UNITED STATES DEPARTMENT OF INTERIOR

GEOLOGICAL SURVEY

SOFTWARE LISTING FOR THE SETUP AND CALIBRATION OF THE EG&G MODEL 630 VECTOR MEASURING CURRENT METER

by

Gregory K. Miller

Open-File Report 89 - 533

Woods Hole, Mass.

1989

TABLE OF CONTENTS

INTRODUCTION

This software program is designed to provide a means of preparing and testing the EG&G Ocean Products,Inc. model 630 vector measuring-current meter (VMCM) for field use. This program, written in C, can be run on any IBM personal computer or clone and is patterned loosely after the program (offered by EG&G) that runs on a Hewlett Packard HP-85 computer. This U.S. Geological Survey program, however, does not require the user to know the VMCM sensor configuration or the data format to test the system. Also, the data are presented in both raw hexadecimal form and converted to the final value.

Three tests are provided for the user to evaluate the VMCM's performance. The first test (page 4) is a system check-out that can be used to check the VMCM prior to deployment. This test checks the SAIL interface (the communications link between the computer and the VMCM), the VMCM's memory, and board current drains, and provides tests for each of the sensors selected for recording. The second test (page 16) provides a means of calibrating each of the VMCM's sensors. The program simply acquires the data from the VMCM and displays the raw data as a hexadecimal value and its converted value. The third test (page 20) emulates a "dumb" terminal that is connected to the SAIL interface. This permits the user to evaluate the VMCM's performance using a standard command set built into the VMCM's operating system.

Display functions used in this software are part of a commercially available package called C Tools Plus by Blaise Computing, Berkley, California.

```
/*                                DIAG.H                                  */

/* rev 1.01 4/20/89 by gkm */

/*----------------------- MESSAGE DEFINES -----------------*/

static char *opt[] = {
" HEX VALUE     RECORD NUMBER     SEC. FROM STARTUP       <F1 - EXIT>",
" HEX VALUE   NORTH VECTOR COUNT        CM/SEC            <F1 - EXIT>",
" HEX VALUE    EAST VECTOR COUNT        CM/SEC            <F1 - EXIT>",
" HEX VALUE      ROTOR2 COUNT           CM/SEC            <F1 - EXIT>",
" HEX VALUE      ROTOR1 COUNT           CM/SEC            <F1 - EXIT>",
" HEX VALUE        COMPASS DEGREES                       <F1 - EXIT>",
" HEX VALUE     TEMPERATURE VOLTAGE   DEGREES(C)         <F1 - EXIT>",
" HEX VALUE      PRESSURE VOLTAGE                       <F1 - EXIT>",
" HEX VALUE          VOLTAGE                            <F1 - EXIT>",
" HEX VALUE      ROTOR CURRENT(microamps)               <F1 - EXIT>",
" HEX VALUE     MEMORY CURRENT(microamps)               <F1 - EXIT>",
" HEX VALUE     TRANSPORT CURRENT(microamps)            <F1 - EXIT>",
" HEX VALUE     PROCESSOR CURRENT(microamps)            <F1 - EXIT>",
" HEX VALUE     SPARE CURRENT(microamps)                <F1 - EXIT>",
" HEX VALUE     COMPASS CURRENT(microamps)              <F1 - EXIT>",
" HEX VALUE     SERIAL CURRENT(microamaps)              <F1 - EXIT>",
" HEX VALUE     A/D CURRENT(microamps)                  <F1 - EXIT>",
" HEX VALUE     ^LCC                                    <F1 - EXIT>"
};

/* -------------------- KEYBOARD DEFINES ---------------- */

#define F1 0x3B00
#define F2 0x3C00
#define F3 0x3D00
#define F4 0x3E00
#define F5 0x3F00
#define F6 0x4000
#define F7 0x4100
#define F8 0x4200
#define F9 0x4300
#define F10 0x4400


#define  BLANK    0x20
#define  LINEFEED 0x0A
#define  ENTER    0x0D
#define  BACKSP    0x08
#define  ESCAPE    0x1B
#define  BAR      205
#define  NULBYTE '\0'
#define  NORMAL     7
```

```
/* ------------------- SAMPLE TIMES ------------------- */

static char *sam_tim[] = {
" 2 SECONDS",
" 4 SECONDS",
" 7.5 SECONDS",
" 15 SECONDS",
" 30 SECONDS",
" 56.25 SECONDS",
" 1 MINUTE",
" 1.875 MINUTES",
" 2 MINUTES",
" 3.75 MINUTES",
" 4 MINUTES",
" 7.5 MINUTES",
" 15 MINUTES",
" 30 MINUTES",
" 60 MINUTES",
" 2 HOURS"};

/* -------------------- ROM DEFINES -------------------- */

static char *rom_def[15] = {
        "10B1", "C0000",
        "3100", "C0200",
        "060F", "C4400",
        "2710", "C4600",
        "264E", "C4800",
        "1C8D", "C4A00",
        "251C", "C4C00"
        };

/* ------------------ CURRENT LIMITS ------------------ */

static int cur_lim[] = {999,1700,999,5000,25000,999,999,12000};

static int config,sampling,fore,back,col,row,attr;

/*---------------------- FUNCTION DECLARATIONS -------------*/

extern void main(void);

                        /* VMCM TESTS */

extern void check_tst(void);
extern void ram_tst(void);
extern void rom_tst(void);
extern void ad_tst(int);
extern void sen_tst(int);
extern void rotor_tst(int);
extern void cal_tst(void);
extern void cal_proc(int,int,int);
extern void sail_tst(void);
```

```
                        /* SAIL FUNCTIONS */

extern long rotor_ct(int,char *);
extern int diag_mode(int);
extern int sail_cmd(void);

                        /* COM PORT FUNCTIONS */

extern int port_init(void);
extern int data_get(void);
extern void out_char(char);

                        /* SCREEN FUNCTIONS */

extern int main_menu_wn(void);
extern void scrn_setup(char *);
extern void test_box(int,int,int,int,char *);

                        /* DATA CONVERSION FUNCTIONS */

extern double cur_dsp(int,int,char *);
extern long cnvrt_24(unsigned char *);
extern double cnvrt_4(unsigned char *);
extern float decode_ch(double,int,int);
```

```
/*                                DIAG.C                                */

/* REV 1.01 4/20/89 by gkm */

#include <stdio.h>
#include <math.h>
#include <time.h>
#include <bstring.h>
#include <bscreen.h>
#include <bwindow.h>
#include <bkeybd.h>
#include <bisr.h>
#include <asynch_1.h>
#include "diag.h"
#include "gen.h"

unsigned char *buffer,gen_msg[80],cmd_buff[15],data_buff[100];
void main()
{
    int command,ercode,lstatus,mstatus;

    /* --- SET SCREEN INFO --- */

    scrn_ck();

    /* --- ALLOCATE COM PORT BUFFER SPACE --- */

    if((buffer = malloc(512)) == NULL)
        {
        err_wn("no memory available");
        exit(-1);
        }

    /* --- INITIALIZE COM1 AS THE SAIL PORT --- */

    init_a1(COM1,2,0,1,3,&lstatus,&mstatus);
    if(ercode = open_a1(COM1,412,84,0,0,buffer) != 0)
        {
        com_err_wn("Could not open port.",ercode,0);
        exit(-1);
        }
    if((ercode = setop_a1(COM1,9,0)) != 0)
        {
        com_err_wn("Could not set port options",ercode,0);
        exit(-1);
        }
```

```
    /* --- LOOP FOR COMMAND --- */

    for(;;)
        {

        /* --- DISPLAY PROGRAM TITLE --- */

        signon_box("EG&G VMCM DIAGNOSTIC\0",
            "USGS SEDIMENT TRANSPORT PROGRAM\0",
            "REV 1.01 4/20/89\0");

        /* --- EXECUTE THE TEST --- */

        command = main_menu_wn();
        switch(command)
            {
            case F1:
                title_box(" VMCM CHECKOUT\0");
                check_tst();
                break;
            case F2:
                title_box("VMCM CALIBRATION MODE\0");
                cal_tst();
                break;
            case F3:
                title_box("VMCM SAIL INTERFACE\0");
                sail_tst();
