NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

NPS STATE VECTOR ANALYSIS AND RELATIVE MOTION PLOTTING SOFTWARE FOR STS-51 APPENDIX C

by

Lieutenant Lee A. Barker

March, 1994

Thesis Advisor: Dr. Rudolf Panholzer

Approved for public release; distribution is unlimited.

APPENDIX C

NPS SOFTWARE SOURCE CODE

Appendix C contains source code for the NPS software. The following files are included and appear in the order shown:

NPS directory files -
dif_data.c
dif_plot.c
f_and_g.c
lsfilter.c
menuplot.c
nps.c
nps.h
nps_dap.c
nps_edit.c
nps_filt.c
nps_m50.c
nps_m50.h
nps_none.c
nps_plot.c
nps_pred.c
py_data.c
py_plot.c
rv_data.c
rv_plot.c

NPS_SIM directory files -
gcom_com.c
gcom_dev.c
gdisk.c
gnone.c
nps_main.c
scom.c
scom_old.c
sdisk.c
sdisk_2.c
sim.c
sim.h
smem.c
sprop.c
spropdsk.c

COM directory files -
com_inta.asm
com_intc.c
orb_file.c
orb_pkt.c
orb_port.c
tans_pkt.c
tansfile.c
tansport.c
tanspkt2.c

ORBMECH1 directory files -
amatrix.c
bgprop.c
cowell.c
der.c
gem9.c
getiers.c
gotpot.c
iers.h
itowgs84.c
jacatm.c
jacdat.c
m50rnp.c
rk4.c
rnpd.c
rnpm50.c
util.c
util.h

UTIL directory files -
edit.c
edit_fld.c
edit_rt.c
emm_cons.c
emm_lowa.asm
emm_map.c
emm_mapm.c
key_get.asm
key_int9.asm
swap_d.asm
swap_f.asm
swap_s.asm
tdbl_hms.asm
tdbl_str.asm
thms_dbl.asm
thms_str.asm
timer.asm
v_j2000.c
v_m50.c
v_matrix.c
v_quat.c
v_rvbar.c
v_vector.c

INC directory files -
com_port.h
com_port.inc
const.h
edit.h
edit_rt.h
emm.h
keys.h
nps_if.h
orbmech1.h
packet.h
pkt_if.h
swap.h
tans_pkt.h
times.h
times.inc
types.h
vector.h

TEST directory files -
com_txrx.c
emm_test.c
les.c
map_raw.c
orb.c
tans_vec.c
tansdump.c
test.c
test_com.c
test_dev.c
test_vec.c
test1.c
test2.c
vect_old.c
vmode.c

TANS directory files -
(not included)

TANS_SIM directory files -
(not included)

Borland C++ Makefile-
makefile

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#include <alloc.h>
#include <mem.h>
#include <io.h>
#include <fcntl.h>
#include <math.h>

#include "nps.h"

/*=============================================================
============================
 *-------------------------------------------------------------*/
int nps_diffplot__data_init(Nps_diffplot_info *this,
                            ushort max_mem, ushort config_mem, Emm_info *emm)
{
   int handle;
   Nps_log_file_header header;

   /*-----------------------------------------------------------
    *  Try to get EMM memory first, and 64K of it.  Otherwise use normal mem.
    */
   if (emm_info_exists(emm))  emm_construct_new(&this->emm, emm, 4);

   if (emm_exists(&this->emm))
   {
        char name[10];

        memcpy(name, this->log_file_name, 8);
        emm_name_set(&this->emm, name);

        this->ring_max = 0xffff / sizeof(Diff_vector);
        max_mem = sizeof(Diff_vector) * this->ring_max;
        this->ring_buffer = MK_FP(emm->phys.segments[0], 0);

        emm_map_multi(&this->emm, EMM_phys_page, 4, emm->map_0123);
   }

   /*-----------------------------------------------------------
    *  Ring_max may already be initialized, if not then compute it from the
    * max_mem argument.  Allocate the ring buffer.
    */
   else
   {
        if (config_mem > 1024)
           this->ring_max = config_mem / sizeof(Diff_vector);

        if (this->ring_max == 0)
           this->ring_max = max_mem / sizeof(Diff_vector);
```

```
        this->ring_buffer = malloc(max_mem = sizeof(Diff_vector) * this->ring_max);

        if (this->ring_buffer == NULL)  return NO;
    }

    /*-------------------------------------------------------------------
     *  Clear the ring buffer, and try to open my log-file.
     */
    memset(this->ring_buffer, 0, max_mem);

    if ((handle = open(this->log_file_name, O_RDONLY|O_BINARY)) < 0)  return YES;

    /*-------------------------------------------------------------------
     *  Read and verify the log-file header pattern, then fill my ring buffer
     * from the file.  I assume all samples are in time order in the file.
     */
    if (read(handle, &header, sizeof(header)) == sizeof(header)
     && memcmp(&header, this->log_file_hdr, sizeof(header)) == 0)
    {
        this->next_sample = read(handle, this->ring_buffer, max_mem) / sizeof(Diff_vector);
        this->prev_sample = this->next_sample - 1;

        if (this->next_sample >= this->ring_max)
        {
            this->next_sample = 0;
            this->ring_full = YES;
        }
        else if (this->next_sample <= 0)
            this->prev_sample =  this->next_sample =  0;

        this->begin_time = this->ring_buffer[0].time;
    }

    close(handle);
    return YES;
}

/*===================================================================
=========================
 *-------------------------------------------------------------------*/
void nps_diffplot__data_stop(Nps_diffplot_info *this)
{
    if (emm_exists(&this->emm))
        emm_destroy(&this->emm);
}

void nps_diffplot__data_save(Nps_diffplot_info *this)
{
    int handle;
    ushort size;
```

