw->topoff=sclregs[IP];            /* ...                       */
         ubytes(activw);}};                     /* disassemble               */
                                                /*                           */
/*************************************************                           */
/*                                                                           */
/*                              G O                                          */
/*                                                                           */
/*      Start execution until optional breakpoint.                           */
/*                                                                           */
/*************************************************                           */
                                                /*                           */
go(token)                                       /* begin execution           */
    int token;                                  /* command code              */
    {if (bf) brkpt(gopoop[2],gopoop[3]);        /* set breakpoint            */
     if (af)                                    /* start address present?    */
        {lreg(IP,gopoop[1]);                    /* load IP                   */
         lreg(CS,gopoop[0]);};                  /* load CS                   */
     wrscl(GO);                                 /* send step command         */
     gregs();                                   /* get registers             */
     if (bf) loadscl(gopoop[2],gopoop[3],1,&bpinst); /* restore instruction  */
     dregs();                                   /* display registers         */
     wchs(CR);                                  /* new line                  */
     diss(&screen,sclregs[CS],sclregs[IP]);     /* display next instruction  */
     if (token==GU)                              /* trace & unassemble?        */
        {if (!mkwnd()) return(0);                /* try to make one            */
         activw->daseg=sc1regs[CS];              /* ... yep, set dump address  */
         activw->topoff=sc1regs[IP];             /* ...                        */
         ubytes(activw);}};                      /* disassemble                */
                                                 /*                            */
/*************************************************                            */
/*                                                                            */
/*                              B R K P T                                     */
/*                                                                            */
/*      Set breakpoint.                                                       */
/*                                                                            */
/*************************************************                            */
                                                 /*                            */
brkpt(seg,off)                                   /* set a breakpoint           */
    int seg,                                     /* address segment            */
        off;                                     /* ... and offset.            */
    {static unsigned char trap=0xcc;             /* breakpoint instruction     */
     if (!scdump(seg,off,2)) return(0);          /* send dump command          */
     bpinst=rdsc1();                             /* get replaced instruction   */
     if (rdsc1()!=PROMPT)                        /* check for good dump        */
        {error(BADSC1);                          /* send error                 */
         return(0);};                            /* return                     */
     bpact=1;                                    /* flag active breakpoint     */
     loadsc1(seg,off,1,&trap);}                  /* set breakpoint             */
                                                 /*                            */
/*************************************************                            */
/*                                                                            */
/*                              L O A D S C 1                                 */
/*                                                                            */
/*      Load SC-1 memory.                                                     */
/*                                                                            */
/*************************************************                            */
                                                 /*                            */
loadsc1(seg,off,len,poop)                        /* load target memory         */
    int seg,                                     /* address segment ...        */
        off,                                     /* ... and offset             */
        len;                                     /* byte count                 */
    char *poop;                                  /* data address               */
    {wrsc1(LOAD);                                /* send load memory command   */
     saddr(seg,off,len);                         /* send address               */
     for (;len>0;len--) wrsc1(*poop++);          /* load data                  */
     if (rdsc1()!=PROMPT) error(BADSC1);}        /* good load?                 */
                                                 /*                            */
/*************************************************                            */
/*                                                                            */
/*                              G R E G S                                     */
/*                                                                            */
/*      Read registers from SC-1.                                             */
/*                                                                            */
/*************************************************                            */
                                                 /*                            */
gregs()                                          /* get registers              */
    {int i,                                      /* iteration variable         */
         r;                                      /* register temp              */
     if (rdsc1()!=DUMPREG) return(0);            /* return error               */
     for (i=0;i<14;i++)                          /* print regs.                */
        {r=rdsc1();                              /* get low reg half           */
         sc1regs[i]=r|(rdsc1()<<8);};            /* get register high half     */
     if (rdsc1()!=PROMPT) return(0);             /* get prompt                 */
     return(1);}                                 /* return success             */
                                                 /*                            */
/*************************************************                            */
/*                                                                            */
/*                              D R E G S                                     */
/*                                                                            */
/*      Display registers from SC-1.                                         */
/*                                                                           */
/*************************************************                           */
                                                /*                           */

dregs()                                         /* display registers         */
    {int i;                                     /* iteration variable        */
     static char rtext[]={"AXBXCXDXSIDIBPSPCSDSESSSIPFL"}; /* 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('=');                             /* ...                       */
         hexw(&screen,sclregs[i>>1]);           /* print register            */
         wchs(' ');                             /* space between registers   */
         wchs(' ');                             /* space between registers   */
         if (i==12) wchs(CR);}                  /* new line in middle        */
     return(1);}                                /* good register dump        */
                                                /*                           */
/*************************************************                           */
/*                                                                           */
/*                              S C D U M P                                  */
/*                                                                           */
/*      Send command to dump a block of SC-1 memory.                         */
/*                                                                           */
/*************************************************                           */
                                                /*                           */
scdump(seg,off,len)                             /* dump SC-1 memory          */
    int seg,                                    /* segment address           */
        off,                                    /* offset                    */
        len;                                    /* byte count+1              */
    {wrscl(DMEM);                               /* send dump command to SC1  */
     saddr(seg,off,len);                        /* send address and length   */
     if (rdscl()!=DUMPMEM)                      /* check response            */
        {error(BADSC1);                         /* bad response              */
         return(0);}                            /* return error              */
     return(1);}                                /* good SC-1 response        */
                                                /*                           */
/*************************************************                           */
/*                                                                           */
/*                              S A D D R                                    */
/*                                                                           */
/*      Send memory address to SC-1.                                         */
/*                                                                           */
/*************************************************                           */
                                                /*                           */
saddr(seg,off,len)                              /* send address to SC-1      */
    int seg,                                    /* address segment ...       */
        off,                                    /* ... and offset            */
        len;                                    /* byte count                */
    {wrscl(seg);                                /* send low seg addr.        */
     wrscl(seg>>8);                             /* send high seg addr.       */
     wrscl(off);                                /* send low offset           */
     wrscl(off>>8);                             /* send high offset          */
     wrscl(len);}                               /* send byte count+1         */
                                                /*                           */
/*************************************************                           */
/*                                                                           */
/*                              L R E G                                      */
/*                                                                           */
/*      Load an SC-1 register.                                               */
/*                                                                           */
/*************************************************                           */
                                                /*                           */
lreg(reg,x)                                     /* load register             */
    int reg,                                    /* register #                */
        x;                                      /* data                      */
    {wrscl(LREGS);                              /* send load command        */
     wrscl(reg<<1);                             /* send register #          */
     wrscl(x);                                  /* send low half            */
     wrscl(x>>8);                               /* send high half           */
     if (rdscl()!=PROMPT) error(BADSC1);}       /* check for prompt         */
                                                /*                          */
/*************************************************                           */
/*                                                                           */
/*                              W N D I R                                    */
/*                                                                           */
/*      Display directory of dump windows.                                   */
/*                                                                           */
/*************************************************                           */
                                                /*                          */
wndir()                                         /* window directory         */
    {static char *title=                        /* directory title          */
  {"Window    Type    Addr    Lines               Window    Type    Addr    Lines"};
     int i;                                     /* iteration variable       */
     wchs(CR);                                  /* new line                 */
     stype(title);                              /* print title              */
     for (i=0;i<4;i++)                          /* scan windows             */
        {wchs(CR);                              /* new line                 */
         wndex(i);                              /* display window discription*/
         stype("             ");                /* space to next column     */
         wndex(i+4);}}                          /* next column              */
                                                /*                          */
/*************************************************                           */
/*                                                                           */
/*                              W N D E X                                    */
/*                                                                           */
/*      Display dump window poop.                                            */
/*                                                                           */
/*************************************************                           */
                                                /*                          */
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]);              /* print type               */
     else                                       /* window unused            */
        stype("Free");                          /*                          */
     stype("  ");                               /* space                    */
     if (box[i].used==0)                        /* window filled?           */
        stype("             ");                 /* nope blank it            */
     else                                       /* window filled            */
        {if (box[i].type==BL)                   /* window blank?            */
            stype("        ");                  /* print string             */
         else                                   /* text there               */
            {hexw(&screen,box[i].daseg);        /* type address             */
             wchs(':');                         /*                          */
             hexw(&screen,box[i].topoff);};     /*                          */
         stype("   ");                          /* space                    */
         hex(&screen,box[i].lines-2);}}         /* line count               */
                                                /*                          */
                                                /****************************/
```

```
/******************************************************************************/
/*                                                                            */
/*                              E R R O R                                     */
/*                                                                            */
/*      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",
     "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);}
```