                break;
            case F4:
                close_a1(COM1);
                scnewdev(3,25);
                exit(0);
            default:
                break;
            }
        }
}
```

```
/* ------------- CHECKOUT TEST ----------------------- */

void check_tst()
{
    char *tmp;
    int ercode,tries;
    int year,month,day,sec,baud,addr;
    char date_str[31];
    DOSREG regs;

    /* --------------- DISPLAY TEST LABEL -------------- */

    test_box(4,0,8,39,"");
    stsdate(date_str,&year,&month,&day);
    scdspmsg(4,20 - (strlen(date_str)/2),fore,back,date_str);
    ststime(date_str,&year,&month,&day,&sec);
    scdspmsg(5,20 - (strlen(date_str)/2),fore,back,date_str);
    tmp = reply_wn("Enter your name: \0");
    strcpy(gen_msg,tmp);
    scdspmsg(6,20 - (strlen(gen_msg)/2),fore,back,gen_msg);
    tmp = reply_wn("Enter VMCM serial no.: \0");
    strcpy(gen_msg,tmp);
    scdspmsg(7,20 - (strlen(gen_msg)/2),fore,back,gen_msg);

    /* ------------ TEST SAIL COMMUNICATIONS ------------ */

    test_box(4,40,8,79," SAIL TEST \0");
    if(port_init() !=0)
        return;
    retop_a1(COM1,1,&baud);
    (baud == 2)? (baud = 300):(baud = 110);
    sprintf(gen_msg,"Baud rate = %d",baud);
    scdspmsg(5,42,fore,back,gen_msg);
    addr = get_no_wn(" Enter the sail address for the VMCM: \0",0,99);
    sprintf(gen_msg,"SAIL address = %02d",addr);
    scdspmsg(6,42,fore,back,gen_msg);

    for(tries = 1;tries < 5;tries++)
        {
        sprintf(cmd_buff,"#%02d",addr + 80);
        if((sail_cmd() == 0) && (strlen(data_buff) > 22))
            {
            scdspmsg(7,42,fore,back,"VMCM : \0");
            stcsub(gen_msg,data_buff,5,24,28);
            scdspmsg(7,52,fore,back,gen_msg);
            break;
            }
        }
    if(tries == 5)
        {
        err_wn("VMCM did not respond\0");
        return;
        }
```

```
if(tries > 1)
    {
    sprintf(gen_msg,"WARNING,%d tries before VMCM responce",
        tries);
    if(sys.dev == COLOR)
        fore = RED;
    scdspmsg(8,42,fore,back,gen_msg);
    }
if(diag_mode(addr) != 0)
    return;

/* ------------ GET SAMPLING MODE ------------ */

test_box(9,0,19,33, " SAMPLING CONFIGURATION \0");
scdspmsg(10,2,fore,back,"RECORD : STANDARD RECORD\0");

if((config & 0x80) != 0)
    scdspmsg(11,2,fore,back,"          ROTOR/COMPASS\0");
if((config & 0x20) != 0)
    scdspmsg(12,2,fore,back,"          PRESSURE\0");
if((config & 0x40) != 0)
    scdspmsg(13,2,fore,back,"          TEMPERATURE\0");
if((config & 0x10) != 0)
    scdspmsg(14,2,fore,back,"          VOLTAGE\0");
if((config & 0x08) != 0)
    scdspmsg(15,2,fore,back,"          BOARD CURRENTS\0");
sprintf(gen_msg,"SAMPLE INTERVAL = %4.2f SEC",
    ((float)(config & 0x07) + 1.)*.25);
scdspmsg(16,2,fore,back,gen_msg);
tries = (sampling&240)/16;
sprintf(gen_msg,"TEST INTERVAL   = %s",sam_tim[tries]);
scdspmsg(17,2,fore,back,gen_msg);
tries = (sampling&15);
sprintf(gen_msg,"RECORD INTERVAL = %s",sam_tim[tries]);
scdspmsg(18,2,fore,back,gen_msg);
if(get_yn_wn(22,"Are these settings correct(Y/N)? \0") == 'N')
    {
    scdspmsg(19,5,fore,back," CHECK SWITCH SETTINGS \0");
    hit_cont_wn();
    return;
    }
ram_tst();
rom_tst();
if((config & 0x08) != 0)
    {
    ad_tst(addr);
    out_char(BACKSP);
    }
sen_tst(addr);
out_char(BACKSP);

rotor_tst(addr);
out_char(BACKSP);
```

```
    /* --- PRINT THE TEST RESULTS --- */

    if(get_yn_wn(22,"Print results on the Printer?(Y/N) \0") == 'Y')
        {
        pause("MAKE SURE PRINTER IS ON AND READY\0");
        isinvint(5,&regs);
        }
    if((ercode = break_a1(COM1)) != 0)
        com_err_wn("Could not send break command\0", ercode,0);
    hit_cont_wn();
}

    /* --------------- RAM TEST ---------------------- */

void ram_tst()
{
    int tries;
    unsigned x;

    test_box(9,34,12,48," RAM TEST \0");
    scdspmsg(10,35,fore,back,"testing..\0");
    tries = 1;
    for(x = 0x4000;x < 0xFD90;x = x + 0x80)
        {
        sprintf(cmd_buff,"v%04x%c",x,0x0D);
        if(sail_cmd() != 0)
            {
            err_wn("No response for RAM test\0");
            return;
            }
        else if(stschind('0',data_buff) != -1)
            {
            sprintf(gen_msg,"PASS #%d",tries);
            scdspmsg(11,38,fore,back,gen_msg);
            }
        else
            {
            sprintf(gen_msg,"FAIL #%d",tries);
            scdspmsg(11,38,fore,back,gen_msg);
            break;
            }
        if(x == 0x4380)
            x = 0xFD00;
        tries++;
        }
    if(tries >= 9)
        scdspmsg(12,36,fore,back," RAM IS OK \0");
    else
        scdspmsg(12,36,fore,back," RAM IS BAD \0");
}
```

```
/* --------------- ROM TEST ---------------------- */

void rom_tst()
{
    int i,tries;

    test_box(13,34,16,48," ROM TEST \0");
    scdspmsg(14,35,fore,back,"testing..\0");
    i = 1;
    for(tries = 1;tries < 13;tries++)
        {
        sprintf(cmd_buff,"%s%c",rom_def[tries],0x0D);
        sail_cmd();
        stcsub(gen_msg,data_buff,8,4,6);

        if(strcmp(gen_msg,rom_def[tries - 1]) == 0)
            {
            sprintf(gen_msg,"PASS #%d",i);
            scdspmsg(15,38,fore,back,gen_msg);
            }
        else
            {
            sprintf(gen_msg,"FAIL#%d",i);
            scdspmsg(15,38,fore,back,gen_msg);
            break;
            }
        tries++;
        i++;
        }
    if(tries >= 13)
        scdspmsg(16,36,fore,back," ROM IS OK \0");
    else
        scdspmsg(16,36,fore,back," ROM IS BAD \0");
}
```

```
/* ------------ AD BOARD TEST ---------------------- */

void ad_tst(addr)
int addr;
{
    int written,offset;
    BWINDOW *pwin;

    test_box(9,49,20,79," AD TEST \0");
    offset = 24;
    if((config & 0x40) != 0)
        offset = offset +4;
    if((config & 0x20) != 0)
        offset = offset + 4;

    sprintf(cmd_buff,"RN%c",0x0D);
    if(sail_cmd() != 0)
        {
        err_wn("AD test failure\0");
        return;
        }
    pwin = msg_wn_on(" COLLECTING DATA... \0");
    sprintf(cmd_buff,"#%02dr",addr);
    wrtst_a1(COM1,strlen(cmd_buff),cmd_buff,&written);
    do
        {
        if(data_get() == 0)
            {
            msg_wn_off(pwin);
            return;
            }
        }
    while(data_buff[0] != 70);
    msg_wn_off(pwin);
```

```
    /* --- DISPLAY AND CHECK CURRENTS --- */

    row = 10;
    col = 51;
    cur_dsp(offset,9,"VOLTAGE =\0");
    scdspmsg(11,51,fore,back,"Current in microamps\0");
    row = 12;
    if(cur_dsp(offset + 4,10,"ROTOR     =\0") > 999.)
        scdspmsg(12,72,fore,back,"ERROR\0");
    row++;
    if(cur_dsp(offset + 8,11,"MEMORY    =\0") > 1700.)
        scdspmsg(13,72,fore,back,"ERROR\0");
    row++;
    if(cur_dsp(offset + 12,12,"XPORT     =\0") > 999.)
        scdspmsg(14,72,fore,back,"ERROR\0");
    row++;
    if(cur_dsp(offset + 16,13,"PROCESSOR =\0") > 5000.)
        scdspmsg(15,72,fore,back,"ERROR\0");
    row++;
    if(cur_dsp(offset + 20,14,"SPARE     =\0") > 25000.)
        scdspmsg(16,72,fore,back,"ERROR\0");
    row++;
    if(cur_dsp(offset + 24,15,"COMPASS   =\0") > 999.)
        scdspmsg(17,72,fore,back,"ERROR\0");
    row++;
    if(cur_dsp(offset + 28,16,"SERIAL    =\0") > 999.)
        scdspmsg(18,72,fore,back,"ERROR\0");
    row++;
    if(cur_dsp(offset + 32,17,"AD        =\0") > 12000.)