```
    /*-------------------------------------------------------------
     *  Try to open/create the log-file.  Write the header pattern.
     */
    if ((handle = open(this->log_file_name, O_RDWR|O_BINARY|O_CREAT, 0666)) < 0)  return;

    write(handle, this->log_file_hdr, sizeof(Nps_log_file_header));

    /*-------------------------------------------------------------
     *  I must write the samples in time order.
     */
    if (emm_exists(&this->emm))
        emm_map_multi(&this->emm,  EMM_phys_page, 4, this->emm.info->map_0123);

    if (this->ring_full)
    {
        size = (this->ring_max - this->next_sample) * sizeof(Diff_vector);
        if (size)  write(handle, this->ring_buffer + this->next_sample, size);
    }
    size = this->next_sample * sizeof(Diff_vector);
    if (size)  write(handle, this->ring_buffer, size);

    _write(handle, 0, 0);  // Truncate file to current size
    close(handle);
}

/*=============================================================
 *-------------------------------------------------------------*/
void nps_diffplot__data_flush(Nps_diffplot_info *this)
{
    this->ring_full = NO;
    this->prev_sample =  this->next_sample =  0;
}

/*=============================================================
 *-------------------------------------------------------------*/
void nps_diffplot__data_update(Nps_diffplot_info *this)
{
    Nps_state_vector_info *target = this->target;
    Nps_state_vector_info *chaser = this->chaser;
    Diff_vector *dvec = this->ring_buffer + this->next_sample;
    Vector delta;

    /*-------------------------------------------------------------
     */
    if (target->got_one == NO  ||  chaser->got_one == NO)  return;

    if (emm_exists(&this->emm))
        emm_map_multi(&this->emm,  EMM_phys_page, 4, this->emm.info->map_0123);
```

```
    vector_diff(&chaser->sv.pos, &target->sv.pos, &delta);
    dvec->diff[NPS_diff_pos] = vector_magnitude(&delta);

    vector_diff(&chaser->sv.vel, &target->sv.vel, &delta);
    dvec->diff[NPS_diff_vel] = vector_magnitude(&delta);

    dvec->time = chaser->sv.time;

    if (this->prev_sample == this->next_sample && this->begin_time == 0.0)
        this->begin_time = chaser->sv.time;

    /*-----------------------------------------------------------------
     */
    this->prev_sample = this->next_sample;
    this->next_sample++;
    if (this->next_sample >= this->ring_max)
    {
        this->next_sample = 0;
        this->ring_full = YES;
    }
}

/*==========================================================================
==========================
 *-----------------------------------------------------------------------*/
void nps_diffplot__display_update(Nps_diffplot_info *this, double t_time, double c_time)
{
    Nps_state_vector_info *target = this->target;
    Nps_state_vector_info *chaser = this->chaser;
    Vector delta;

    /*-----------------------------------------------------------------
     */
    if (target->got_one == NO || chaser->got_one == NO)
    {
        nps_diffplot__time(t_time, c_time);
        return;
    }

    /*-----------------------------------------------------------------
     */
    if (this->active_plot == NPS_diff_pos)
        vector_diff(&chaser->sv.pos, &target->sv.pos, &delta);
    else
        vector_diff(&chaser->sv.vel, &target->sv.vel, &delta);

    if (this->prev_sample == this->next_sample && this->begin_time == 0.0)
        this->begin_time = chaser->sv.time;

    nps_diffplot__point(this->displays + this->active_plot, vector_magnitude(&delta),
```

```
    chaser->sv.time - this->begin_time, this->active_plot, t_time, c_time);
}

/*=============================================================
============================
 * Returns:
 * 0    Didn't use the key
 * 1    Used the key
 * 2    Used the key, My subsystem is done
 *-------------------------------------------------------------*/
int nps_diffplot__key_handler(Nps_diffplot_info *this, int key)
{
   int rc = YES, refresh = YES;

   Nps_plot_info *display = this->displays + this->active_plot;
   Nps_plot_axis *axis_horz = display->axis + display->axis_horz;
   Nps_plot_axis *axis_vert = display->axis + display->axis_vert;
   Nps_plot_axis *axis;

   double range_delta;

   /*-----------------------------------------------------------
    */
   switch (key)
   {
        case KEY_esc:  return 2;

        /*-----------------------------------------------------
         */
        case KEY_up:    axis_vert->range *= 2;  break;
        case KEY_down:  axis_vert->range /= 2;  break;
        case KEY_left:  axis_horz->range /= 2;  break;
        case KEY_right: axis_horz->range *= 2;  break;