```
/******************************************************************************/
/*                                                                            */
/*                                 M E N U                                    */
/*                                                                            */
/*      Display menus on windows.                                             */
/*                                                                            */
/************************************************                             */
                                                      /*                      */
#include <supglob.c>                                  /* locate global data   */
                                                      /*                      */
menu(n,e)                                             /* display menu         */
    int n,                                            /* menu number          */
        e;                                            /* erase screen flag    */
    {static char *m1text[]={                          /* main menu text       */
     " ",
     "        HP/SAPMD  ACCESS",
     " ",
     "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[]={                          /* command/interrogate text  */
     "SG ddd/hh:mm:ss          Set GMT",
     "SM ddd/hh:mm:ss          Set elapsed time",
     "TM                       Read GMT and MET",
     "DR xx[,yy]               Dump 80C31 ram from xx for yy bytes",
     "DS xx                    Dump 80C31 SFR xx",
     "DE xxxx[,yy]             Dump 80C31 external memory xxxx for yy bytes",
     "ER xx                    Enter 80C31 ram at xx",
     "ES xx                    Enter 80C31 SFR xx",
     "EE xxxx                  Enter 80C31 external memory",
     "P filename[,xxxx]        Program filename into EEPROM at xxxx",
     "MON                      Toggle monitor data window display",0};
     static char *m3text[]={                          /* self-test text       */
     " ",
     "     HP/SAPMD SELF-TEST",
     " ",
     "1.  N/A",
     "2.  EEPROM TEST",
     "3.  N/A",
     "4.  N/A",
     "5.  N/A",
     "6.  80C31 RAM TEST",
     "7.  80C31 ROM TEST",
     " ",0};
     removw(activw);                                  /* remove current window     */
     if (e) erase(&screen);                           /* clear screen         */
     switch (n)                                       /* which menu?          */
        {case 1:                                      /* main menu            */
            show(m1);                                 /* display main menu    */
            pmenu(m1,m1text,1,20);                    /* display text         */
            activw=m1;                                /* flag active window   */
            break;                                    /* next                 */
         case 2:                                      /* command/interrogate  */
            show(m2);                                 /* display menu         */
            pmenu(m2,m2text,1,5);                     /* display menu text    */
            activw=m2;                                /* no window active     */
            break;                                    /* next                 */
         case 3:                                      /* self-test            */
            show(m3);                                 /* display self-test menu    */
            pmenu(m3,m3text,1,25);                    /* print text           */
            activw=m3;                                /* flag active window   */
            break;                                    /* next                 */
         case 4:                                      /* monitor window       */
```

```
            show(m5);                           /* display window            */
            activw=m5;                          /* active window             */
            break;                              /* next                      */
         case 5:                                /* display data window       */
            show(m6);                           /* display                   */
            activw=m6;}};                       /*                           */
                                                /*                           */
/*************************************************                           */
/*                                                                           */
/*                              P M E N U                                    */
/*                                                                           */
/*      Print menu text on screen.                                           */
/*                                                                           */
/*************************************************                           */
                                                /*                           */
pmenu(id,text,line,col)                         /* print menu                */
    struct window *id;                          /* window pointer            */
    char *text[];                               /* menu text                 */
    int line,col;                               /* display column            */
    {int i;                                     /* iteration variable        */
     char *ln;                                  /* current menu line index   */
     id->cury=line;                             /* move to top of window     */
     for (i=0;(ln=text[i])!=0;i++)              /* get a menu line           */
        {id->curx=col;                          /* skip spaces               */
         while (*ln!='\0') wchw(id,*ln++);      /* print line                */
         id->cury++;}}                          /* next line                 */
                                                /*                           */
                                                /*****************************/
```

```
/******************************************************************************/
/*                                                                           */
/*                              P R P R E S S                                */
/*                                                                           */
/*      Print pressure data file.                                            */
/*                                                                           */
/*************************************************                           */
                                                /*                           */
#include <supglob.c>                            /* locate global data        */
                                                /*                           */
prpress()                                       /* print data                */
    {int i,                                     /* iteration variable        */
         pnum;                                  /* current sample            */
     if (!getfile()) return(1);                 /* get filename              */
     freopen("sapmd.lst","w",stdout);           /* re-assign output stream   */
     for (pnum=0;&prsam[pnum]<samptr;pnum+=80)  /* print data                */
        ptpage(pnum);                           /* print page                */
     freopen("prn","w",stdout);                 /* re-assign output stream   */
     for (pnum=0;&prsam[pnum]<samptr;pnum+=80)  /* print data                */
        ptpage(pnum);                           /* print page                */
     freopen("con","w",stdout);                 /* display data              */
     return(0);}                                /* no error                  */
                                                /*                           */
/*************************************************                           */
/*                                                                           */
/*                              P T P A G E                                  */
/*                                                                           */
/*      Print a page of pressure data.                                       */
/*                                                                           */
/*************************************************                           */
                                                /*                           */
ptpage(sm)                                      /* display page              */
    int sm;                                     /* starting sample           */
    {static char *headr=                        /* window header             */
     {"SAMPLE               PRESSURE            SAMPLE               PRESSURE"};
     static char *title=                        /* page title                */
     {"                     SAPMD PRESSURE DATA      SAPMD SERIAL #"};
     static char *space={"      "};             /* blanks                    */
     int i,                                     /* iteration variable        */
         j,                                     /* iteration variable        */
         k;                                     /* iteration variable        */
     fputc(FF,stdout);                          /* new page                  */
     printf("%s%s%4d\n\n%s\n\n",iptr,title,prsam[sm].serial,headr);
     for (j=0;j<40;j++)                         /* 1 column                  */
        {psam(sm);                              /* print 1 column            */
         printf("%s","              ");         /* ...                       */
         psam(sm+40);                           /* print 2nd column          */
         printf("\n");                          /* new line                  */
         sm++;}}                                /* next sample               */
                                                /*                           */
/*************************************************                           */
/*                                                                           */
/*                               P S A M                                     */
/*                                                                           */
/*      Print a sample.                                                      */
/*                                                                           */
/*************************************************                           */
                                                /*                           */
psam(sm)                                        /* print sample poop         */
    int sm;                                     /* sample number             */
    {if (&prsam[sm]<samptr)                     /* good sample?              */
        printf("  %3d                  %5.2f",prsam[sm].sample,
                prsam[sm].press);}
                                                /*                           */
                                                /*****************************/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              R E C O V E R                                */
/*                                                                           */
/*      Retrieve pressure data.                                              */
/*                                                                           */
/*********************************************                               */
                                                /*                           */
#include <supglob.c>                            /* locate global data        */
                                                /*                           */
#define BUFSIZE 20000                           /* Data buffer size          */
                                                /*                           */
recover()                                       /* fetch data                */
    {FILE *pfile;                               /* pressure data file        */
     union {int i;                              /* sample index              */
            unsigned char b[5];} s;             /*                           */
     unsigned char rch,                         /* SAPMD response character  */
                   rchbuf[BUFSIZE];             /* Data buffer               */
     int i,                                     /* iteration variables       */
         n,                                     /* Buffer index              */
         cks;                                   /* checksum                  */
     prompt("ENTER FILENAME: ");                /* prompt for file           */
     i=rdln();                                  /* read filename             */
     if (i==HOME || i==LEFT) return(1);         /* bail out                  */
     skbl();                                    /* skip blanks to filename   */
     if (*iptr==CR) return(1);                  /* null line?                */
     for (i=0;i<sizeof(line);i++)               /* stomp EOL                 */
        if (line[i]==CR) line[i]=0;             /* ...                       */
     pfile=fopen(iptr,"wb");                    /* open file.                */
     if (sacmd(DUMPRESS,0,0))                   /* send dump command         */
        if (!versg(PRESSFILE))                  /* pressure data coming?     */
            {cks=0;                             /* clear checksum            */
             n = 0;                             /* zero buffer index         */
             while (1)                          /* read until end            */
                {for (i=0;i<2;i++)              /* read all data             */
                    {if ((rch=rdsg())==EOPDATA) goto l1; /* done?            */
                     if (rch==ABORT)            /* error?                    */
                        {p_error(13);           /* send error                */
                         fwrite(rchbuf,sizeof(rch),n,pfile); /* write data   */
                         fclose(pfile);         /* close file                */
                         return(1);};           /* bail out                  */
                     if (n < BUFSIZE)           /* room left in buffer?      */
                     {                          /*                           */
                        rchbuf[n] = rch;        /* put byte in buffer        */
                        n++;                    /* increment buffer index    */
                     }                          /*                           */
                     else                       /* if overrun, send error    */
                     {                          /*                           */
                        p_error(14);            /* send error                */
                        fwrite(rchbuf,sizeof(rch),n,pfile); /* write data    */
                        fclose(pfile);          /* close file                */
                        return(1);              /* bail out                  */
                     }                          /*                           */
/*   Next line no longer used                                                */
/*                   fputc(rch,pfile);             write character           */
                     s.b[i]=rch;};              /* make ascii byte           */
                 s.b[2]='\0';                   /* terminate string          */
                 cks+=bhex(&s);};               /* accumulate checksum       */
l1:                                             /* EOD                       */
             fwrite(rchbuf,sizeof(rch),n,pfile);/* write data buffer         */
             fclose(pfile);                     /* close file                */
             for (i=0;i<4;i++)                  /* read checksum             */
                if ((s.b[i]=rdsg())==ABORT)     /* error?                    */
                    {p_error(15);               /* send message              */
                     return(1);};               /* bail out                  */
             s.b[4]='\0';                       /* terminate string          */
        if (cks!=bhex(&s))                     /* compare checksums          */
           error(BADCHECK);                    /* send error                 */
        else                                   /* no error                   */
           {wchs(CR);                          /*  just info.                */
            stype("Pressure data recovered");}; /* print message             */
        return(1);};                           /* complete                   */
p_error(16);                                   /* strange response           */
return(1);}                                    /* return error               */
                                               /*                            */
                                               /*****************************/
```