        scdspmsg(19,72,fore,back,"ERROR\0");

    return;
}
```

```
/* ---------------- TEST THE SENSORS ------------------ */

void sen_tst(addr)
int addr;
{
     int written,t_range,p_range,offset;
     double ad_val,value;
     char hex_dsp[5];

     /* --- TEST THE TEMPERATURE --- */

     sprintf(cmd_buff,"#%02dr",addr);
     wrtst_a1(COM1,strlen(cmd_buff),cmd_buff,&written);
     test_box(21,49,23,79," TEMPERATURE TEST \0");
     if((config & 0x40) != 0)
          {
          if(get_yn_wn(22,"Test the temperature (Y/N)? \0") == 'Y')
               {
               t_range= get_no_wn("Which temperature range(1 to 4):
                    \0",1,4);
               offset = 14;
               if((config & 0x80) != 0)
                    offset = 24;
               scdspmsg(23,54,fore,back," HIT F1 TO HALT TEST \0");
               while(data_get() != 0)
                    {
                    if(data_buff[0] == 70)
                         {
                         stcsub(hex_dsp,data_buff,offset,4,5);
                         ad_val = cnvrt_4(hex_dsp);
                         value = decode_ch(ad_val,7,t_range);
                         scclrmsg(22,50,28);
                         sprintf(gen_msg,"%6.3f DEG. C",value);
                         scdspmsg(22,54,fore,back,gen_msg);
                         }
                    }
               }
          else
               scdspmsg(22,51,fore,back,"TEMPERATURE NOT TESTED\0");
          }
     else
          scdspmsg(22,51,fore,back,"TEMPERATURE NOT TESTED\0");
```

```
/* --- TEST THE PRESSURE --- */

p_range = 0;
test_box(17,34,19,48," PRESSURE \0");
if((config & 0x20) != 0)
    {
    if(get_yn_wn(22,"Test the Pressure (Y/N)? \0") == 'Y')
        {
        p_range = get_no_wn("Enter pressure range: \0",1,999);
        offset = 14;
        if((config & 0x80) != 0)
            offset = offset + 10;
        if((config & 0x40) != 0)
            offset = offset + 4;
        scdspmsg(19,36,fore,back," F1 TO HALT \0");
        while(data_get() != 0)
            {
            if(data_buff[0] == 70)
                {
                stcsub(hex_dsp,data_buff,offset,4,5);
                ad_val = cnvrt_4(hex_dsp);
                value = decode_ch(ad_val,8,p_range);
                scclrmsg(18,35,12);
                sprintf(gen_msg,"%12.3f",value);
                scdspmsg(18,35,fore,back,gen_msg);
                }
            }
        }
        else
            scdspmsg(18,36,fore,back,"NOT TESTED\0");
    }
else
    scdspmsg(18,36,fore,back,"NOT TESTED\0");
```

```
    /* --- TEST THE COMPASS --- */

    test_box(20,34,23,48," COMPASS \0");
    if((config & 0x80) != 0)
        {
        if(get_yn_wn(22,"Test the compass (Y/N)? \0") == 'Y')
            {
            offset = 22;
            scdspmsg(23,36,fore,back," F1 TO HALT \0");
            while(data_get() != 0)
                {
                if(data_buff[0] == 70)
                    {
                    stcsub(hex_dsp,data_buff,offset,2,3);
                    ad_val = cnvrt_4(hex_dsp);
                    value = decode_ch(ad_val,6,1);
                    scclrmsg(21,35,12);
                    sprintf(gen_msg,"%5.2f DEG.",value);
                    scdspmsg(21,37,fore,back,gen_msg);
                    }
                }
            }
        else
            scdspmsg(21,36,fore,back,"NOT TESTED\0");
        }
    else
        scdspmsg(21,36,fore,back,"NOT TESTED\0");

}
```

```
/* ---------------- ROTOR TEST ------------------------ */

void rotor_tst(addr)
int addr;
{
    int i,written;
    char rotor1_hex[5],rotor2_hex[5];
    long rotor1,rotor2;
    BWINDOW *pwin;

    test_box(20,0,24,33," ROTOR TEST \0");
    scdspmsg(21,6,fore,back,"ZERO      +10       -10\0");
    if(get_yn_wn(22,"Test the rotors (Y/N)? \0") == 'N')
        {
        scdspmsg(22,8,fore,back,"ROTORS NOT TESTED\0");
        return;
        }
    do
        {
        rotor1 = 0;
        rotor2 = 0;
        pause("STOP BOTH ROTORS FROM ROTATING\0");
        scclrmsg(22,1,31);
        scclrmsg(23,1,31);
        sprintf(cmd_buff,"#%02dr",addr);
        wrtst_a1(COM1,strlen(cmd_buff),cmd_buff,&written);
        pwin = msg_wn_on(" COLLECTING DATA... \0");
        for(i = 0;i < 10;i++)
            {
            if((data_get() != 0) && (data_buff[0] == 70))
                {
                stcsub(rotor1_hex,data_buff,18,4,5);
                rotor1 = cnvrt_24(rotor1_hex) + rotor1;
                stcsub(rotor2_hex,data_buff,14,4,5);
                rotor2 = cnvrt_24(rotor2_hex) + rotor2;
                }
            }
        msg_wn_off(pwin);
        (rotor1 == 0)? (sprintf(gen_msg,"#1 %d PASS",rotor1)):
            sprintf(gen_msg,"#1 %d FAIL",rotor1);
        scdspmsg(22,2,fore,back,gen_msg);
        (rotor2 == 0)? sprintf(gen_msg,"#2 %d PASS",rotor2):
            sprintf(gen_msg,"#2 %d FAIL",rotor2);
        scdspmsg(23,2,fore,back,gen_msg);

        /* --- BOTTOM ROTOR +10 TURNS --- */

        rotor1 = rotor_ct(18,"ROTATE BOTTOM ROTOR +10 TURNS\0");
        rotor1 = rotor1/4;
        (rotor1 == 10)? (sprintf(gen_msg,"%d PASS",rotor1)):
            sprintf(gen_msg,"%d FAIL",rotor1);
        scdspmsg(22,14,fore,back,gen_msg);
```

```
/* ------------- CALIBRATION TEST ---------------------- */

void cal_tst()
{
    int high,low,valid,scale,offset,channel,addr,written;

    if(port_init() != 0)
        return;
    addr = get_no_wn(" Enter the sail address for the VMCM: \0",
          0,99);

    if(diag_mode(addr) != 0)
        {
        err_wn("Could not get instrument parameters\0");
        return;
        }

    /* --- SETUP THE SCREEN --- */

    scrn_setup("CALIBRATION TEST\0");
    sccurst(&row,&col,&high,&low);

    puts(" 1 = RECORD COUNT\t  2 = NORTH VECTOR\t  3 = EAST VECTOR");
    puts(" 4 = ROTOR2\t\t  5 = ROTOR1\t\t  6 = COMPASS");
    puts(" 7 = TEMPERATURE\t  8 = PRESSURE\t\t  9 = VOLTAGE");
    puts("10 = ROTOR CURRENT\t 11 = MEMORY CURRENT\t
          12 = TRANSPORT CURRENT");
    puts("13 = PROCESSOR CURRENT\t 14 = SPARE CURRENT\t
          15 = COMPASS CURRENT");
    puts("16 = SERIAL CURRENT\t 17 = A/D CURRENT\t 18 = ^LCC");

    do
        {
        valid = TRUE;
        channel = get_no_wn(" Enter variable to view ( 1 to 18 ):
                \0",1,18);
        switch(channel)
            {
            case 1:
            case 2:
            case 3:
                scale = get_no_wn(" Enter the sample interval in
                    seconds: ",0,7200);
                offset = channel*4 - 2;
                break;
            case 4:
            case 5:
            case 6:
                scale = get_no_wn(" Enter the sample interval in
                    seconds: ",0,7200);
                if((config & 0x80) != 0)
                    offset = channel*4 - 2;
                else
                    valid = FALSE;
                break;
```

```
        /* --- BOTTOM ROTOR -10 TURNS --- */

        rotor1 = rotor_ct(18,"ROTATE BOTTOM ROTOR -10 TURNS\0");
        rotor1 = rotor1/4;
        (rotor1 == -10)? (sprintf(gen_msg,"%d PASS",rotor1)):
            sprintf(gen_msg,"%d FAIL",rotor1);
        scdspmsg(22,24,fore,back,gen_msg);

        /* --- TOP ROTOR +10 TURNS --- */

        rotor2 = rotor_ct(14,"ROTATE TOP ROTOR +10 TURNS\0");
        rotor2 = rotor2/4;
        (rotor2 == 10)? sprintf(gen_msg,"%d PASS",rotor2):
            sprintf(gen_msg,"%d FAIL",rotor2);
        scdspmsg(23,14,fore,back,gen_msg);

        /* --- TOP ROTOR -10 TURNS --- */

        rotor2 = rotor_ct(14,"ROTATE TOP ROTOR -10 TURNS\0");
        out_char(BACKSP);
        rotor2 = rotor2/4;
        (rotor2 == -10)? sprintf(gen_msg,"%d PASS",rotor2):
            sprintf(gen_msg,"%d FAIL",rotor2);
        scdspmsg(23,24,fore,back,gen_msg);
        }
    while(get_yn_wn(3,"REPEAT TEST?(Y/N) \0") == 'Y');
}
```

```
case 7:
    if((config & 0x40) == 0)
        valid = FALSE;
    else
        {
        scale = get_no_wn(" Enter temperature
             range(1 - 4): \0",1,4);
        offset = 14;
        if((config & 0x80) != 0)
            offset = 24;
        }
    break;
case 8:
    if((config & 0x20) == 0)
        valid = FALSE;
    else
        {
        scale = get_no_wn(" Enter pressure range: \0",
             0,99);
        offset = 14;
        if((config & 0x80) != 0)
            offset = offset + 10;
        if((config & 0x40) != 0)
            offset = offset + 4;
        }
    break;
case 9:
    if((config & 0x10) == 0)
        valid = FALSE;
    else
        {
        scale = 1;
        offset = 14;
        if((config & 0x80) != 0)