        /*-----------------------------------------------------
         */
        case KEY_cntl_up:
        case KEY_cntl_down:
        case KEY_cntl_left:
        case KEY_cntl_right:
           if (key == KEY_cntl_up  ||  key == KEY_cntl_down)
           {
                axis = axis_vert;
                range_delta = axis->range/4.0;
           }
           else
           {
                axis = axis_horz;
                range_delta = axis->range/6.0;
           }
```

```
            if (key == KEY_cntl_up || key == KEY_cntl_right)
            {
                if (axis->origin < 0 && -axis->origin <= range_delta)
                    axis->origin = 0.0;
                else axis->origin += range_delta;
            }
            else
            {
                if (axis->origin > 0 && axis->origin <= range_delta)
                    axis->origin = 0.0;
                else axis->origin -= range_delta;
            }
            break;

        /*-----------------------------------------------------------------
         */
        case KEY_space:
            if (this->menu_list[this->menu_id] == NULL) this->menu_id = 0;
            else                                        this->menu_id++;
            break;

        case KEY_f9:
            this->active_plot =
              (this->active_plot == NPS_diff_pos ? NPS_diff_vel : NPS_diff_pos);
            break;

        default: rc = NO; break;
    }

    /*-----------------------------------------------------------------
     */
    if (rc == YES && refresh == YES) nps_diffplot__display_init(this);

    return rc;
}

/*=================================================================
=====================
 *-----------------------------------------------------------------*/
void nps_diffplot__display_init(Nps_diffplot_info *this)
{
    Diff_vector *sample;
    ushort samples;

    int type = this->active_plot;
    Nps_plot_info *display = this->displays + this->active_plot;
    double begin_time;

    /*-----------------------------------------------------------------
     */
```

```
    if (emm_exists(&this->emm))
        emm_map_multi(&this->emm, EMM_phys_page, 4, this->emm.info->map_0123);

    nps_diffplot__init(display, type, this->menu_list[this->menu_id]);

    /*-----------------------------------------------------------------
     *  Figure out how many Diff samples are in the ring buffer, then process
     * all the valid samples.
     */
    if (this->ring_full == YES)
    {
        samples = this->ring_max;
        begin_time = this->begin_time = this->ring_buffer[this->next_sample].time;
    }
    else
    {
        samples = this->next_sample;
        begin_time = this->begin_time = this->ring_buffer[0].time;
    }

    /*-----------------------------------------------------------------
     */
    for (sample = this->ring_buffer; samples > 0; samples--, sample++)
        nps_diffplot__point_fast(display, sample->diff[type], sample->time - begin_time);
}
```

```
/*=============================================================
=========================
 *-------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>

#include <times.h>
#include "nps.h"

/*=============================================================
=========================
 *-------------------------------------------------------------*/
extern Nps_graphics_info nps_graphics_info;

/*=============================================================
=========================
 *  I use these structures to avoid extra floating-point work during realtime.
 *-------------------------------------------------------------*/
static struct                           // Plot times
{
   struct viewporttype view;
   int line1_y, line2_y;
} time;

static struct                           // Mag/Time numbers
{
   struct viewporttype view;
   int line1_y, line2_y;
} mag_time;

static struct
{
   int x1, x2, y1, y2;
} axis;

#define STEP_PIXELS 100
#define TIME_PIXELS (STEP_PIXELS * 6)
#define MAG_PIXELS  (STEP_PIXELS * 4)

/*=============================================================
=========================
 *-------------------------------------------------------------*/
static void points_init(void)
{
   int xmax = nps_graphics_info.xmax;
   int ymax = nps_graphics_info.ymax;
   int c_height = nps_graphics_info.c_height;
   int c_width = nps_graphics_info.c_width;

   /*-----------------------------------------------------------
```

```
    */
    time.view.left   = xmax * 0.85;                    // Top right corne-
    time.view.right  = time.view.left + c_width * 16;
    time.view.top    = ymax * 0.05;
    time.view.bottom = time.view.top + c_height * 3;
    time.view.clip   = YES;
    if (time.view.right > xmax)  time.view.right = xmax - 1;
    if (time.view.bottom > ymax) time.view.bottom = ymax - 1;

    time.line1_y = c_height;
    time.line2_y = (c_height * 5)/2;

    /*-----------------------------------------------------------------
    */
    mag_time.view.left   = xmax * 0.855;
    mag_time.view.right  = mag_time.view.left + c_width * 15;
    mag_time.view.top    = ymax * 0.865;
    mag_time.view.bottom = mag_time.view.top + c_height * 3;
    mag_time.view.clip   = YES;
    if (mag_time.view.right >= xmax)  mag_time.view.right = xmax - 1;

    mag_time.line1_y = c_height;
    mag_time.line2_y = (c_height * 5)/2;

    /*-----------------------------------------------------------------
    */
    axis.x1 = 10;
    axis.x2 = xmax - 1;
    axis.y1 = 0;
    axis.y2 = ymax * 0.834;
}

/*=================================================================
=========================
 *-----------------------------------------------------------------*/
void nps_diffplot__init(Nps_plot_info *info, int diff_type, char **menu)
{
    static char
        mag_title[] = "mag(r)",
        mag_label[] = "Kft",
        fmt1[] = "%3.1f";

    int xmid = nps_graphics_info.xmid;
    int c_height = nps_graphics_info.c_height;
    int c_width = nps_graphics_info.c_width;