```
/******************************************************************************/
/*                                                                           */
/*                              S E L F T E S T                              */
/*                                                                           */
/*      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'                    */
     unsigned char c,                           /* temporary                 */
                   rch,                         /* self test response        */
                   ech;                         /* error character           */
     int i;                                     /* temp                      */
     menu(3,1);                                 /* display menu              */
     while (1)                                  /* forever ...               */
        {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;                             /* save option               */
             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;                    /* adjust                    */
                        if (!sacmd(SELFTEST,&c,1)) /* 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 ((ech=rdsg())!=ABORT) /* get err */
                                            {stype(emsgs[ech-'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 l1;    /* skip ram poop             */
                                 case RAMERR:   /* ram self-test error       */
                                    wchs(CR);   /* new line                  */
                                    stype(ramerr); /* print message          */
l1:                                             /* 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;};            /* exit loop                 */
                   continue;               /* next option               */
                case 1:                    /* commands no longer avail  */
                case 3:
                case 4:                    /* A/D CONVERTOR TEST        */
                case 5:                    /* PRESSURE TRANDUCER TEST   */
                default:                   /* bad option                */
                   error(BADCMD);          /* print message             */
                   continue;};             /* next iteration            */
        break;}};                          /* next iteration            */
if (token==Q)                              /* quit?                     */
   {scrup(0,0,24,79,0);                    /* clear screen              */
    i=inp(0x21);                           /* read 8259 interrupt mask  */
    outp(0x21,i|0x10);                     /* stop serial interrupts    */
    exit(0);};                             /* stop.                     */
if (token==CMD)                            /* command file?             */
   {if (i=excfile()) error(i);}            /* open command file         */
else                                       /* not a command file?       */
   if (token!=EOL) error(BADCMD);}}        /* null line?                */
                                           /*                           */
                                           /*****************************/
```

```
/******************************************************************************/
/*                                                                            */
/*                              S T A T U S                                   */
/*                                                                            */
/*      Maintain status line and dump window.                                 */
/*                                                                            */
/**************************************************                           */
                                                /*                            */
#include <supglob.c>                            /* locate global data         */
                                                /*                            */
status()                                        /* display status             */
    {static char *statxt[]=                     /* status line text           */
     {"EEPROM-ON","SELF-TEST","GSE","ACQUISITION","COMPLETE","ERROR"};
     int i,                                     /* iteration variable         */
         j,                                     /* iteration variable         */
         k,                                     /* iteration variable         */
         adr,                                   /* dump address               */
         bct;                                   /* dump byte count            */
     unsigned char st;                          /* status byte                */
     st=rdst();                                 /* get status byte            */
     m4->curx=0;                                /* position cursor            */
     m4->cury=0;                                /* ...                        */
     for (i=0;i<6;i++)                          /* scan status bits           */
        {sat(0x0f);                             /* set obg                    */
         if ((st<<i)&0x20) sat(0xf8);           /* check for set bit          */
         wtype(m4,statxt[i]);                   /* display status             */
         m4->curx+=4;};                         /* next column                */
     m4->curx+=6;                               /* ...                        */
     stct=(stct+1)&0x7fff;                      /* increment message count    */
     cursor(m4);                                /* position cursor            */
     sat(0x0f);                                 /* obg                        */
     printf("%5d",stct);                        /* display count              */
     sat(7);                                    /* normal video               */
     if (st&0x40)                               /* check for status or dump   */
        {adr=rdst();                            /* get address                */
         bct=rdst();                            /* ... and byte count         */
         if (activw==m5)                        /* monitor window active?     */
            {m5->cury=0;                        /* position cursor            */
             for (i=bct;i>0;i-=16)              /* count bytes displayed      */
                {wchw(m5,CR);                   /* new line                   */
                 hexw(m5,adr);                  /* display address            */
                 wchw(m5,':');                  /* separate address           */
                 wchw(' ');                     /*                            */
                 adr+=16;                       /* next address               */
                 k=0;                           /* printed byte counter       */
                 for (j=i>16?16:i;j>0;j--)      /* count bytes on line        */
                    {wchw(m5,k++==8?'-':' ');   /*                            */
                     hex(m5,rdst());}}}         /* read and print byte        */
         else                                   /* no monitor window          */
             for (i=0;i<bct;i++) rdst();}}      /* discard data               */
                                                /*                            */
                                                /******************************/
```

```
/******************************************************************************/
/*                                                                            */
/*                         GLOBAL DECLARATIONS                                */
/*                                                                            */
/*************************************************                            */
                                                /*                            */
#include <stdio.h>                              /* get file poop              */
#include <process.h>                            /* get exit                   */
#include <stdlib.h>                             /* get toupper                */

#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                             /* ascii code: backspace      */
#define CR 0xd                                  /* ascii code: carriage ret.  */
#define LF 0xa                                  /* ascii code: line feed      */
#define FF 'L'-0x40                             /* ascii code: form feed      */
#define TAB 9                                   /* ascii code: tab            */
#define CTRLC 3                                 /* ascii code: control-C      */
#define CTRLA 1                                 /* ascii code: control-A      */
#define CTRLR 0x12                              /* ascii code: control-R      */
#define ESC 0x1b                                /* ascii code: escape         */
#define CEOF 0x1a                               /* ascii code: eof            */
#define SPL 0                                   /* IBM code: keypad char. seq*/
#define DULC 0xc9                               /* IBM code: doub. UL corner */
#define DURC 0xbb                               /* IBM code: doub. UR corner */
#define DUMD 0xcb                               /* IBM code: doub. UP middle */
#define DLMD 0xca                               /* IBM code: doub. LO middle */
#define DLLC 0xc8                               /* IBM code: doub. LL corner */
#define DLRC 0xbc                               /* IBM code: doub. LR corner */
#define DLN 0xcd                                /* IBM code: doub. line      */
#define DVB 0xba                                /* IBM code: doub. vert. line*/
#define SLN 0xc4                                /* IBM code: single line     */
#define SVB 0xb3                                /* IBM code: sngl. vert. line*/
#define DRSLN 0xc7                              /* IBM code: left doub. sngl.*/
#define SUVB 0xc2                               /* IBM code: sngl. DN middle */
#define DLSLN 0xb6                              /* IBM code: rght. doub. sngl*/
#define DLDLN 0xcc                              /* IBM code: doub. left      */
#define SLVB 0xcf                               /* IBM code: sngl. UP middle */
#define DRDLN 0xb9                              /* IBM code: doub. right     */
#define DMSLN 0xd7                              /* IBM code: middle          */
                                                /*                            */
#define NUL 0                                   /* token: nothing             */
#define UP 1                                    /* token: up arrow            */
#define DOWN 2                                  /* token: down arrow          */
#define LEFT 3                                  /* token: left arrow          */
#define RIGHT 4                                 /* token: right arrow         */
#define PGDN 5                                  /* token: page down           */
#define PGUP 6                                  /* token: page up             */
#define INS 7                                   /* token: insert              */
#define DEL 8                                   /* token: del                 */
#define NUMBER 9                                /* token: number              */
#define EOL 10                                  /* token: carriage return     */
```

```
#define CTA 11                          /* token: control-A          */
#define CTC 12                          /* token: control-C          */
#define CTR 13                          /* token: control-R          */
#define COMMA 14                        /* token: comma              */
#define SG 15                           /* token: set-gmt            */
#define SM 16                           /* token: set-met            */
#define TM 17                           /* token: time               */
#define ES 18                           /* token: enter SFR          */
#define COLON 19                        /* token: colon              */
#define DR 20                           /* token: dump ram           */
#define DE 21                           /* token: dump code          */
#define ER 22                           /* token: enter ram          */
#define EE 23                           /* token: enter external mem */
#define EQU 26                          /* token: equal sign         */
#define Q 42                            /* token: quit               */
#define P 43                            /* token: program            */
#define HOME 44                         /* token: home               */
#define ND 45                           /* token: end                */
#define MON 46                          /* token: monitor            */
#define DS 47                           /* token: dump SFR           */
#define SLASH 48                        /* token: '/'                */
#define LW 49                           /* token: load windows       */
#define IB 50                           /* token: input from port    */
#define OB 51                           /* token: output to port     */
#define E 53                            /* token: enter current type */
#define CMD 54                          /* token: at sign            */
#define LA 55                           /* token: rubber launch      */
                                        /*                           */
#define SETGMT 'P'                      /* SAPMD command: set-gmt    */
#define SETMET 'Q'                      /* SAPMD command: set-met    */
#define DUMPRAM 'I'                     /* SAPMD command: dump ram   */
#define DUMPSFR 'M'                     /* SAPMD command: dump SFR   */
#define DUMPEXT 'J'                     /* SAPMD command: dump code  */
#define LOADRAM 'G'                     /* SAPMD command: load ram   */
#define LOADSFR 'L'                     /* SAPMD command: load SFR   */
#define LOADEE 'H'                      /* SAPMD command: load EEPROM*/
#define SELFTEST 'K'                    /* SAPMD command: self-test  */
#define DUMPRESS 'N'                    /* SAPMD command: dump press.*/
#define ILNK 'Z'                        /* SAPMD command: abort      */
                                        /*                           */
#define RAMDATA 'G'                     /* SAPMD response: ram data  */
#define EXTDATA 'J'                     /* SAPMD response: code data */
#define TESTCOMP 'K'                    /* SAPMD response: test comp.*/
#define SFRDATA 'M'                     /* SAPMD response: SFR data  */
#define PRESSFILE 'N'                   /* SAPMD response: press. dta*/
#define EOPDATA 'P'                     /* SAPMD response: EOD       */
#define EEPERR 'U'                      /* SAPMD response: EEPROM err*/
#define RAMERR 'V'                      /* SAPMD response: ram error */
#define ABORT 'X'                       /* SAPMD response: error     */
#define ACK 'W'                         /* SAPMD response: complete  */
                                        /*                           */
#define GOTIT 0                         /* message: data recovered   */
#define BADSAPMD 1                      /* error code: SAPMD error   */
#define BADCHECK 2                      /* error code: checksum error*/
#define BADCMD 3                        /* error code: bad command   */
#define BADFILE 4                       /* error code: bad hex file  */
#define BADCHAR 5                       /* error code: bad character */
#define NOFILE 8                        /* error code: file not found*/
#define EOFERR 9                        /* error code: early EOF     */
#define NONAME 10                       /* error code: no filename   */
#define BADCAL 11                       /* error code: bad cal. file */
#define NOLAFILE 12                     /* error code: no la.cmd     */
                                        /*                           */
#define GMTADR 0x14                     /* 80C51 address: GMT & MET  */
                                        /*                           */
#define BL 0                            /* window type: blank        */
```