            offset = offset + 10;
        if((config & 0x40) != 0)
            offset = offset + 4;
        if((config & 0x20) != 0)
            offset = offset + 4;
        }
    break;
```

```
        default:
            if((config & 0x08) == 0)
                valid = FALSE;
            else
                {
                scale = 1;
                offset = 14 + (channel - 10)*4;
                if((config & 0x80) != 0)
                    offset = offset + 10;
                if((config & 0x40) != 0)
                    offset = offset + 4;
                if((config & 0x20) != 0)
                    offset = offset + 4;
                if((config & 0x10) != 0)
                    offset = offset + 4;
                }
            break;
        }
    if(valid == FALSE)
        err_wn("CHANNEL NOT AVAILABLE");
    }
while(valid == FALSE);
scrn_setup(opt[channel -1]);
```

```
    /* --- START ACQUIRING DATA FROM THE VMCM --- */

    sprintf(cmd_buff,"#%02dr",addr);
    wrtst_a1(COM1,strlen(cmd_buff),cmd_buff,&written);

    /* --- LOOP UNTIL F1 KEY IS HIT --- */

    while(data_get() != 0)
        {
        if(data_buff[0] == 70)
            {
            cal_proc(channel,offset,scale);
            (row < 20)? (row++) : (scscroll(1,NORMAL,7,0,20,
                    79,SCR_UP));
            col = 0;
            sccurset(row,col);
            }
        }
    out_char(BACKSP);
}

/* ----------- PROCESS A CHANNEL OF DATA ------------- */

void cal_proc(channel,offset,scale)
int channel,offset,scale;
{
    char hex_dsp[5];
    double ad_val,value;
    long rotor;

    switch(channel)
        {
        case 1:
            stcsub(hex_dsp,data_buff,offset,4,5);
            scdspmsg(row,col+9,-1,-1,hex_dsp);
            ad_val = cnvrt_4(hex_dsp);
            sprintf(gen_msg,"%10.f",ad_val);
            scdspmsg(row,col+19,-1,-1,gen_msg);
            value = decode_ch(ad_val,channel,scale);
            sprintf(gen_msg,"%12.3f",value);
            scdspmsg(row,col+37,-1,-1,gen_msg);
            break;
        case 2:
        case 3:
        case 4:
        case 5:
            stcsub(hex_dsp,data_buff,offset,4,5);
            scdspmsg(row,col+9,-1,-1,hex_dsp);
            rotor = cnvrt_24(hex_dsp);
            sprintf(gen_msg,"%10d",rotor);
            scdspmsg(row,col+19,-1,-1,gen_msg);
            value = decode_ch(rotor,channel,1);
            sprintf(gen_msg,"%12.3f",value);
            scdspmsg(row,col+37,-1,-1,gen_msg);
            break;
```

```
        case 6:
            stcsub(hex_dsp,data_buff,offset,2,3);
            scdspmsg(row,col+9,-1,-1,hex_dsp);
            ad_val = cnvrt_4(hex_dsp);
            value = decode_ch(ad_val,channel,scale);
            sprintf(gen_msg,"%12.3f",value);
            scdspmsg(row,col+22,-1,-1,gen_msg);
            break;
        case 7:
        case 8:
            stcsub(hex_dsp,data_buff,offset,4,5);
            scdspmsg(row,col+9,-1,-1,hex_dsp);
            ad_val = cnvrt_4(hex_dsp);
            if((unsigned long)ad_val > 4095)
                value = 10*(ad_val - 32768.0)/4096;
            else
                value = -10*(ad_val)/4096;
            sprintf(gen_msg,"%10.3f",value);
            scdspmsg(row,col+22,-1,-1,gen_msg);
            value = decode_ch(ad_val,channel,scale);
            sprintf(gen_msg,"%12.3f",value);
            scdspmsg(row,col+40,-1,-1,gen_msg);
            break;
        case 9:
        case 10:
        case 11:
        case 12:
        case 13:
        case 14:
        case 15:
        case 16:
        case 17:
            stcsub(hex_dsp,data_buff,offset,4,5);
            scdspmsg(row,col+9,-1,-1,hex_dsp);
            ad_val = cnvrt_4(hex_dsp);
            value = decode_ch(ad_val,channel,scale);
            sprintf(gen_msg,"%12.3f",value);
            scdspmsg(row,col+20,-1,-1,gen_msg);
            break;
        case 18:
            stcsub(hex_dsp,data_buff,offset,2,3);
            scdspmsg(row,col+9,-1,-1,hex_dsp);
            break;
        default:
            break;
        }
}
```

```
/* ------------- SAIL INTERFACE FUNCTION ---------------- */

void sail_tst()
{
    int ch,high,low;
    unsigned char rd_ch;
    int ercode,iqsize;
    unsigned status;

    if(port_init() != 0)
        return;

    /* --- SETUP THE SCREEN --- */

    scrn_setup("<F1 - EXIT>\0");
    sccurst(&row,&col,&high,&low);

    /* --- LOOP UNTIL F1 KEY IS HIT --- */
for(;;)
    {
    if (kbready(&rd_ch,&ch) != 0)
        {
        if(ch == 59)
            {
            kbflush();
            return;
            }
        switch(rd_ch)
            {
            case BACKSP:
                out_char(BACKSP);
                iflsh_a1(COM1);
                break;
            case ESCAPE:
                if((ercode = break_a1(COM1)) != 0)
                    com_err_wn("Could not send break command",
                        ercode,0);
                break;
            default:
                rd_ch = getch();
                    out_char(rd_ch);
                break;
            }
        kbflush();
        }
    else
        {
        if(rdch_a1(COM1,&rd_ch,&iqsize,&status) == 0)
            {
            switch(rd_ch)
                {
                case ENTER:
                    if (row == 20)
                        scscroll(1,NORMAL,7,0,20,79,SCR_UP);
                    else
```

```
                    row++;
                col = 0;
                sccurset(row,col);
                 break;
            case BACKSP:
                if (col > 1)
                    {
                    col--;
                    sccurset(row,col);
                    }
                scwrite(BLANK,1);
                sccurset(row,col);
                break;
             case ESCAPE:
                scwrite(BLANK,77);
                    if (col < 79)
                       col++;
                    else
                        {
                    if (row < 20)
                        row++;
                    else
                        scscroll(1,NORMAL,7,0,20,79,SCR_UP);
                    col = 0;
                    sccurset(row,col);
                        }
            case LINEFEED:
                break;
            default:
                if(rd_ch > 31 && rd_ch < 256)
                    scwrite(rd_ch,1);
                    if (col < 79)
                    col++;
                else
                    {
                    if (row < 20)
                        row++;
                    else
                        scscroll(1,NORMAL,7,0,20,79,SCR_UP);
                    col = 0;
                    }
                sccurset(row,col);
                break;
            }
        }
    }
}
```

```
/******************** SAIL FUNCTIONS ********************/

/* ------------ GET ROTOR COUNTS ------------------- */

long rotor_ct(offset,label)
int offset;
char *label;
{
     int ch;
     unsigned char rd_ch;
     char hex_buff[5];
     long count;
     BWINDOW *pwin;

     count = 0;
          pwin = start_wn_on(label);
          while(kbready(&rd_ch,&ch) == 0)
               {
               if((data_get() != 0) && (data_buff[0] == 70))
                    {
                    stcsub(hex_buff,data_buff,offset,4,5);
                    count = cnvrt_24(hex_buff) + count;
                    }
               }
          msg_wn_off(pwin);
          kbflush();
          pwin = msg_wn_on(" COLLECTING DATA... \0");
          if((data_get() != 0) && (data_buff[0] == 70))
               {
               stcsub(hex_buff,data_buff,offset,4,5);
               count = cnvrt_24(hex_buff) + count;
               }
          msg_wn_off(pwin);
          return(count);
}
```

```
/* --------------- SET DIAGNOSTICS MODE --------------- */

int diag_mode(addr)
int addr;
{
    int i;
    unsigned char hex_str[3];

    sprintf(cmd_buff,"#%02dD",addr);
    if((sail_cmd() != 0) && (stschind('*',data_buff) == -1))
        {
        err_wn("No response from VMCM diagnostics\0");
        return(-1);
        }
    stcsub(hex_str,data_buff,1,2,3);

    /* --- GET SAMPLING MODES --- */

    sprintf(cmd_buff,"s%4x ",0xFFE0);
    if(sail_cmd() != 0)
        {
        err_wn("Unable to get sampling times\0");
        return(-1);
        }

    stcsub(hex_str,data_buff,6,2,3);
    for(i = 0; i < 2;i++)
        {
        if(hex_str[i] > 57)
            hex_str[i]  = hex_str[i] - 55;
        else
            hex_str[i] = hex_str[i] - 48;
        }
    sampling = hex_str[0]*16 + hex_str[1];
    sprintf(cmd_buff," ");
    if(sail_cmd() != 0)
        {