    Nps_plot_axis *x_axis = info->axis + NPS_axis_X;
    Nps_plot_axis *y_axis = info->axis + NPS_axis_Y;

    int cnt, x1, x2, x3, y1, y2, y3;
```

```
double step, scale;
char work[24];

/*-----------------------------------------------------------------
 */
if (time.view.bottom == 0)  points_init();

cleardevice();
setcolor(7);

settextstyle(SMALL_FONT, HORIZ_DIR, nps_graphics_info.font_size);

/*-----------------------------------------------------------------
 *  Draw the title and mag/time axis.
 */
settextjustify(CENTER_TEXT,CENTER_TEXT);

outtextxy(xmid, c_height, info->title);

line(axis.x1,axis.y1, axis.x1,axis.y2);
line(axis.x1,axis.y2, axis.x2,axis.y2);

outtextxy(xmid, axis.y2 + c_height * 3, "time(sec)" );

/*-----------------------------------------------------------------
 *  Draw horizontal ticks (time).
 *
 *  The range tells me how much to show on one axis, and is divided into
 * six sections.  "Step" is the value of one section.  Each section is hard
 * coded to be 100 pixels.
 */
settextjustify(CENTER_TEXT,CENTER_TEXT);

step = x_axis->range / 6.0;
y1 = axis.y2 - 5;  y2 = axis.y2 + 5;  y3 = y2 + c_height/2;

for (cnt = 1;  cnt <= 6;  cnt++)
{
     x1 = axis.x1 + STEP_PIXELS * cnt;
     line(x1,y1, x1,y2);
     sprintf(work, fmt1, step * cnt + x_axis->origin);
     outtextxy(x1, y3, work);
}

/*-----------------------------------------------------------------
 *  Draw vertical ticks (mag).
 *
 *  The range tells me how much to show on one axis, and is divided into
 * four sections, three of which are labeled.  "Step" is the value of one
 * section.  Each section is hard coded to be 100 pixels.
```

```
    */
    settextstyle(SMALL_FONT, VERT_DIR, nps_graphics_info.font_size);

    step = y_axis->range / 4.0;

    x1 = axis.x1;  x2 = axis.x1 + 5;  x3 = x1 - c_height;

    if (step < 1.0)        { mag_label[0] = ' ';  scale = 1000.0; }
    else           { mag_label[0] = 'K';  scale = 1.0; }
    outtextxy(x3, axis.y1 + c_width * 10, mag_label);

    for (cnt = 1;  cnt <= 3;  cnt++)
    {
        y1 = axis.y2 - STEP_PIXELS * cnt;
        line(x1,y1, x2,y1);
        sprintf(work, fmt1, (step * cnt + y_axis->origin) * scale);
        outtextxy(x3, y1, work);
    }

    settextstyle(SMALL_FONT, HORIZ_DIR, nps_graphics_info.font_size);

    /*-------------------------------------------------------------------
     *  Draw number box, time labels, and menu bar.
     */
    settextjustify(LEFT_TEXT,CENTER_TEXT);

    x1 = mag_time.view.left - c_width * 6;
    mag_title[4] = (diff_type == NPS_diff_pos ? 'r' : 'v');
    outtextxy(x1, mag_time.view.top + mag_time.line1_y, mag_title);
    outtextxy(x1, mag_time.view.top + mag_time.line2_y, "time");

    x1 = time.view.left - c_width * 8;
    outtextxy(x1, time.view.top + time.line1_y, info->target_title);
    outtextxy(x1, time.view.top + time.line2_y, info->chaser_title);

    if (menu != NULL)  nps_menuplot__init(menu);
}

/*=====================================================================
=====================
 *-------------------------------------------------------------------*/
void nps_diffplot__time(double t1, double t2)
{
    int c_width = nps_graphics_info.c_width;
    char work[20];

    /*-------------------------------------------------------------------
     */
    nps_plot__clear_view(&time.view, NO);
    settextjustify(LEFT_TEXT,CENTER_TEXT);
```

```
    t1 += 86400.0;
    time_dbl_to_string(&t1, work);
    outtextxy(c_width/2, time.line1_y, work);

    t2 += 86400.0;
    time_dbl_to_string(&t2, work);
    outtextxy(c_width/2, time.line2_y, work);

    nps_plot__main_view();
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
void nps_diffplot__values(double diff, double dtime, int diff_type)
{
    static char fmt_time[] = "%9.0f sec", fmt_mag[] = "%9.3f Kft/s";

    int c_width = nps_graphics_info.c_width;

    char work[32];

    /*-------------------------------------------------------------
     */
    nps_plot__clear_view(&mag_time.view, NO);

    settextjustify(LEFT_TEXT,CENTER_TEXT);
    setcolor(7);

    if (diff < 1.0)         { fmt_mag[6] = ' '; diff *= 1000; }
    else                    { fmt_mag[6] = 'K'; }
    fmt_mag[9] = (diff_type == NPS_diff_pos ? 0 : '/');

    sprintf(work, fmt_mag, diff);
    outtextxy(c_width/2, mag_time.line1_y, work);

    sprintf(work, fmt_time, dtime);
    outtextxy(c_width/2, mag_time.line2_y, work);

    nps_plot__main_view();
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
void nps_diffplot__point(Nps_plot_info *info, double diff, double dtime,
                         int diff_type, double t1, double t2)
{
    int xpoint, ypoint;
    Nps_plot_axis *x_axis = info->axis + NPS_axis_X;
```