```
#define DBT 1                                    /* window type: dumped bytes */
#define DWD 2                                    /* window type: dumped words */
#define DA 3                                     /* window type: disassembly  */
                                                 /*                           */
struct window {int scry;                         /* window context block      */
               int curx;                         /* cursor position           */
               int cury;                         /*                           */
               int lines;                        /* number of lines in window */
               int daseg;                        /* address of displayed data */
               int daoff;                        /* ... offset                */
               int topoff;                       /* addr. of top instr. (DA)  */
               char wnum;                        /* window number             */
               char disp;                        /* displayed flag            */
               char type;                        /* window contents flag      */
               char ovr;                         /* segment override flag     */
               char used;};                      /* in-use flag               */
                                                 /*                           */
struct sam {int sample;                          /* sample number             */
            int serial;                          /* SAPMD serial #            */
            float press;
                        } huge prsam[4100],      /* pre-processed samples     */
                          huge *samptr;          /* last processed sample     */
struct cal {int serial;                          /* SAPMD serial #            */
            int offset;                          /* SAPMD transducer adjust.  */
            float coef;} sapmd[100];              /* SAPMD calibration coefs.  */
                                                 /*                           */
struct window screen                             /* underlying screen         */
#ifdef MAIN                                      /* fool worthless compiler   */
                    ={0,0,23,24,0,0,0,0,1,0,-1,1} /* initialize screen        */
#endif                                           /* compiler fooled           */
             ,box[8]                             /* dump windows              */
#ifdef MAIN                                      /* fool compiler             */
                    ={0,0,0,0,0,0,0,'0',0,0,-1,0, /* initialize seg ovr flag */
                      0,0,0,0,0,0,0,'1',0,0,-1,0, /*                         */
                      0,0,0,0,0,0,0,'2',0,0,-1,0, /*                         */
                      0,0,0,0,0,0,0,'3',0,0,-1,0, /*                         */
                      0,0,0,0,0,0,0,'4',0,0,-1,0, /*                         */
                      0,0,0,0,0,0,0,'5',0,0,-1,0, /*                         */
                      0,0,0,0,0,0,0,'6',0,0,-1,0, /*                         */
                      0,0,0,0,0,0,0,'7',0,0,-1,0} /*                         */
#endif                                           /* compiler fooled           */
              ;                                  /*                           */
                                                 /*                           */
struct window *scline[26]                        /* line directory            */
#ifdef MAIN                                      /* fool worthless compiler   */
                         ={0,0,0,0,0,            /* window-on-line flags      */
                           0,0,0,0,0,            /*                           */
                           0,0,0,0,0,            /*                           */
                           0,0,0,0,0,            /*                           */
                           0,0,0,0,0,0}          /* all null                  */
#endif                                           /* compiler fooled           */
             ,*activw                            /* currently active window   */
#ifdef MAIN                                      /* fool worthless compiler   */
                        =0                       /* start with screen         */
#endif                                           /*                           */
             ;                                   /*                           */
                                                 /*                           */
char ch,                                         /* character temp            */
     line[128];                                  /* keyboard command line     */
char *iptr                                       /* command input index       */
#ifdef MAIN                                      /* fool worthless compiler   */
     =line                                       /* initial index             */
#endif                                           /* compiler fooled           */
     ;                                           /*                           */
int token                                        /* token id of lexical unit  */
   ,acc;                                         /* accumulated number        */
```

```
                                                /*                          */
int op0,                                        /* disassembly opcode byte 0 */
    c,                                          /* characters in byte        */
    cflag,                                      /* byte changed flag         */
    echo                                        /* echo kb input flag        */
#ifdef MAIN                                     /* fool compiler             */
    =1                                          /* default to echo           */
#endif                                          /* compiler fooled           */
    ;                                           /*                          */
int stct                                        /* status transmission count */
#ifdef MAIN                                     /* fool compiler             */
    =0                                          /* clear                     */
#endif                                          /* compiler fooled           */
    ;                                           /*                          */
int cmdfile                                     /* command file flag         */
#ifdef MAIN                                     /* fool compiler             */
    =0                                          /* ...                       */
#endif                                          /*                          */
    ;                                           /*                          */
                                                /*                          */
struct window *m1,*m2,*m3,*m4,*m5,*m6;          /* menu windows              */
                                                /*                          */
FILE *cfile;                                    /* command file              */
                                                /*                          */
extern struct window *createw();                /* func: create dump window  */
                                                /*                          */
                                                /*****************************/
```