        err_wn("Unable to get configuration\0");
        return(-1);
        }
    stcsub(hex_str,data_buff,1,2,3);
    for(i = 0; i < 2;i++)
        {
        if(hex_str[i] > 57)
            hex_str[i]  = hex_str[i] - 55;
        else
            hex_str[i] = hex_str[i] - 48;
        }
    config = hex_str[0]*16 + hex_str[1];

    sprintf(cmd_buff,"%c",ENTER);
    if((sail_cmd() != 0) && (stschind('*',data_buff) == -1))
        {
        err_wn("No termination from VMCM diagnostics\0");
        return(-1);
```

```
        }
    stcsub(hex_str,data_buff,1,2,3);
    return(0);
}
/* ------------ SEND A SAIL COMMAND ------------------ */

int sail_cmd()
{
    int ercode,written,iq_size;
    unsigned status;
    int i,no_read,no_left,buffsiz;
    long finish,begin;

    iflsh_a1(COM1);
    for(i = 0;i < 78;i++)
        data_buff[i] = '\0';
    time(&begin);
    if((ercode = wrtst_a1(COM1,strlen(cmd_buff),cmd_buff,
          &written)) != 0)
        {
        iflsh_a1(COM1);
        com_err_wn("Could not write sail command",ercode,0);
        return(-1);
        }
    do
        {
        qsize_a1(COM1,&iq_size,status);
        time(&finish);
        if(finish > (begin + 2))
            return(1);
        }
    while(iq_size <= strlen(cmd_buff));

    /*------- LOOP UNTIL ALL DATA IS IN BUFFER ---------------*/

    while ((iq_size < 80)&&(iq_size != buffsiz))
        {
        buffsiz = iq_size;
        for( i = 0;i < 32000;i++)
            i = i;
        if((ercode = qsize_a1(COM1,&iq_size,&status)) != 0)
            {
            com_err_wn("Could not get input buffer size",
               ercode,status);
            return(-1);
            }
        }
    if((ercode=rdst_a1(COM1,iq_size,data_buff,&no_read,&no_left))!=0)
        {
        com_err_wn("Could not read input buffer (rd_st_a1)",
            ercode,0);
        return(-1);
        }
    return(0);
}
```

```
/*************** COM PORT FUNCTIONS **********************/

/* ----------- OPEN THE COM1 SERIAL PORT -------------- */

int port_init()
{
    int baud,ercode;

    while(baud != 300 && baud != 110)
        baud = get_no_wn(" Enter the Baud Rate to use (110 or 300): ",
            110,300);
    if(baud == 110)
        {
        if((ercode = setop_a1(COM1,1,0)) != 0)
            {
            com_err_wn("Could not set baud rate",ercode,0);
            return(-1);
            }
        }
    else
        {
        if((ercode = setop_a1(COM1,1,2)) != 0)
            {
            com_err_wn("Could not set baud rate",ercode,0);
            return(-1);
            }
        }
    return(0);
}
```

```
/* -------------- ACQUIR DATA FROM THE VMCM ---------- */

int data_get()
{
    unsigned status;
    int i,ercode,buffsiz,iq_size,no_read,no_left,ch;
    unsigned char rd_ch;

    do
        {
        if((ercode = qsize_a1(COM1,&iq_size,&status)) != 0)
            {
            com_err_wn("Could not get input buffer size",
               ercode,status);
            return(0);
            }

        if (kbready(&rd_ch,&ch) != 0)
            {
            if(ch == 59)
                {
                kbflush();
                return(0);
                }
            kbflush();
            }
        }
    while(iq_size < 3);

    /*------- LOOP UNTIL ALL DATA IS IN BUFFER ---------------*/

    while ((iq_size < 80)&&(iq_size != buffsiz))
        {
        buffsiz = iq_size;

        /* --- DELAY FOR BAUD RATE --- */

        for( i = 0;i < 32000;i++)
            i = i;
        if((ercode = qsize_a1(COM1,&iq_size,&status)) != 0)
            {
            com_err_wn("Could not get input buffer size",
               ercode,status);
            return(0);
            }
        }
    if((ercode=rdst_a1(COM1,iq_size,data_buff,&no_read,&no_left))!=0)
        {
        com_err_wn("Could not read input buffer (rd_st_a1)",
            ercode,0);
        return(0);
        }
    return(1);
}
```

```
/* ----------- OUTPUT FUNCTION FOR COM 1 --------------- */

void out_char(ch)
char ch;
{
    int ercode;

     if ((ercode = wrtch_a1(COM1,ch)) != 0)
        com_err_wn("Cannot write to the output queue (wrtch_a1).",
            ercode,0);
     return;
}
```

```
/****************** SCREEN FUNCTIONS ***********************/

/*--------- DISPLAY SELECT MESSAGE (WINDOW) --------------*/

int main_menu_wn()
{

    BWINDOW *pwin1,*pwin2,*pwin3,*pwin4;
    BORDER bord;
    int cursor_was_off,high,low;
    int option;
    long start,finish;

    /*------------ DETERMINE MONITOR TYPE -----------------*/

    if(sys.dev == MONO)
        {
        fore = -1;
        back = -1;
        attr = REVERSE;
        }
    else
        {
        fore = BLACK;
        back = RED;
        attr = back*16 + fore;
        }

    /*---------------- CREATE THE WINDOWS ------------------*/

    if((pwin1 = wncreate(2,13,attr)) == NIL)
        window_err_wn("Could not create window #1",0);
    if((pwin2 = wncreate(2,13,attr)) == NIL)
        window_err_wn("Could not create window #2",0);
    if((pwin3 = wncreate(2,13,attr)) == NIL)
        window_err_wn("Could not create window #3",0);
    if((pwin4 = wncreate(2,13,attr)) == NIL)
        window_err_wn("Could not create window #4",0);
    bord.type = 1;
    if(sys.dev == COLOR)
        bord.attr = WHITE;
    else
        bord.attr = BLACK;

    cursor_was_off = sccurst(&row,&col,&high,&low);
    sys.corner.row = 22;

    /* --- DISPLAY SYSTEM TEST WINDOW --- */

    sys.corner.col = 2;
    wndsplay(pwin1,&sys,&bord);
    wncurmov(0,5);
    wnwrap(0,"F1\0",fore,back,CHARS_ONLY);
    wncurmov(1,1);
    wnwrap(0,"SYSTEM TEST",fore,back,CHARS_ONLY);
```

```
/* --- DISPLAY MONITOR WINDOW --- */

sys.corner.col = 22;
wndsplay(pwin2,&sys,&bord);
wncurmov(0,5);
wnwrap(0,"F2\0",fore,back,CHARS_ONLY);
wncurmov(1,2);
wnwrap(0,"CAL MODE\0",fore,back,CHARS_ONLY);

/* --- DISPLAY SYSTEM DIAGNOSTIC WINDOW --- */

sys.corner.col = 42;
wndsplay(pwin3,&sys,&bord);
wncurmov(0,5);
wnwrap(0,"F3\0",fore,back,CHARS_ONLY);
wncurmov(1,1);
wnwrap(0," DIAGNOSTIC\0",fore,back,CHARS_ONLY);

/* --- DISPLAY SYSTEM EXIT WINDOW --- */

sys.corner.col = 62;
wndsplay(pwin4,&sys,&bord);
wncurmov(0,5);
wnwrap(0,"F4\0",fore,back,CHARS_ONLY);
wncurmov(1,3);
wnwrap(0," EXIT\0",fore,back,CHARS_ONLY);
scpgcur(1,0,0,0);

/* --- GET FUNCTION KEY ---- */

do
    {
    option = get_key();
    }
while(option != F1 && option != F2 && option != F3 &&
      option != F4);
switch(option)
    {
    case F1:
        wnselect(pwin1);
        break;
    case F2:
        wnselect(pwin2);
        break;
    case F3:
        wnselect(pwin3);
        break;
    case F4:
        wnselect(pwin4);
        break;
    }
    wnattr(BLACK,WHITE);
```

```
    /* --------------- LOOP FOR A DELAY ------------------- */

        time(&start);
        do
            {
            time(&finish);
            }
        while(finish - start < 1);

    /*------------------ REMOVE THE WINDOW ------------------*/

    msg_wn_off(pwin1);
    msg_wn_off(pwin2);
    msg_wn_off(pwin3);
    msg_wn_off(pwin4);
    sccurset(row,col);
    scpgcur(1,6,7,CUR_ADJUST);

    return(option);
}

/* ----------- SETUP FUNCTION SCREEN ---------------- */

void scrn_setup(msg)
char *msg;
{
    /* --- DISPLAY A MESSAGE BOX --- */

    if(sys.dev == MONO)
        {
        fore = -1;
        back = -1;
        }
    else
        {
        fore = WHITE;
        back = RED;
        }

    scbox(3,0,5,79,0,-1,NORMAL);
    scatrect(3,0,5,79,fore,back);
    scdspmsg(4,40 - strlen(msg)/2,fore,back,msg);

    /* --- CLEAR THE DISPLAY SCREEN AND SET CURSOR --- */

    sccurset(6,0);
    scattrib(BLUE,BLACK,(char)BAR,80);
    scscroll(0,NORMAL,7,0,24,79,SCR_UP);
    row = 7;
    col = 0;
    sccurset(row,col);
    scpgcur(0,6,7,CUR_ADJUST);
}
```

```
/* -------------- DRAW THE TEST BOX ----------------- */

void test_box(x1,y1,x2,y2,label)
int x1,y1,x2,y2;
char * label;
{
    int position;

    if(sys.dev == COLOR)
        {
        fore = BLACK;
        back = CYAN;
        attr = back*16 + fore;
        }
    else
        {
        fore = -1;
        back = -1;
        attr = REVERSE;
        }
    scbox(x1,y1,x2,y2,0,0,attr);
    scatrect(x1,y1,x2,y2,fore,back);
    position = y1 + ((y2 - y1)/2) - (strlen(label)/2);
    scdspmsg(x1,position,fore,back,label);
}
```

```
/***************** DATA CONVERSION FUNCTIONS ***********/

/* -------------- DISPLAY CURRENT VALUES ------------ */

double cur_dsp(offset,chan,msg)
int offset,chan;
char * msg;
{
    char hex_buff[5];
    double ad_val,value;

    stcsub(hex_buff,data_buff,offset,4,5);
    ad_val = cnvrt_4(hex_buff);