```
    Nps_plot_axis *y_axis = info->axis + NPS_axis_Y;

    /*-------------------------------------------------------------
     */
    diff *= KILOM_TO_KILOFT;

    nps_diffplot__time(t1, t2);

    nps_diffplot__values(diff, dtime, diff_type);

    /*-------------------------------------------------------------
     */
    xpoint = (int)((dtime - x_axis->origin) * (double)TIME_PIXELS / x_axis->range) + axis.x1;
    ypoint = (int)(-(diff - y_axis->origin) * (double)MAG_PIXELS  / y_axis->range) + axis.y2;

    if (xpoint >= 0  &&  xpoint < nps_graphics_info.xmax
     && ypoint >= 0  &&  ypoint < nps_graphics_info.ymax)  putpixel(xpoint, ypoint, 7);
}

/*==========================================================
==========================
 *-------------------------------------------------------------*/
void nps_diffplot__point_fast(Nps_plot_info *info, double diff, double dtime)
{
    int xpoint, ypoint;
    Nps_plot_axis *x_axis = info->axis + NPS_axis_X;
    Nps_plot_axis *y_axis = info->axis + NPS_axis_Y;

    /*-------------------------------------------------------------
     */
    settextjustify(CENTER_TEXT,CENTER_TEXT);
    setcolor(7);

    diff *= KILOM_TO_KILOFT;

    xpoint = (int)((dtime - x_axis->origin) * (double)TIME_PIXELS / x_axis->range) + axis.x1;
    ypoint = (int)(-(diff - y_axis->origin) * (double)MAG_PIXELS  / y_axis->range) + axis.y2;

    if (xpoint >= 0  &&  xpoint < nps_graphics_info.xmax
     && ypoint >= 0  &&  ypoint < nps_graphics_info.ymax)  putpixel(xpoint, ypoint, 7);
}
```

```
/*=============================================================
==========================
 *-------------------------------------------------------------*/
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <graphics.h>

#include "nps.h"

/*=============================================================
==========================
 *  This function recieves an initial position and velocity vector for a body
 * orbiting the earth and produces those classical elements needed by the f and
 * g functions for orbit prediction.
 *-------------------------------------------------------------*/
void getclas(Vector *pos, Vector *vel, Orbmech_classicals *class, Perturbations *P)
{
   Vector angmom, evec, exr, nvec, nxe;
   double mu, r, v, E, Eo, a, h, r_mult, v_mult, e, cosnu, sinnu, nu;
   double n, sinom, cosom;

   mu = P->Gravity.GMD.planet_mu;

   r = vector_magnitude(pos);
   v = vector_magnitude(vel);
   E = v*v/2 - mu/r;              //energy not eccentric anomaly
   a = -mu/E/2;

   vector_cross(pos, vel, &angmom);
   h = vector_magnitude(&angmom);

   r_mult = v*v/mu - 1/r;
   v_mult = vector_dot(pos, vel)/mu;

   evec.x =  r_mult*pos->x - v_mult*vel->x;
   evec.y =  r_mult*pos->y - v_mult*vel->y;
   evec.z =  r_mult*pos->z - v_mult*vel->z;

   e = vector_magnitude(&evec);

   cosnu = vector_dot(pos, &evec)/r/e;

   vector_cross(&evec, pos, &exr);
   sinnu = vector_dot(&angmom, &exr)/h/e/r;
   nu = atan2(sinnu,cosnu);

   nvec.x = -angmom.y;  nvec.y = angmom.x;  nvec.z = 0.0;
   vector_cross(&nvec, &evec, &nxe);
```

```
   n = vector_magnitude(&nvec);
   sinom = vector_dot(&angmom,&nxe)/h/e/n;
   cosom = vector_dot(&nvec,&evec)/e/n;

   class->a  = a;
   class->e  = e;
   class->Eo = Eo = 2*atan(tan(nu/2)*sqrt((1-e)/(1+e)));
   class->Mo = Eo - e*sin(Eo);
   class->Om = atan2(angmom.x,-angmom.y);
   class->i  = acos(angmom.z/h);
   class->om = atan2(sinom,cosom);
   class->n  = sqrt(mu/a/a/a);
}

/*=============================================================
============================
 *  This function uses the Laguere-Conway algorithm for solving Kepler's
 * equation M = E - e*sin(E), for the eccentric anomaly E, given the mean
 * anomaly M.
 *-------------------------------------------------------------*/
void laguere_conway(Vector *MeE)
{
   double M, e, E, f, fprm, f2p, term, a, b, dE;

   M = MeE->x;
   e = MeE->y;