```
/******************************************************************************/
/*                                                                            */
/*                              W I N D O W                                   */
/*                                                                            */
/*      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        */
                                                 /*                           */
/*************************************************                            */
/*                                                                            */
/*                              C R E A T E W                                 */
/*                                                                            */
/*      Create a dump window on the screen so it can be dumped all over.     */
/*                                                                            */
/*************************************************                            */
                                                 /*                           */
struct window *createw(id,linect)                /* create a window           */
    int id,                                      /* window id                 */
        linect;                                  /* size of window            */
    {if (!box[id].used)                          /* free slot?                */
        {box[id].used=1;                         /* claim slot                */
         box[id].lines=linect;                   /* set # lines in box        */
         box[id].curx=1;                         /* set cursor position       */
         box[id].cury=1;                         /*                           */
         box[id].disp=0;                         /* not displayed             */
         box[id].type=BL;                        /* window blank              */
         return(&box[id]);};                     /* return window id          */
     return(0);}                                 /* can't create window       */
                                                 /*                           */
/*************************************************                            */
/*                                                                            */
/*                              S H O W                                       */
/*                                                                            */
/*      Attempt to place the specified window on the screen.  Return success */
/* or failure indicator.                                                      */
/*                                                                            */
/*************************************************                            */
                                                 /*                           */
show(id)                                         /* present window            */
    struct window *id;                           /* window id                 */
    {int i,                                      /* iteration variable        */
         j;                                      /* iteration variable        */
     if (id->disp) return(1);                    /* window already displayed  */
     for (i=0;i<=24;i++)                         /* check all lines on screen */
        if (!scline[i])                          /* check line for window     */
            {if (25-i<id->lines) break;          /* enough lines left?        */
             for (j=i;j<i+id->lines;j++) scline[j]=id; /* mark line in use   */
             id->scry=i;                         /* mark screen row           */
             screen.scry=j;                      /* top free screen line      */
             screen.lines-=id->lines;            /* reduce lines on screen    */
             screen.cury-=id->lines;             /* adjust cursor within      */
             erase(id);                          /* blank window              */
             frame(id);                          /* draw its frame            */
             id->disp=1;                         /* mark window displayed     */
             return(1);};                        /* return window id          */
     return(0);}                                 /* can't create window       */
                                                 /*                           */
/*************************************************                            */
/*                                                                            */
/*                              R E M O V W                                   */
/*                                                                            */
/*      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)                                      /* delete window and erase    */
    struct window *id;                          /* window id                  */
    {int i,                                     /* iteration variable         */
         j,                                     /* iteration variable         */
         wbot;                                  /* first line past window     */
     if (!id) return(1);                        /* is this a window?          */
     if (!id->used) return(0);                  /* does window exist?         */
     if (!id->disp) return(1);                  /* is it displayed?           */
     wbot=id->scry+id->lines;                   /* first line past window     */
     for (i=wbot;scline[i];i+=scline[i]->lines) /* last window line           */
        {scline[i]->scry-=id->lines;            /* adjust screen row for move*/
         for (j=scline[i]->scry;                /* update line flag table     */
              j<scline[i]->scry+scline[i]->lines;j++) /* for new window       */
            scline[j]=scline[i];};              /* moved window               */
     scrup(id->scry,0,i-1,79,id->lines);        /* shift windows              */
     for (j=id->lines;j>0;j--) scline[i-j]=0;   /* flag scrolled lines free   */
     screen.scry-=id->lines;                    /* adjust top of screen       */
     screen.cury+=id->lines;                    /* adjust cursor within       */
     screen.lines+=id->lines;                   /* adjust lines on screen     */
     id->disp=0;                                /* flag not displayed         */
     if (id==activw) actv(scline[0]);           /* new activw window          */
     return(1);}                                /* free box                   */
                                                /*                            */
/**************************************************                           */
/*                                                                            */
/*                              W C H W                                       */
/*                                                                            */
/*      Write the passed character on the specified window at the current     */
/* cursor position and bump the cursor.                                       */
/*                                                                            */
/**************************************************                           */
                                                /*                            */
wchw(id,ch)                                     /* write ch on window[id]     */
    struct window *id;                          /* window id                  */
    char ch;                                    /* character                  */
    {if (!id->used) return(0);                  /* no such window             */
     if (ch==CR)                                /* check for new line         */
        {id->curx=1;                            /* cursor to start of line    */
         if (id->cury==id->lines-2)             /* check for bottom           */
            uscroll(id);                        /* scroll window              */
         else                                   /* not at bottom line         */
            id->cury++;                         /* next line                  */
         cursor(id);}                           /* move cursor                */
     else                                       /* not cr or lf               */
        {cursor(id);                            /* position cursor            */
         wch(ch);                               /* write character            */
         if (id->curx<80) id->curx++;};         /* at end of line?            */
     return(1);}                                /* return success             */
                                                /*                            */
/**************************************************                           */
/*                                                                            */
/*                              W C H S                                       */
/*                                                                            */
/*      Write the passed character on the screen window at the current        */
/* cursor position and bump the cursor.                                       */
/*                                                                            */
/**************************************************                           */
                                                /*                            */
wchs(ch)                                        /* write ch on window[id]     */
    char ch;                                    /* character                  */
    {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) screen.curx++;};    /* at end of line?           */
     return(1);}                                 /* return success            */
                                                 /*                           */
/**************************************************                           */
/*                                                                            */
/*                              C U R S O R                                   */
/*                                                                            */
/*      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               */
                                                 /*                           */
/**************************************************                           */
/*                                                                            */
/*                              U S C R O L L                                 */
/*                                                                            */
/*      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                */
                                                 /*                           */
/**************************************************                           */
/*                                                                            */
/*                              D S C R O L L                                 */
/*                                                                            */
/*      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                */
                                                 /*                           */
/**************************************************                           */
/*                                                                            */
/*                              F R A M E                                     */
/*                                                                            */
/*      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);                                 /* draw upper right corner   */
     for (i=id->scry+1;i<id->scry+id->lines-1;i++) /* draw sides             */
        {movcurs(i,0);                          /* move to left side         */
         wch(DVB);                              /* draw left border          */
         movcurs(i,79);                         /* move to right side        */
         wch(DVB);};                            /* draw right border         */
     movcurs(i,0);                              /* move to line start        */
     wch(DLLC);                                 /* draw lower left corner    */
     for (i=1;i<79;i++) wch(DLN);               /* draw lower border         */
     wch(DLRC);}                                /* draw lower right corner   */
                                                /*                           */
/*************************************************                           */
/*                                                                           */
/*                              E R A S E                                    */
/*                                                                           */
/*      Blank the specified window.                                          */
/*                                                                           */
/*************************************************                           */
                                                /*                           */
erase(id)                                       /* blank window              */
    struct window *id;                          /* 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;                          /* 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)                                        /* change active window      */
    struct window *id;                          /* new active window         */
    {activw=id;}                                /* change activw window      */
                                                /*                           */
                                                /*****************************/
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                               C O N I O
;
;       This collection performs console I/O in "don't help me" mode (direct
; console I/O).  These are C callable functions.
;
; Local symbols:
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
        .8086                           ; select instruction set
        PUBLIC  _RDC,_WRCH,_TICK
                                        ;
_TEXT   SEGMENT BYTE PUBLIC 'CODE'      ; code segment
        ASSUME  CS:_TEXT                ;
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                               _ W R C H
;
;       Write a character to the screen.  Character is passed as parameter.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_WRCH   PROC    NEAR                    ; Write to console
        PUSH    BP                      ; perform C procedure entry
        MOV     BP,SP                   ; ...
        MOV     DX,4[BP]                ; get character
        MOV     AH,6                    ; get MS-DOS function code
        INT     21H                     ; write character
        POP     BP                      ; restore ...
        RET                             ; ---> return
_WRCH   ENDP                            ; end-of-_WRCH
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                               _ R D C H
;
;       Check for a keyboard input character.  Return 0FFH if no character
; available, character otherwise.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_RDC    PROC    NEAR                    ; Read from console
        MOV     AH,6                    ; get MS-DOS function code
        MOV     DL,0FFH                 ; request input
        INT     21H                     ; read console
        JNZ     RD1                     ; character ready?
        MOV     AL,0FFH                 ; ... nope, return no character
RD1:                                    ; character in AL
        XOR     AH,AH                   ; clear for return
        RET                             ; ---> return
_RDC    ENDP                            ; end-of-_RDC
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                               _ T I C K
;
;       Check for an interval timeout.  Timeout flag is set by interval
; timer interrupt.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_TICK   PROC    NEAR                    ; check for timeout
        MOV     AX,SEG TIMER            ; point to timer flag
        MOV     ES,AX                   ; ...
        MOV     AX,ES:TIMER             ; get flag
        RET                             ; ---> return
_TICK   ENDP                            ;
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                       TIMER INTERRUPT HANDLER
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
TIMPOOP SEGMENT PARA PUBLIC             ; timer interrupt handler data
TIMER   DW      0                       ; interrupt flag
TIMPOOP ENDS                            ; end-of-TIMPOOP
                                        ;
_TEXT   ENDS                            ; end-of-_TEXT
                                        ;
        END                             ; end-of-console I/O routines
        PAGE    ,132
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                               F I O
;
;       This collection of routines performs SAPMD I/O.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
        .8086                           ; select instruction set
        PUBLIC  _COMIO,_WRSG,_RDS,SERINT,_RDST,_POLST,_POLSG,_PURGE
                                        ;
_TEXT   SEGMENT BYTE PUBLIC 'CODE'      ; code segment
        ASSUME  CS:_TEXT,DS:SERDATA
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                               _ C O M I O
;
;       Take the poop passed in parameters and perform I/O over COM:
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_COMIO  PROC    NEAR                    ; COM: I/O
        PUSH    BP                      ; perform C procedure entry protocol
        MOV     BP,SP                   ; point to parameters
        MOV     AH,6[BP]                ; get command
        MOV     DX,03F8H+3              ; point at RS232 card
        MOV     AL,80H                  ; get init command
        OUT     DX,AL                   ; initialize card
        JMP     SHORT $+2               ; delay
        DEC     DX                      ; point to baud rate
        DEC     DX                      ; ...
        MOV     AL,0                    ; get high baud rate
        OUT     DX,AL                   ; set high baud rate count
        DEC     DX                      ; back up address
        JMP     SHORT $+2               ; delay
        MOV     AL,18                   ; get low divisor (18=6400 baud)
        OUT     DX,AL                   ; set low divisor
        ADD     DX,3                    ; new address
        MOV     AL,AH                   ; get parameter
        AND     AL,01FH                 ; get character specs
        OUT     DX,AL                   ; set character poop
        PUSH    AX                      ; save
        CLI                             ; lock
        MOV     DX,3F8H                 ; point to serial port
        IN      AL,DX                   ; read away any garbage
        JMP     SHORT $+2               ; delay
        IN      AL,21H                  ; read 8259 mask register
        AND     AL,0EFH                 ; clear serial mask
        OUT     21H,AL                  ; set mask
        XOR     AX,AX                   ; get 0
        MOV     ES,AX                   ; point to vectors
        MOV     ES:WORD PTR 30H,OFFSET SERINT; initialize serial vector
        MOV     ES:WORD PTR 32H,SEG SERINT ; ...
        INC     DX                      ; point to interrupt mask on serial cd.
        MOV     AL,1                    ; enable receive interrupts
        OUT     DX,AL                   ; ...
        MOV     DX,3FCH                 ; modem control reg.
        MOV     AL,0BH                  ;
        OUT     DX,AL                   ;
        STI                             ; unlock
        POP     AX                      ; restore
        POP     BP                      ; ...
        RET                             ; ---> return
_COMIO  ENDP                            ; end-of-_COMIO
```