    value = decode_ch(ad_val,chan,1);
    sprintf(gen_msg,"%s %8.3f",msg,value);
    scdspmsg(row,col,fore,back,gen_msg);
    return(value);
}


/* --------- CONVERT A FLOATING POINT NUMBER -------- */

long cnvrt_24(buff)
 unsigned char *buff;
{
    int i;
    long num;
    unsigned char val[4];

    val[0] = 48;
    val[1] = 48;
    if(strlen(buff) > 3)
        {
        val[0] = *buff++;
        val[1] = *buff++;
        }
    val[2] = *buff++;
    val[3] = *buff;
    for(i = 0; i < 4;i++)
        {
        if(val[i] > 57)
            val[i]  = val[i] - 55;
        else
            val[i] = val[i] - 48;
        }
    num = (val[0])*256 + (val[1]*16 + val[2]);
    if(num > 2047)
        num = (num - 4096);
    if(val[3] != 15)
        num = num*pow(2,val[3]);
    return(num);
}
```

```
/* -------- CONVERT 4-BYTE ASCII TO WORKING NUMBER ------- */

double cnvrt_4(buff)
unsigned char *buff;
{
    int i;
    double num;
    unsigned char val[4];

    val[0] = 48;
    val[1] = 48;
    if(strlen(buff) > 3)
        {
        val[0] = *buff++;
        val[1] = *buff++;
        }
    val[2] = *buff++;
    val[3] = *buff;
    for(i = 0; i < 4;i++)
        {
        if(val[i] > 57)
            val[i]  = val[i] - 55;
        else
            val[i] = val[i] - 48;
        }
    num = (val[0]*16 + val[1])*256 + (val[2]*16 + val[3]);
    return(num);
}

/* ------------- CALCULATE CHANNEL VALUE ------------- */

float decode_ch(adval,chan,scale)
double adval;
int chan,scale;
{
    float result,y;

    switch(chan)
        {
        case 1:
            result = adval*scale;
            break;
        case 2:
        case 3:
        case 4:
        case 5:
            result = (adval*9.363)/scale;
            break;
        case 6:
            if(adval > 239)
                adval = -(adval - 240);
            result = adval*(180/128);
            break;
```

```
    case 7:
        if((unsigned long)adval > 4095)
            adval = adval - 32768.0;
        else
            adval = -(adval);
        if(scale == 1)
            y = 93.277*(adval + 29292.0)/(29292.0 - adval);
        else if(scale == 2)
            y = 54.437*(adval + 29292.0)/(29292.0 - adval);
        else if(scale == 3)
            y = 31.77*(adval + 29292.0)/(29292.0 - adval);
        else
            y = 54.437*(adval + 9764.0)/(9764.0 - adval);
        result = -273.16+1/(.00248984+.000250157*log(y)+
           .000000336984*pow(log(y),3.0));
        break;
    case 8:
        if((unsigned long)adval > 4095)
            y = (adval - 32768.0)/4096.0;
        else
            y = -(adval/4096.0);
        result = (y*scale);
        break;
    case 9:
        if((unsigned long)adval > 4095)
            y = (adval - 32768.0)/4096.0;
        else
            y = -(adval/4096.0);
        result = -(y*10);
        break;
    case 10:
    case 11:
    case 12:
    case 14:
    case 15:
    case 16:
        if((unsigned long)adval > 4095)
            y = 0.25*((adval - 32768.0)/4096.0);
        else
            y = - 0.25*(adval/4096.0);
        result = (1.0/1000. + 1./100.)*y*1000000.;
        break;
    case 13:
    case 17:
        if((unsigned long)adval > 4095)
            y = 0.25*((adval - 32768.0)/4096.0);
        else
            y = - 0.25*(adval/4096.0);
        result = (1.0/1000. + 1./10.)*y*1000000.;
        break;
    default:
        result = 0;
        break;
    }
return(result);
```

GENERAL FUNCTIONS LISTING

```
/*--------------------------- GEN.H ------------------------*/

/* REV 2.00 7/6/88 by gkm - functions for printing titles,
                            file handling, and entry input */
/* REV 2.10 7/12/88 by gkm - added extended screen functions
                            and pop-up windows to all screen
                            functions                       */
/* REV 2.11 7/13/88 by gkm - added pop-up window messages  */
/* REV 3.01 5/4/89 by gkm   */

#include <bscreen.h>
#include <bwindow.h>

#define TRUE    1
#define FALSE    0
int sys_mode,sys_col;
WHERE sys;                  /* Screen setup variables */

/* --------------- ERROR MESSAGE DEFINES ---------------- */

    /* --- COM PORT ERRORS --- */

    static char *pcodemsg[] = {
        "Successful",
        "Reserved for future use",
        "Invalid COM port number",
        "COM port is not opened",
        "Invalid parameter or function value",
        "Reserved for future use",
        "No serial port found",
        "Output queue is full",
        "Reserved for future use",
        "COM port is already open",
        "Input queue is empty"
        };

    /* --- WINDOW ERRORS --- */

    static char *wcodemsg[] = {
        "Successful",
        "Insufficient memory for allocation",
        "Invalid window dimension",
        "Null pointer encountered",
        "Invalid window pointer",
        "Display device does not exist",
        "Nonexistent display page",
        "Invalide pointer to window node",
        "Display window already displayed",
        "Window is not attached to display location",
        "Window is not removable",
        "Window's data area is covered by another",
        "Unknown option code",
        "Option cannot be changed",
        "Invalid parameter for option"
        };
```

```
/*------------------- FUNCTION DECLARATIONS ---------------*/

                    /* FILE HANDLING FUNCTIONS */

extern FILE *fil_open(void);
extern FILE *fil_open_wn(void);
extern FILE *fil_creat(void);
extern FILE *fil_creat_wn(void);

            /* SCREEN FUNCTIONS */

extern void signon(char *,char *,char *);
extern void signon_box(char *,char *,char *);
extern void title_box(char *);
extern void hit_cont(void);
extern void pause(char *);
extern BWINDOW *start_wn_on(char *);
extern BWINDOW *msg_wn_on(char *);
extern void hit_cont_wn(void);

            /* ENTRY FUNCTIONS */

extern char *reply_wn(char *);
extern char get_char(char,char);
extern int get_no(int,int);
extern int get_no_wn(char *,int,int);
extern char get_yn_wn(int,char *);
extern int get_key(void);

            /* WINDOW FUNCTIONS */

extern BWINDOW *window_make(int,int,int,int);
extern void msg_wn_off(BWINDOW *);

            /* ERROR FUNCTIONS */

extern void window_err_wn(char *,int);
extern void err_wn(char *);
extern void com_err_wn(char *,int,unsigned);

            /* SYSTEM FUNCTIONS */

extern void scrn_ck(void);
```

```
/*------------ GENERAL PURPOSE FUNCTIONS ------------------*/

/* REV 2.00 7/6/88 by gkm - functions for printing titles,
                            file handling, and entry input */
/* REV 2.10 7/12/88 by gkm - added extended screen functions
                            and pop-up windows to all screen
                            functions                      */
/* REV 2.11 7/13/88 by gkm - added pop-up window messages  */
/* REV 3.01 5/4/89 by gkm - rewrote window functions
                          added reply_wn,title_box,get_key,
                          com_err_wn,window_err_wn,pause,
                          start_wn_on, and added row input
                          to get_yn_wn.                    */
#include <stdio.h>
#include <dos.h>
#include <string.h>
#include <bscreen.h>
#include <bwindow.h>
#include <bfile.h>
#include "gen.h"

/******************** FILE HANDLING FUNCTIONS ***************/

/*---------------- OPEN A FILE FOR READING -----------------*/

FILE *fil_open()
{
    /*---- RETURNS A POINTER TO THE OPENED FILE -----------*/

    FILE *read_fil,*fopen();
    char fil[67];

    do
        {
        /*-------------- ENTER FILE NAME ------------------*/

        printf("Enter file name for reading data: \0");
        gets(fil);

        /*-------------- SEE IF FILE EXISTS ----------------*/

        if((read_fil = fopen(fil,"rb")) == NULL)
            printf("Cannot find %s",fil);
        }
    while(read_fil == NULL);

    return(read_fil);
}
```

```
/*--------- OPEN A FILE FOR READING (WINDOW MODE) ---------*/

FILE *fil_open_wn()
{
    /*------- RETURNS A POINTER TO THE OPENED FILE --------*/

    FILE *read_fil,*fopen();
    char fil[67];
    int fore = BLACK,back = WHITE;
    BWINDOW *win_pt;
    int w_row,w_col,scan;

    if((win_pt = window_make(2,70,22,5)) == NULL)
        return(0);
    wnwrap(0,"Enter file name for reading data: \0",-1,-1,CHARS_ONLY);
    wncurpos(&w_row,&w_col);

    do
        {

        /*-------------- ENTER THE FILE NAME --------------*/

        wnquery(fil,sizeof(fil),&scan);

        /*--------------- SEE IF FILE EXISTS --------------*/

        if((read_fil = fopen(fil,"rb")) == NULL)
            {