   E = M;
   f = 1.0;
   while (f > 1.0e-15 )
   {
        f = E - e*sin(E) - M;
        fprm = 1 - e*cos(E);
        f2p = e*sin(E);
        term = 2*sqrt(4*(fprm*fprm) - 5*f*f2p);
        a = fprm + term;
        b = fprm - term;
        if (fabs(a) > fabs(b))
           dE = 5*f/a;
        else
           dE = 5*f/b;
        E = E - dE;
   }
   MeE->z = E;
}


/*=============================================================
============================
 *  This function recieves the a, e, and Eo elements of the classical element
```

```
 * set for an eccentric (central force) orbit, the reference position and
 * velocity from which they were developed, and a time difference from the epoch
 * of pos, vel, and produces a new position with the f and g functions.
 *-----------------------------------------------------------------*/
void f_and_g(Vector *pos, Vector *vel, Vector *pos_new, Vector *vel_new,
             Orbmech_classicals *class, double t_dif)
{
   double  ro, a, e, Eo, Mo, E, M, n, f, g, r, fdot, gdot;
   Vector MeE;

   ro = vector_magnitude(pos);
   a  = class->a;
   e  = class->e;
   Eo = class->Eo;
   Mo = class->Mo;
   n  = class->n;

   M = Mo + t_dif*n;

   MeE.x = M;  MeE.y = e;
   laguere_conway(&MeE);
   E = MeE.z;

   r = a*(1-e*cos(E));

   f = a*(cos(E - Eo) - 1)/ro + 1;
   g = (sin(E-Eo) - e*sin(E) + e*sin(Eo))/n;
   fdot = -sin(E-Eo)*n*a*a/ro/r;
   gdot = a*(cos(E-Eo) - 1)/r + 1;

   pos_new->x = f*pos->x + g*vel->x;
   pos_new->y = f*pos->y + g*vel->y;
   pos_new->z = f*pos->z + g*vel->z;
   vel_new->x = fdot*pos->x + gdot*vel->x;
   vel_new->y = fdot*pos->y + gdot*vel->y;
   vel_new->z = fdot*pos->z + gdot*vel->z;
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <graphics.h>

#include "nps.h"

#define OFF 0
#define ON 1
#define TINY 1.0e-20
#define YES 1
#define NO 0


void lu_bksb6(double a[6][6], int n, int indx[6], double b[6]);
void buffer_raw_orb_gps(State_vector *nps_tans, double tans_buf[4][60],
                        int tans_mat_dat[2]);
int lu_dcmp6(double a[6][6], int n, int indx[6]);


/*=============================================================
=======================
* int ls_filter(State_vector *tans_in, State_vector *tans_out,
*          double tans_buf, int fbi, int point_num, Perturbations *P)
* This filter, created by Lou Zyla is designed to accept buffered tans
* information and produce a smoothed (in a least squares sense) State_vector
* for the most recent point.
*-------------------------------------------------------------*/
int ls_filter(State_vector *tans_in, State_vector *tans_out,
              double tans_buf[4][60], int fbi,
              int point_num, Perturbations *P)

{
        double xha[6], xtr[6][7], tran_mat[3][6], resid_accum[6];
        double ptpmat[6][6], rmat[6][6], rsd[6], tnow, tnext, delta;
        double sum, resx, resy, resz, xtrhold[6][7], clasmat[5][7];
        double time, t_fg_ref;
        Orbmech_classicals classics;
        Vector pos_o, vel_o, pos_new, vel_new;
        int i, j, k, ii, jj, iter, index[6], sing, count, num_points;
        int drag_holder, zonal_holder, tesseral_holder;

        double time_step = 10.0;
        /*-------------------------------------------------
         */

        drag_holder = P->Drag.mode;
        zonal_holder = P->Gravity.number_zonal;
        tesseral_holder = P->Gravity.number_tesseral;
```

```
time_step = 10.0;

*tans_out = *tans_in;
for( iter=1; iter<=3; iter++ )
 {
   xha[0] = tans_out->pos.x;
   xha[1] = tans_out->pos.y;
   xha[2] = tans_out->pos.z;
   xha[3] = tans_out->vel.x;
   xha[4] = tans_out->vel.y;
   xha[5] = tans_out->vel.z;

/*-------------------------------------------------------------
 */
   for( i=0; i<6; i++ )
    {
        for( j=0; j<7; j++ )
         {
           xtr[i][j] = xtrhold[i][j] = xha[i];       //xtr is a 6x7 matrix
         }
    }
/*-------------------------------------------------------------
 */
   for( i=0; i<3; i++ )
    {
        xtr[i][i] = xtrhold[i][i] = xtr[i][i] + 5.0e-3;
        xtr[i+3][i+3] = xtrhold[i+3][i+3] = xtr[i+3][i+3] + 0.5e-3;
    }
/*-------------------------------------------------------------
 */
   for( i=0; i<6; i++ )
    {
        for( j=0; j<6; j++ )
         {
           ptpmat[i][j] = 0.0;
         }
        resid_accum[i] = 0.0;
    }
/*-------------------------------------------------------------
 */
   for( i=0; i<3; i++ )
    {
        ptpmat[i][i] = 1.0;
        resid_accum[i] = tans_buf[i][point_num] - xha[i];
    }
   tnow = t_fg_ref = tans_out->time;
/*-------------------------------------------------------------
 */
   if( fbi == YES )
    {
```