```
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                               _ W R S G
;
;       Write a character to the SAPMD.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_WRSG   PROC    NEAR                    ; write SAPMD
        PUSH    BP                      ; perform C procedure entry protocol
        MOV     BP,SP                   ; point to parameters
        MOV     DX,03F8H+5              ; get status port address
WRS1:                                   ; wait for txrdy
        IN      AL,DX                   ; read status
        TEST    AL,40H                  ; check txrdy
        JZ      WRS1                    ; empty?
        MOV     DX,3FEH                 ; get port address
WRS2:                                   ; wait for clear-to-send
        IN      AL,DX                   ; read status reg
        TEST    AL,10H                  ; test clear-to-send
        JZ      WRS2                    ; SAPMD ready?
        MOV     AL,4[BP]                ; ... yep, get character
        MOV     DX,03F8H                ; get port address
        OUT     DX,AL                   ; send character
        POP     BP                      ; ...
        RET                             ; ---> return
_WRSG   ENDP                            ; end-of-_WRSG
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                               _ R D S G
;
;       Read a response character from SAPMD.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_RDS    PROC    NEAR                    ; read SAPMD response byte
        PUSH    DS                      ; save
        MOV     AX,SEG SERDATA          ; point to input buffers
        MOV     DS,AX                   ; ...
        MOV     BX,RBOUTP               ; get output pointer
RD1:                                    ; check buffer level
        CMP     BX,RBINP                ; compare with input pointer
        JE      RD1                     ; any data in buffer?
        MOV     AL,RBUF[BX]             ; get character
        MOV     AH,0                    ; clear upper word
        INC     BX                      ; move to next position
        AND     BX,0FFH                 ; mod 256
        MOV     RBOUTP,BX               ; save
        POP     DS                      ; ...
        RET                             ; ---> return
_RDS    ENDP                            ; end-of-_RDS
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                               _ R D S T
;
;       Read a status character from SAPMD.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_RDST   PROC    NEAR                    ; read SAPMD status byte
        PUSH    DS                      ; save
        MOV     AX,SEG SERDATA          ; point to input buffers
        MOV     DS,AX                   ; ...
        MOV     BX,DBOUTP                   ; get output pointer
RDS1:                                       ; check buffer level
        CMP     BX,DBINP                    ; compare with input pointer
        JE      RDS1                        ; any data in buffer?
        MOV     AL,DBUF[BX]                 ; get character
        MOV     AH,0                        ; clear upper word
        INC     BX                          ; move to next position
        AND     BX,0FFH                     ; mod 256
        MOV     DBOUTP,BX                   ; save
        POP     DS                          ; ...
        RET                                 ; ---> return
_RDST   ENDP                                ; end-of-_RDST
                                            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                           _ P O L S T
;
;       Check for status character from SAPMD.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                            ;
_POLST  PROC    NEAR                        ; check for SAPMD status byte
        PUSH    DS                          ; save
        MOV     AX,SEG SERDATA              ; point to input buffers
        MOV     DS,AX                       ; ...
        XOR     AX,AX                       ; get data avail flag
        MOV     BX,DBOUTP                   ; get output pointer
        CMP     BX,DBINP                    ; compare with input pointer
        JNE     POLS1                       ; any data in buffer?
POLS2:                                      ; _POLSG entry
        DEC     AX                          ; flag no data
POLS1:                                      ; restore and return
        POP     DS                          ; ...
        RET                                 ; ---> return
_POLST  ENDP                                ; end-of-_POLST
                                            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                           _ P O L S G
;
;       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
_POLSG  ENDP                                ; end-of-_POLSG
                                            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                           _ P U R G E
;
;       Empty SAPMD response buffer.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                            ;
_PURGE  PROC    NEAR                        ; delete responses
        PUSH    DS                          ; save
        MOV     AX,SEG RBINP                ; point to response buffer pointers
        MOV     DS,AX                    ; ...
        CLI                              ; lock
        MOV     RBINP,0                  ; clear pointers
        MOV     RBOUTP,0                 ; ...
        STI                              ; unlock
        POP     DS                       ; restore
        RET                              ; ---> return
_PURGE  ENDP                             ; end-of-PURGE
                                         ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                                S E R I N T
;
;       Serial interrupt handler.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                         ;
SERINT:                                  ; serial interrupt
        PUSH    AX                       ; save registers
        PUSH    DX                       ; ...
        PUSH    BX                       ; ...
        PUSH    SI                       ; ...
        PUSH    ES                       ; ...
        PUSH    DS                       ; ...
        STI                              ; unlock
        MOV     AX,SEG SERDATA           ; point to local data
        MOV     DS,AX                    ; ...
        MOV     DX,3F8H                  ; point to serial buffer
        IN      AL,DX                    ; read character
        CMP     DMFLG,0                  ; check for SAPMD ram dump
        JNE     SERIN4                   ; data coming?
        TEST    AL,80H                   ; check for status or response
        JNZ     SERIN1                   ; status
        MOV     SI,RBINP                 ; get buffer index
        MOV     RBUF[SI],AL              ; plant character
        INC     SI                       ; point to next character
        AND     SI,0FFH                  ; mod 256
        CMP     SI,RBOUTP                ; check with output pointer
        JNE     SERIN2                   ; overflow?
        MOV     SI,RBINP                 ; restore pointer
SERIN2:                                  ; restore and return
        MOV     RBINP,SI                 ; save input pointer
SERIN3:                                  ; exit
        MOV     AL,20H                   ; get EOI code
        OUT     20H,AL                   ; release 8259
        POP     DS                       ; restore ...
        POP     ES                       ; ...
        POP     SI                       ; ...
        POP     BX                       ; ...
        POP     DX                       ; ...
        POP     AX                       ; ...
        IRET                             ; ---> resume
SERIN1:                                  ; status or dump
        TEST    AL,40H                   ; check for status or dump
        JZ      SERIN5                   ; status?
SERIN6:                                  ; count byte
        INC     DMFLG                    ; ... nopė, dump. Flag it.
SERIN5:                                  ; status
        MOV     SI,DBINP                 ; get buffer index
        MOV     DBUF[SI],AL              ; plant character
        INC     SI                       ; point to next character
        AND     SI,0FFH                  ; mod 256
        CMP     SI,DBOUTP                ; check with output pointer
        JNE     SERIN8                   ; overflow?
        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
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                              _ W I N D O W
;
;       These routines support the C routines in WINDOW.C.  The perform
; direct screen control using the ROM BIOS.
;
; Local symbols:
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
        .8086                           ; select instruction set
        PUBLIC  _WCH,_SCRUP,_SCRDN,_MOVCURS,_SETVPAGE,_SCH,_SAT
        PUBLIC  _PSHCURS,_SETCURS,_POPCURS
                                        ;
_TEXT   SEGMENT BYTE PUBLIC 'CODE'      ; code segment
        ASSUME  CS:_TEXT
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                              _ W C H
;
;       Write a character to the screen.  Character is passed as parameter.
; Character is written at current cursor position using current attributes.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_WCH    PROC    NEAR                    ; Write to console
        PUSH    BP                      ; perform C procedure entry
        MOV     BP,SP                   ; ...
        MOV     AX,SEG WINDPOOP         ; point to window data
        MOV     ES,AX                   ; ...
        MOV     AX,4[BP]                ; get character
        MOV     BH,ES:VPAGE             ; get video page number
        MOV     CX,1                    ; get character count
        MOV     AH,9                    ; get BIOS function code
        MOV     BL,ES:ATTRIB            ; get current attributes
        INT     10H                     ; write character
        MOV     AH,3                    ; get command
        INT     10H                     ; read cursor position
        CMP     DL,79                   ; check column
        JGE     WCH1                    ; end-of-line?
        INC     DL                      ; nope, next column
        MOV     AH,2                    ; get command
        INT     10H                     ; set cursor position
WCH1:                                   ; don't move cursor
        POP     BP                      ; restore ...
        RET                             ; ---> return
_WCH    ENDP                            ; end-of-_WCH
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                              _ M O V C U R S
;
;       Move the cursor to the specified screen position.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_MOVCURS PROC   NEAR                    ; move 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     DH,4[BP]                ; ...
        MOV     DL,6[BP]                ; ...
        MOV     AH,2                    ; get command code
        INT     10H                     ; position cursor
        POP     BP                      ; ...
        RET                             ; ---> return
_MOVCURS ENDP                           ; end-of-_MOVCURS
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                                 _ P S H C U R S
;
;       Save the current cursor attributes.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_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
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                                 _ P O P C U R S
;
;       Restore the current cursor attributes.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_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 postion
        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:                                   ; no saved cursor
        POP     SI                      ; ... restore
        POP     BP                      ; ...
        RET                             ; ---> return
_POPCURS ENDP                           ; end-of-_POPCURS
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                                 _ S E T C U R S
```