            if(sys.dev == COLOR)
                fore = RED;
            scdspmsg(23,6,fore,back,"Cannot find ");
            if(sys.dev == COLOR)
                fore = BLUE;
            scdspmsg(23,18,fore,back,fil);
            wncurmov(w_row,w_col);
            }
        }
    while(read_fil == NULL);

    /*------------------- REMOVE THE WINDOW ---------------*/

    msg_wn_off(win_pt);
    return(read_fil);
}
```

```
/*--------------- OPEN A FILE FOR WRITING -----------------*/

FILE *fil_creat()
{
    /*------- RETURNS A POINTER TO THE OPENED FILE --------*/

    FILE *writ_fil,*fopen();
    char c,fil[67];

    do
        {
        /*------------- ENTER FILE NAME -------------------*/

        printf("Enter file name for writing data: ");
        gets(fil);

        /*------------- SEE IF FILE EXISTS ----------------*/

        if(fopen(fil,"r") == NULL)     /* No, then open it */
            {
            if((writ_fil = fopen(fil,"wb")) == NULL)
                printf("Cannot open %s for writing\n",fil);
            }

        /* IF FILE EXISTS, ASK IF TO OVERWRITE EXISTING DATA */
        else
            {
            printf("%s already exists. Overwrite?[N]\b\b",fil);
            do
                {
                c = toupper(getch());
                }
            while(c != 'N' && c != 'Y');
            printf("%c]\n",c);

            if(c == 'Y')
                {
                if((writ_fil = fopen(fil,"wb")) == NULL)
                    printf("Could not open %s for writing\n",fil);

                }
            else
                writ_fil = NULL;
            }
        }
    while(writ_fil == NULL);
    return(writ_fil);
}
```

```
/*-------- OPEN A FILE FOR WRITING (WINDOW MODE) ---------*/

FILE *fil_creat_wn()
{
    /*------- RETURNS A POINTER TO THE OPENED FILE --------*/

    FILE *fopen(),*writ_fil;
    int handle;
    char c,fil[67];
    int error,fore = BLACK,back = WHITE;
    BWINDOW *win_pt;
    int w_row,w_col,scan;

    if((win_pt = window_make(2,70,22,5)) == NULL)
        return(0);
    wnwrap(0,"Enter file name for writing data: ",-1,-1,CHARS_ONLY);
    wncurpos(&w_row,&w_col);

    do
        {

        /*-------------- ENTER THE FILE NAME --------------*/

        wnquery(fil,sizeof(fil),&scan);

        /*--------------- SEE IF FILE EXISTS --------------*/

        error = flnew(fil,&handle,AT_GENERAL);
        switch(error)
            {
            case 0:                  /* Successful opening */
                flclose(handle);
                if((writ_fil = fopen(fil,"wb")) == NULL)
                    {
                    if(sys.dev == COLOR)
                        fore = BLUE;
                    scdspmsg(23,6,fore,back,fil);
                    wncurmov(w_row,w_col);
                    if(sys.dev == COLOR)
                        fore = RED;
                    scdspmsg(23,21,fore,back,"cannot be opened.");
                    }
                break;
            case 3:
                if(sys.dev == COLOR)
                    fore = RED;
                scdspmsg(23,6,fore,back,"Invalid path!");
                writ_fil = NULL;
                break;
            case 4:
                if(sys.dev == COLOR)
                    fore = RED;
                scdspmsg(23,6,fore,back,"Too many open files!");
                writ_fil = NULL;
                break;
```

```
            case 5:
                if(sys.dev == COLOR)
                    fore = BLUE;
                scdspmsg(23,6,fore,back,fil);
                wncurmov(w_row,w_col);
                if(sys.dev == COLOR)
                    fore = RED;
                scdspmsg(23,21,fore,back,
                    "cannot be opened for writing");
                break;
            case 80:
                if(sys.dev == COLOR)
                    fore = BLUE;
                scdspmsg(23,6,fore,back,fil);
                wncurmov(w_row,w_col);
                if(sys.dev == COLOR)
                    fore = RED;
                scdspmsg(23,21,fore,back,
                    "already exists. Overwrite?(y/n)");
                do
                    {
                    c = toupper(getch());
                    }
                while(c != 'N' && c != 'Y');

                if(c == 'Y')
                    {
                    if((writ_fil = fopen(fil,"wb")) == NULL)
                        {
                        if(sys.dev == COLOR)
                            fore = BLUE;
                        scdspmsg(23,6,fore,back,fil);
                        wncurmov(w_row,w_col);
                        if(sys.dev == COLOR)
                            fore = RED;
                        scdspmsg(23,21,fore,back,"cannot be opened.");
                        }
                    }
                else
                    {
                    scclrmsg(23,6,50);
                    scclrmsg(22,39,15);
                    writ_fil = NULL;
                    }
                break;
            default:
                writ_fil = NULL;
                break;
            }
        }
    while(writ_fil == NULL);
    /*------------------- REMOVE THE WINDOW ---------------*/
    msg_wn_off(win_pt);
    return(writ_fil);
}
```

```
/************************ SCREEN FUNCTIONS ******************/

/*------------------- DISPLAY SIGNON MESSAGES --------------*/

void signon(prgm_ttl,clnt_ttl,ver_ttl)
char *prgm_ttl,*clnt_ttl,*ver_ttl;
{
    system("cls");
    puts(prgm_ttl);
    puts(clnt_ttl);
    puts(ver_ttl);
}

/*---------- DISPLAY SIGNON MESSAGES (BOX DISPLAY) --------*/

void signon_box(prgm_ttl,clnt_ttl,ver_ttl)
char *prgm_ttl,*clnt_ttl,*ver_ttl;
{
    int fore,back,col;

    if(sys.dev == MONO)
        {
        fore = WHITE;
        back = BLACK;
        }
    else
        {
        fore = WHITE;
        back = BLUE;
        }

    scclear();
    scbox(0,0,4,79,0,-1,NORMAL);
    scatrect(0,0,4,79,fore,back);
    col = (80 - strlen(prgm_ttl))/2;
    scdspmsg(1,col,fore,back,prgm_ttl);
    col = (80 - strlen(clnt_ttl))/2;
    scdspmsg(2,col,fore,back,clnt_ttl);
    col = (80 - strlen(ver_ttl))/2;
    scdspmsg(3,col,fore,back,ver_ttl);
}
```

```
/* ---------- DISPLAY TITLE (BOX DISPLAY) ------------ */

void title_box(title)
char *title;
{
    int fore,back,count;

    if(sys.dev == MONO)
        {
        fore = WHITE;
        back = BLACK;
        }
    else
        {
        fore = WHITE;
        back = BLUE;
        }

    scclear();
    scbox(0,0,2,79,0,-1,NORMAL);
    scatrect(0,0,2,79,fore,back);
    count = (80 - strlen(title))/2;
    scdspmsg(1,count,fore,back,title);
}

/*------------------- HIT ANY KEY TO CONTINE ---------------*/

void hit_cont()
{
    puts("Hit any key to continue....");
    getch();
}
```

```
/*-------------- PAUSE WITH A MESSAGGE (WINDOW) ----------*/

void pause(msg)
char *msg;
{
    int length;
    BWINDOW *pwin;

    if(strlen(msg) > 24)
        length = strlen(msg) + 4;
    else
        length = 28;
    if((pwin = window_make(4,length,10,19)) == NULL)
        return;
    if(wnsetopt(pwin,WN_CUR_OFF,1) == NULL)
        window_err_wn("Could not disable window cursor",0);
    wncursor(pwin);
    if(wncurmov(1,2) == NULL)
        window_err_wn("Could not move window cursor",0);
    wnwrap(0,msg,-1,-1,CHARS_ONLY);
    if(wncurmov(2,length/2 - 12) == NULL)
        window_err_wn("Could not move window cursor",0);
    wnwrap(0," HIT ANY KEY WHEN READY",-1,-1,CHARS_ONLY);

    getch();
    msg_wn_off(pwin);
}

/*-------------- DISPLAY A START MESSAGGE (WINDOW) --------- */

BWINDOW *start_wn_on(msg)
char *msg;
{
    int length;
    BWINDOW *pwin;

    if(strlen(msg) > 24)
        length = strlen(msg) + 4;
    else
        length = 28;
    if((pwin = window_make(4,length,10,19)) == NULL)
        return(0);
    if(wncurmov(1,2) == NULL)
        window_err_wn("Could not move window cursor",0);
    wnwrap(0,msg,-1,-1,CHARS_ONLY);
    if(wncurmov(2,length/2 - 12) == NULL)
        window_err_wn("Could not move window cursor",0);
    wnwrap(0," HIT F1 KEY WHEN DONE ",-1,-1,CHARS_ONLY);
    if(wnsetopt(pwin,WN_CUR_OFF,1) == NULL)
        window_err_wn("Could not disable window cursor",0);
    wncursor(pwin);
    return(pwin);
}
```

```
/*------------- DISPLAY A MESSAGGE (WINDOW) ----------------*/

BWINDOW *msg_wn_on(msg)
char *msg;
{
    BWINDOW *pwin;

    if((pwin = window_make(3,strlen(msg) + 4,10,19)) == NULL)
        return(0);
    if(wncurmov(1,2) == NULL)
        window_err_wn("Could not move window cursor",0);
    wnwrap(0,msg,-1,-1,CHARS_ONLY);
    if(wnsetopt(pwin,WN_CUR_OFF,1) == NULL)