```
        num_points = 60;
    }
    else
    {
        num_points = point_num + 1;
    }

    i = point_num-1;
    if( i<0 ) i = 59;

    for( k=0; k<7; k++ )
    {
        pos_o.x = xtrhold[0][k];
        pos_o.y = xtrhold[1][k];
        pos_o.z = xtrhold[2][k];
        vel_o.x = xtrhold[3][k];
        vel_o.y = xtrhold[4][k];
        vel_o.z = xtrhold[5][k];

        getclas(&pos_o, &vel_o, &classics, P);

        clasmat[0][k] = classics.a;
        clasmat[1][k] = classics.e;
        clasmat[2][k] = classics.Eo;
        clasmat[3][k] = classics.Mo;
        clasmat[4][k] = classics.n;
    }

    for( count=1; count<num_points; count++ )
    {

        tnext = tans_buf[3][i];
        pos_o.x = xha[0];
        pos_o.y = xha[1];
        pos_o.z = xha[2];
        vel_o.x = xha[3];
        vel_o.y = xha[4];
        vel_o.z = xha[5];
        P->Drag.mode = ON;
        P->Gravity.number_zonal = 4;
        P->Gravity.number_tesseral = 4;

        delta = tnext - tnow;

        bgprop(tnow, &pos_o, &vel_o, P, delta, time_step,
                &pos_o, &vel_o);

        xha[0] = pos_o.x;
        xha[1] = pos_o.y;
        xha[2] = pos_o.z;
```

```
xha[3] = vel_o.x;
xha[4] = vel_o.y;
xha[5] = vel_o.z;
/*-------------------------------------------------
 */
time = tnext - t_fg_ref;

for( k=0; k<7; k++ )
 {
   pos_o.x = xtrhold[0][k];
   pos_o.y = xtrhold[1][k];
   pos_o.z = xtrhold[2][k];
   vel_o.x = xtrhold[3][k];
   vel_o.y = xtrhold[4][k];
   vel_o.z = xtrhold[5][k];

   classics.a  = clasmat[0][k];
   classics.e  = clasmat[1][k];
   classics.Eo = clasmat[2][k];
   classics.Mo = clasmat[3][k];
   classics.n  = clasmat[4][k];

   f_and_g(&pos_o,&vel_o,&pos_new,&vel_new,&classics,time);

   xtr[0][k] = pos_new.x;
   xtr[1][k] = pos_new.y;
   xtr[2][k] = pos_new.z;
   xtr[3][k] = vel_new.x;
   xtr[4][k] = vel_new.y;
   xtr[5][k] = vel_new.z;
 }
/*-------------------------------------------------
 */
tnow = tnext;

for( ii=0; ii<3; ii++ )
 {
   for( jj=0; jj<3; jj++ )
    {
        tran_mat[ii][jj] = (xtr[ii][jj]-xtr[ii][6])/5.0e-3;
        tran_mat[ii][jj+3] = (xtr[ii][jj+3]-xtr[ii][6])/0.5e-3;
    }
 }
/*-------------------------------------------------
 */
for( ii=0; ii<6; ii++ )
 {
   for( j=0; j<6; j++ )
    {
        sum = 0.0;
```

```
                for( k=0; k<3; k++ )
                 {
                   sum = sum + tran_mat[k][ii]*tran_mat[k][j];
                 }
                ptpmat[ii][j] = ptpmat[ii][j] + sum;
              }
            }
            /*-----------------------------------------------
             */
            resx = tans_buf[0][i] - xha[0];
            resy = tans_buf[1][i] - xha[1];
            resz = tans_buf[2][i] - xha[2];
            /*-----------------------------------------------
             */
            for( k=0; k<6; k++ )
            {
              resid_accum[k] = resid_accum[k] + resx*tran_mat[0][k] +
                               resy*tran_mat[1][k] + resz*tran_mat[2][k];
            }

            i--;
            if( i<0 ) i = 59;

          }       //for( count=0; count<num_points; count++ )
          /*-----------------------------------------------------
           */
          for( i=0; i<6; i++ )
          {
              rsd[i] = resid_accum[i];
              for( j=0; j<6; j++ )
              {
                rmat[i][j] = ptpmat[i][j];
              }
          }
          /*-----------------------------------------------------
           */

          sing = lu_dcmp6(rmat, 6, index);
          lu_bksb6(rmat, 6, index, rsd);
          /*-------------------------- ------------------------
           */
          tans_out->pos.x = tans_out->pos.x + rsd[0];
          tans_out->pos.y = tans_out->pos.y + rsd[1];
          tans_out->pos.z = tans_out->pos.z + rsd[2];
          tans_out->vel.x = tans_out->vel.x + rsd[3];
          tans_out->vel.y = tans_out->vel.y + rsd[4];
          tans_out->vel.z = tans_out->vel.z + rsd[5];

        }   //   for( iter=1; iter<=3; iter++ )
```

```
        P->Drag.mode = drag_holder;
        P->Gravity.number_zonal = zonal_holder;
        P->Gravity.number_tesseral = tesseral_holder;

        return(sing);
}
/*=============================================================
=====================
* int lu_dcmp6(a, n, indx, sing)...This function is the L-U decomposition
* from Numerical Recipies.
*-------------------------------------------------------------*/
int lu_dcmp6(double a[6][6], int n, int indx[6])