```
;
;       Set the current cursor attributes.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_SETCURS PROC   NEAR                    ; set cursor
        PUSH    BP                      ; save
        MOV     BP,SP                   ; point to parameters
        MOV     CX,4[BP]                ; get attributes
        MOV     AH,1                    ; get function code
        INT     10H                     ; read cursor poop
        POP     BP                      ; ...
        RET                             ; ---> return
_SETCURS ENDP                           ; end-of-_SETCURS
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                                 _ S C R U P
;
;       Scroll window up.  Bottom line is blanked.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_SCRUP  PROC    NEAR                    ; Scroll window up
        MOV     AH,6                    ; get function code
SCR1:                                   ; _SCRDN entry
        PUSH    BP                      ; save
        MOV     BP,SP                   ; point to parameters
        push    ax
        MOV     ax,SEG WINDPOOP         ; point to window poop
        MOV     ES,ax                   ; ...
        pop     ax
        MOV     BH,ES:ATTRIB            ; get attributes
        MOV     CH,4[BP]                ; get upper left row
        MOV     CL,6[BP]                ; get upper left column
        MOV     DH,8[BP]                ; get lower right row
        MOV     DL,10[BP]               ; get lower right column
        MOV     AL,12[BP]               ; get # lines
        INT     10H                     ; scroll
        POP     BP                      ; restore
        RET                             ; ---> return
_SCRUP  ENDP                            ; end-of-scroll up
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                                 _ S C R D N
;
;       Scroll active window down.  Top line is blanked.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_SCRDN  PROC    NEAR                    ; scroll window down
        MOV     AH,7                    ; get function code
        JMP     SCR1                    ; go scroll
_SCRDN  ENDP                            ; end-of-_SCRDN
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                                 _ S E T V P A G E
;
;       Set video page number displayed on screen.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_SETVPAGE PROC  NEAR                    ; set video page number
        PUSH    BP                      ; save
        MOV     BP,SP                   ; point to parameter
        MOV     AX,SEG VPAGE            ; point at page #
        MOV     ES,AX                   ; ...
        MOV     AL,4[BP]                ; get page #
        MOV     ES:VPAGE,AL             ; plant page #
        MOV     AH,5                    ; get BIOS function code
        INT     10H                     ; change page
        POP     BP                      ; restore
        RET                             ; ---> return
_SETVPAGE ENDP                          ; end-of-_SETVPAGE
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                               _ S C H
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_SCH    PROC    NEAR                    ; read character from screen
        MOV     AX,SEG VPAGE            ; point at page #
        MOV     ES,AX                   ; ...
        MOV     BH,ES:VPAGE             ; get video page #
        MOV     AH,8                    ; get command code
        INT     10H                     ; read character
        XOR     AH,AH                   ; clear attributes
        RET                             ; ---> return
_SCH    ENDP                            ; end-of-_SCH
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       Set character attributes
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
_SAT    PROC    NEAR                    ; set character attributes
        PUSH    BP                      ; save
        MOV     BP,SP                   ; point to stack
        MOV     AX,SEG ATTRIB           ; point to attributes
        MOV     ES,AX                   ; ...
        MOV     AX,4[BP]                ; get attributes
        MOV     ES:ATTRIB,AL            ; plant new attributes