        window_err_wn("Could not disable window cursor",0);
    wncursor(pwin);
    return(pwin);
}

/*------- HIT ANY KEY TO CONTINUE (WINDOW) ----------------*/

void hit_cont_wn()
{
    BWINDOW *pwin;

    /*--------------- DISPLAY THE MESSAGE -----------------*/

    if((pwin = window_make(1,28,23,5)) == NULL)
        return;
    if(wncurmov(0,2) == NULL)
        window_err_wn("Could not move window cursor",0);
    wnwrap(0,"HIT ANY KEY TO CONTINUE",-1,-1,CHARS_ONLY);
    if(wnsetopt(pwin,WN_CUR_OFF,1) == NULL)
        window_err_wn("Could not disable window cursor",0);
    wncursor(pwin);
    getch();

    /*----------------- REMOVE THE WINDOW -----------------*/

    msg_wn_off(pwin);
}
```

```
/********************** ENTRY FUNCTIONS ********************/

/* -------------- GET A REPLY(WINDOW MODE) ------------ */

char *reply_wn(msg)
char *msg;
{
    /*------- RETURNS A POINTER TO THE RESPONSE --------*/

    BWINDOW *pwin;
    int scan;
    char reply[80];

    if((pwin = window_make(1,70,22,5)) == NULL)
        return(0);
    wnwrap(0,msg,-1,-1,CHARS_ONLY);

    /* -------------- GET THE REPLY -------------- */

    wnquery(reply,sizeof(reply),&scan);

    msg_wn_off(pwin);
    reply[sizeof(reply) + 1] = '\0';
    return(reply);
}

/*---- GET A CHARACTER ENTRY AND CHECK FOR VALID OPTION ---*/

char get_char(lower,upper)
char upper,lower;
{
    char c;

    do
        {
        c = toupper(getch());
        }
    while(c < lower || c > upper);
    printf("%c\n",c);
    return (c);
}
```

```
/*------------------- GET A NUMBER ENTRY ------------------*/

int get_no(lower,upper)
int lower,upper;
{
    int no;
    char msg[20];

    do
        {
        gets(msg);
        no = atoi(msg);
        }
    while(no < lower || no > upper);
    return no;
}

/*--------------- GET A NUMBER ENTRY (WINDOW MODE) --------*/

get_no_wn(msg,lower,upper)
char *msg;
int lower,upper;
{
    BWINDOW *pwin;
    int i,num,w_row,w_col,scan;
    char response[10];

    if((pwin = window_make(1,strlen(msg) + 15,22,5)) == NULL)
        return(0);
    wnwrap(0,msg,-1,-1,CHARS_ONLY);
    if(wncurpos(&w_row,&w_col) == NULL)
        window_err_wn("Could not save window cursor position",0);

    /*-------------- GET THE CHECKED NUMBER ENTRY ----------*/

    do
        {
        wnquery(response,sizeof(response),&scan);
        num = atoi(response);
        if(wncurmov(w_row,w_col) == NULL)
            window_err_wn("Could not move window cursor",0);
        for(i = 0;i < strlen(response);i++)
            wnwrap(1," ",-1,-1,CHARS_ONLY);
        if(wncurmov(w_row,w_col) == NULL)
            window_err_wn("Could not move window cursor",0);
        }
    while(num < lower || num > upper);

    msg_wn_off(pwin);
    return(num);
}
```

```
/*------- PROMPT USER FOR YES OR NO -----------------*/

char get_yn_wn(loc,msg)
char *msg;
int loc;
{
    BWINDOW *pwin;
    int w_row,w_col,scan;
    char c,response[5];

    if((pwin = window_make(1,strlen(msg) + 5,loc,5)) == NULL)
        return(0);
    wnwrap(0,msg,-1,-1,CHARS_ONLY);
    if(wncurpos(&w_row,&w_col) == NULL)
        window_err_wn("Could not save window cursor position",0);

    /*---------- LOOP UNTIL A YES OR NO IS ENTERED --------*/

    do
        {
        wnquery(response,sizeof(response),&scan);
        c = toupper(response[0]);
        if(wncurmov(w_row,w_col) == NULL)
            window_err_wn("Could not move window cursor",0);
        }
    while(c != 'Y' && c != 'N');

    msg_wn_off(pwin);
    return(c);
}


/* ---------------- GET NEXT KEY FROM KEYBOARD ---------  */

int get_key()
{
     union   REGS    regs;

       regs.h.ah = 0;
     int86(0x16, &regs, &regs);
     return(regs.x.ax);
}
```

```
/********************** WINDOW CREATE FUNCTION ************/

BWINDOW *window_make(size_r,size_c,row,col)
int size_r,size_c,row,col;
{
    int fore,back;
    BWINDOW *pwin;
    BORDER bord;
    int w_row,w_col;

    /*------------ DETERMINE MONITOR TYPE -----------------*/

    if(sys.dev == MONO)
        {
        fore = -1;
        back = -1;
        }
    else
        {
        fore = RED;
        back = BLACK;
        }

    /*---------------- CREATE THE WINDOW ------------------*/

    if((pwin = wncreate(size_r,size_c,REVERSE)) == NULL)
        {
        window_err_wn("Could not create window",0);
        return(0);
        }
    bord.type = 1;
    if(sys.dev == COLOR)
        bord.attr = RED;
    else
        bord.attr = -1;

    sys.corner.row = row;
    sys.corner.col = col;
    if(wndsplay(pwin,&sys,&bord) == NULL)
        window_err_wn("Could not display window",b_wnerr);
    if(wnsetopt(pwin,WN_DELAYED,0) == NULL)
        window_err_wn("Could not set window option",0);
    return(pwin);
}
```

```
/*------------ REMOVE THE DISPLAYED MESSAGE (WINDOW) -------*/

void msg_wn_off(pwin)
BWINDOW *pwin;
{
    int ercode;

    if(wnremove(pwin) == NULL)
        window_err_wn("Could not remove window",b_wnerr);
    if((ercode = wndstroy(pwin)) != NULL)
        window_err_wn("Could not destroy window",ercode);
}
```

```
/******************** ERROR FUNCTIONS **********************/

/* --------- DISPLAY THE WINDOW ERROR MESSAGE ---------- */

void window_err_wn(pmsg,code)
char *pmsg;
int code;
{
    int fore,back;
    BWINDOW *pwin;
    BORDER bord;
    int cursor_was_off,row,col,high,low;
    char msg[80];

    /*------------ DETERMINE MONITOR TYPE -----------------*/

    if(sys.dev == MONO)
        {
        fore = -1;
        back = -1;
        }
    else
        {
        fore = RED;
        back = BLACK;
        }

    /*---------------- CREATE THE WINDOW ------------------*/

    if((pwin = wncreate(3,60,REVERSE)) == NULL)
        {
        puts("ERROR: Could not create the error window");
        return;
        }
    bord.type = 1;
    if(sys.dev == COLOR)
        bord.attr = RED;
    else
        bord.attr = BLACK;

    sys.corner.row = 19;
    sys.corner.col = 5;
```

```
    /*--------------- DISPLAY THE MESSAGE -----------------*/

    cursor_was_off = sccurst(&row,&col,&high,&low);
    wndsplay(pwin,&sys,&bord);
    wncurmov(0,27);
    wnwrap(0,"ERROR",bord.attr,-1,CHARS_ONLY);
    wncurmov(1,2);
    wnwrap(0,pmsg,-1,-1,CHARS_ONLY);
    wncurmov(2,2);
  if (code != 0)
        {
     sprintf(msg," Error code - %02d  %s",code,wcodemsg[code]);
         wnwrap(0,msg,-1,-1,CHARS_ONLY);
       }
    hit_cont_wn();

    /*----------------- REMOVE THE WINDOW -----------------*/

    msg_wn_off(pwin);
}

/*--------- DISPLAY ERROR MESSAGE (WINDOW) ----------------*/

void err_wn(msg)
char *msg;
{
    BWINDOW *pwin;
    int len;

    len = strlen(msg) + 4;

    /*--------------- DISPLAY THE MESSAGE -----------------*/

    if((pwin = window_make(2,len,18,5)) == NULL)
        return;
    if(wncurmov(0,(len/2) - 3) == NULL)
        window_err_wn("Could not move window cursor",0);
    wnwrap(0,"ERROR",RED,-1,CHARS_ONLY);
    if(wncurmov(1,2) == NULL)
        window_err_wn("Could not move window cursor",0);
    wnwrap(0,msg,-1,-1,CHARS_ONLY);
    hit_cont_wn();

    /*----------------- REMOVE THE WINDOW -----------------*/

    msg_wn_off(pwin);
}
```

```
/* --------- DISPLAY THE COM PORT ERROR MESSAGE ---------- */

void com_err_wn(pmsg,code,status)
char *pmsg;
int code;
unsigned status;
{
    char msg[50];
    BWINDOW *pwin;

    /*--------------- DISPLAY THE MESSAGE -----------------*/

    if((pwin = window_make(4,60,19,5)) == NULL)
        return;
    if(wncurmov(0,27) == NULL)
        window_err_wn("Could not move window cursor",0);
    wnwrap(0,"ERROR",RED,-1,CHARS_ONLY);
    if(wncurmov(1,2) == NULL)
        window_err_wn("Could not move window cursor",0);
    wnwrap(0,pmsg,-1,-1,CHARS_ONLY);
    if(wncurmov(2,2) == NULL)
        window_err_wn("Could not move window cursor",0);
  if (code != 0)
        {
     sprintf(msg," Error code - %02d  %s",code,pcodemsg[code]);
        wnwrap(0,msg,-1,-1,CHARS_ONLY);
        }
    if(wncurmov(3,2) == NULL)
        window_err_wn("Could not move window cursor",0);
  if (status != 0)
     {
        sprintf(msg," Port status - %04x",status);
        wnwrap(0,msg,-1,-1,CHARS_ONLY);
        }
    hit_cont_wn();

    /*----------------- REMOVE THE WINDOW -----------------*/

    msg_wn_off(pwin);
}
```

```
/* --------------- SET HARDWARE ENVIRONMENT ------------ */

void scrn_ck()
{
    int mode,columns,act_page;

    /* --- LOAD EQUIPMENT VARIABLES --- */

    scequip();

    /* --- SAVE CURRENT SETUP --- */

    sys.dev = scmode(&mode,&columns,&act_page);
    sys.page = act_page;
    sys_mode = mode;
    sys_col = columns;
}
```