        POP     BP                      ; restore
        RET                             ; ---> return
_SAT    ENDP                            ; end-of-_SAT
                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                               LOCAL DATA
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;
WINDPOOP SEGMENT PARA PUBLIC            ; window data
ATTRIB  DB      07h                     ; character attributes
VPAGE   DB      0                       ; current video page
CURPTR  DW      0                       ; cursor stack pointer
RCOL    DW      0,0,0                   ; cursor row and column
CTYPE   DW      0,0,0                   ; cursor type
WINDPOOP ENDS                           ; end-of-local data
                                        ;
_TEXT   ENDS                            ; end-of-_TEXT
                                        ;
        END                             ; end-of-console I/O routines
```

HP/SGA "READ ME"

README.HP January 3, 1989

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

1062-ATP-01
March 1988

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 : 4/07/88
Quality Assurance Engineer

SOUTHWEST RESEARCH INSTITUTE
P.O. DRAWER 28510
SAN ANTONIO, TEXAS 78284

Change Log Here

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

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 Modification 4S	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
1062-SGA-01	Shuttle Gauge Access Software User's Guide
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. Conconformance

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.

ITEM MODEL MANUFACTURER S/N CAL DUE

A) ____

B) ____

D) ____

E) ____

F) ____

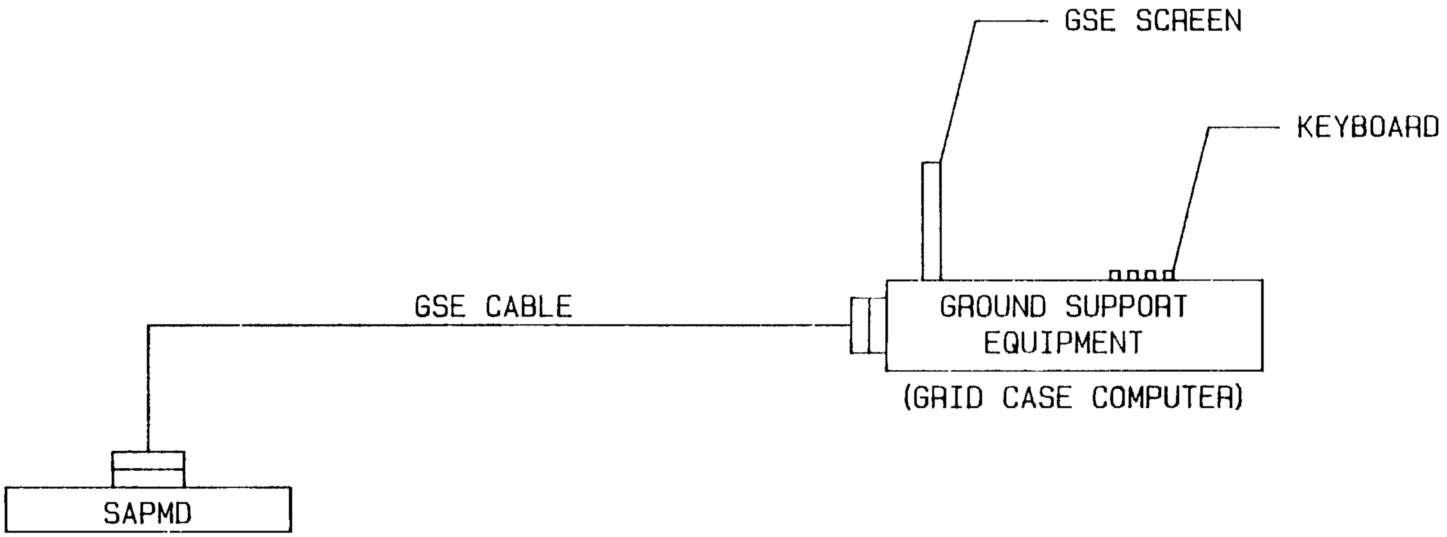

FIGURE 4.1 INITIAL ELECTRICAL TEST CONFIGURATION

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.

ITEM MODEL MANUFACTURER S/N CAL DUE

A)

B)

D)

E)

F)

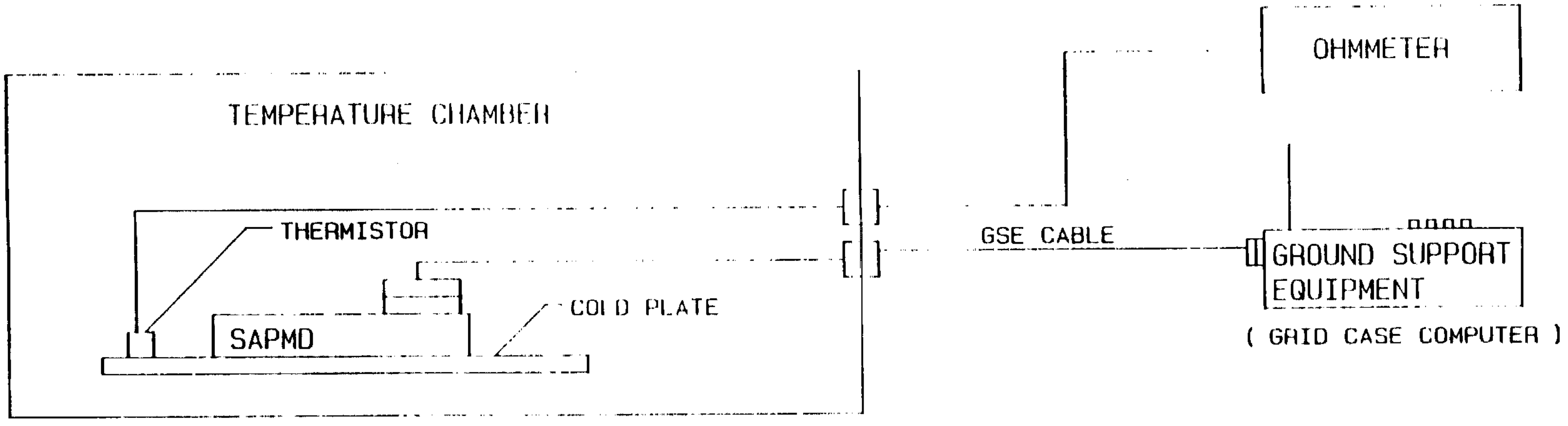

FIGURE 5.1-1 TEMPERATURE TEST CONFIGURATION

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.

____ S) 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.

ITEM MODEL MANUFACTURER S/N CAL DUE

A) ____________________

B) ____________________

D) ____________________

E) ____________________

F) ____________________

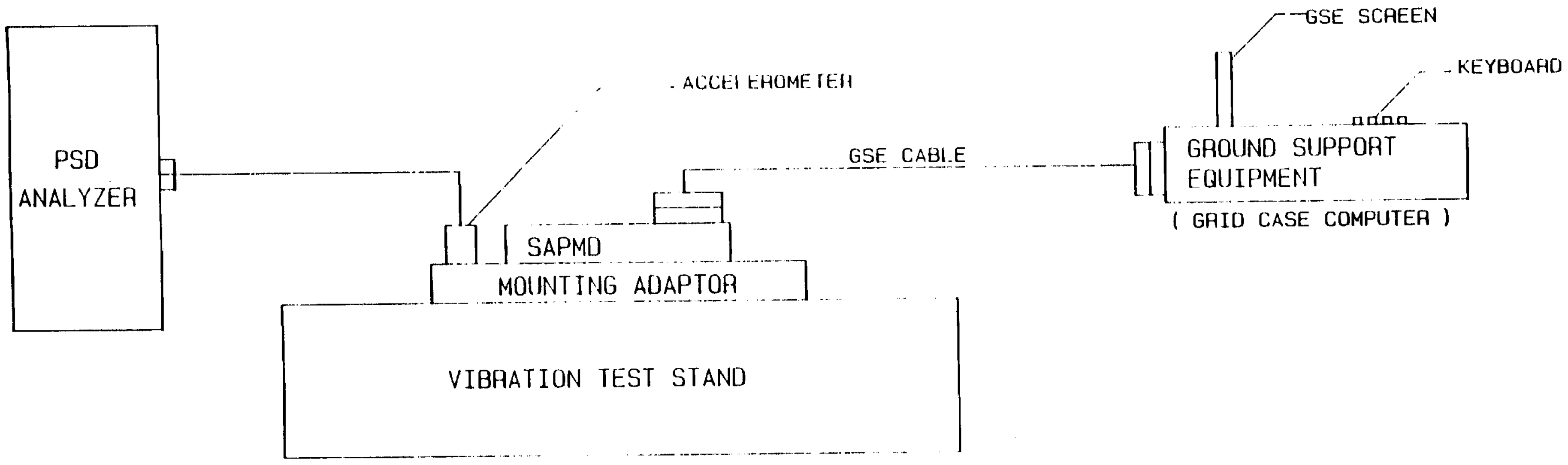

FIGURE 6.1-1 VIBRATION TEST CONFIGURATION, X AND Y AXIS

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.

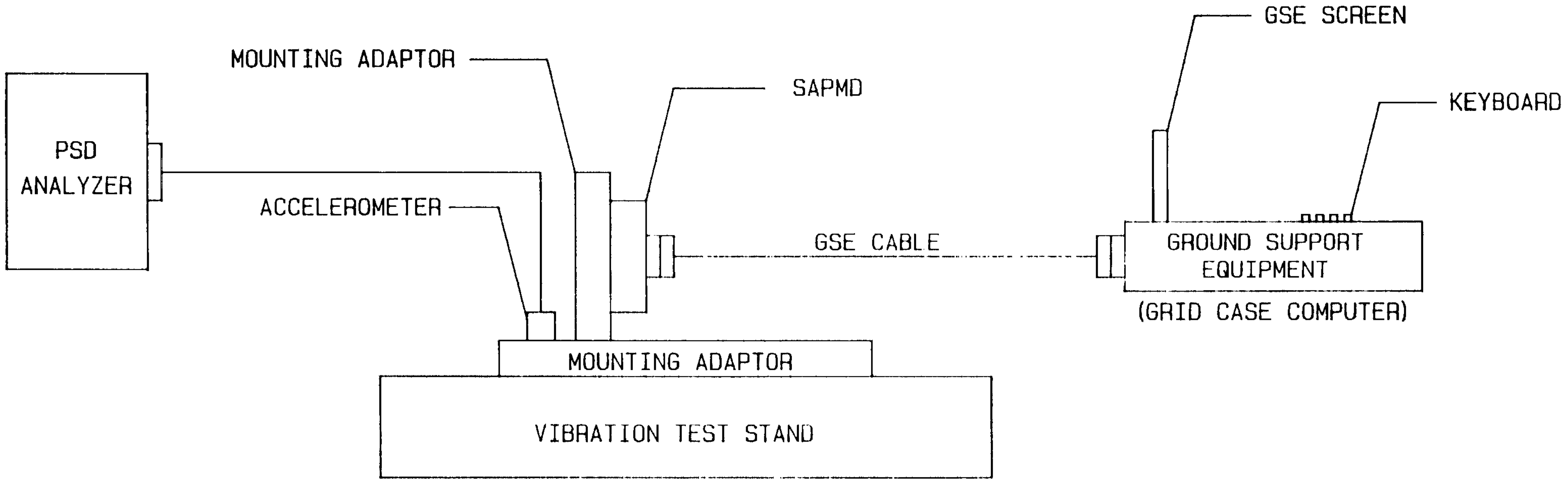

FIGURE 8.1-1 VIBRATION TEST CONFIGURATION, Z AXIS

8.3. Z Axis Measurement Equipment

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

	ITEM	MODEL	MANUFACTURER	S/N	CAL DUE
A)					
B)					
D)					
E)					
F)					

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 it 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.

ITEM MODEL MANUFACTURER S/N CAL DUE

A) ______________________________

B) ______________________________

D) ______________________________

E) ______________________________

F) ______________________________

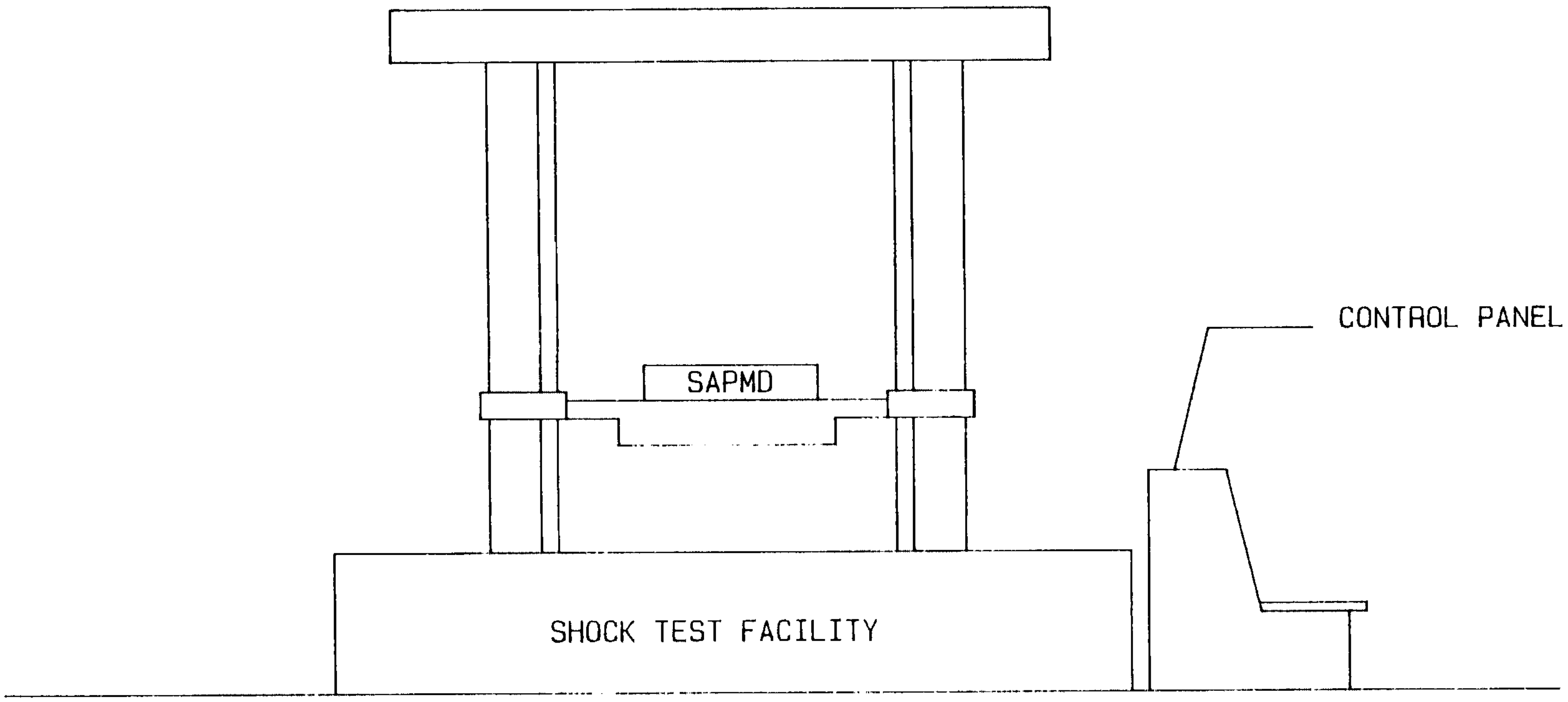

FIGURE 9.1-1 SHOCK TEST CONFIGURATION

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.

ITEM MODEL MANUFACTURER S/N CAL DUE

A) ____

B) ____

D) ____

E) ____

F) ____

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.

____ D) Following the instructions in par. 5.2 of the SGA Software User's Guide, set the GMT value in the SAPMD to local time.

____ E) 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.

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

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

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

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

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

____ K) Leave the SAPMDs in the temperature chamber for 48 hours.

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

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

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

____ O) 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.

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

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

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

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

____ T) 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.