{
        double big, dum, sum, temp, vv[6];

        int i, imax, j, k, sing;

        sing = NO;
        for( i=0; i<n; i++ )
         {
           big = 0.0;
           for( j=0; j<n; j++ )
            {
                if((temp = fabs(a[i][j])) > big )  big = temp;
            }
           if( big == 0.0 )  sing = YES;
           vv[i] = 1.0/big;
         }
        for( j=0; j<n; j++ )
         {
           for( i=0; i<j; i++ )
            {
                sum = a[i][j];
                for( k=0; k<i; k++ )
                 {
                   sum -= a[i][k]*a[k][j];
                 }
                a[i][j] = sum;
            }
           big = 0.0;
           for( i=j; i<n; i++ )
            {
                sum = a[i][j];
                for( k=0; k<j; k++ )
                 {
                   sum -= a[i][k]*a[k][j];
                 }
                a[i][j] = sum;
                if( (dum = vv[i]*fabs(sum)) >= big)
```

```
                {
                  big = dum;  imax = i;
                }
            }
            if( j != imax )
            {
                for( k=0; k<n; k++ )
                {
                  dum = a[imax][k];
                  a[imax][k] = a[j][k];
                  a[j][k] = dum;
                }
                vv[imax] = vv[j];
            }
            indx[j] = imax;
            if( a[j][j] == 0.0 )  a[j][j] = TINY;
            if( j != n-1 )
            {
                dum = 1.0/a[j][j];
                for( i=j+1; i<n; i++ )  a[i][j] *= dum;
            }
        }
        return(sing);
}
/*=============================================================
=======================
* lu_bksb6(double a[6][6], int n, int indx[6], double b[6])
* this function performs the back substitution after a decomposition.
* It is from Numerical Recipies.
*-------------------------------------------------------------*/
void lu_bksb6(double a[6][6], int n, int indx[6], double b[6])

{

        float sum;
        int i, ip, j, ii=0;


        for( i=0; i<n; i++ )
        {
          ip = indx[i];
          sum = b[ip];
          b[ip] = b[i];
          if (ii)
          {
              for( j=ii; j<=i-1; j++ )
              {
                sum -= a[i][j]*b[j];
              }
          }
```

```
            else
             {
                 if (sum) ii = i;
             }
            b[i] = sum;
          }
         for( i=n-1; i>=0; i-- )
          {
            sum = b[i];
            for ( j=i+1; j<n; j++ )
             {
                 sum -= a[i][j]*b[j];
             }
            b[i] = sum/a[i][i];
          }
}
#ifdef OLD_FILTER
/*=============================================================
=======================
* buffer_raw_tans(State_vector *nps_tans, double tans_buf[4][60],
*                 int tans_mat_dat[2])
* This function creates a ring buffer for unmodified tans information as it
* arrives in the NPS subroutines.  The buffer is used by LSFILTER.C to smooth
* some of the error due to the SA code out of the tans state vector
* information.
*-------------------------------------------------------------*/
void buffer_raw_orb_gps(State_vector *nps_tans, double tans_buf[4][60],
                        int tans_mat_dat[2])

{
        int tans_wrt;

        tans_wrt = tans_mat_dat[1];

        tans_buf[0][tans_wrt] = nps_tans->pos.x;
        tans_buf[1][tans_wrt] = nps_tans->pos.y;
        tans_buf[2][tans_wrt] = nps_tans->pos.z;
        tans_buf[3][tans_wrt] = nps_tans->time;
        tans_mat_dat[1] = tans_wrt + 1;
        if( tans_mat_dat[1] == 60 )
         {
           tans_mat_dat[1] = 0;
           tans_mat_dat[0] = YES;
         }
}
#endif OLD_FILTER
```

```
/*=========================================================================
=========================
 *-----------------------------------------------------------------------*/
#include <string.h>

#include "nps.h"

/*=========================================================================
=========================
 *-----------------------------------------------------------------------*/
extern Nps_graphics_info nps_graphics_info;

/*=========================================================================
=========================
 *  I use these structures to avoid extra floating-point work during realtime.
 *-----------------------------------------------------------------------*/
static struct                                   // Plot menus
{
   struct viewporttype view;
   int height, line1_y, line2_y;
} menu;

/*=========================================================================
=========================
 *-----------------------------------------------------------------------*/
static void points_init(void)
{
   int xmax = nps_graphics_info.xmax;
   int ymax = nps_graphics_info.ymax;
   int c_height = nps_graphics_info.c_height;

   /*-------------------------------------------------------------------
    */
   menu.view.left   = 0;
   menu.view.right  = xmax - 1;
   menu.view.top    = ymax - 1 - (menu.height = c_height * 3 - c_height/4);
   menu.view.bottom = ymax - 1;
   menu.view.clip   = YES;

   menu.line1_y = c_height - c_height/4;
   menu.line2_y = menu.line1_y + c_height;
}

/*=========================================================================
=========================
 *-----------------------------------------------------------------------*/
void nps_menuplot__init(char **labels)
{
   int box, x, x_delta;
   char *text;
```

```
    /*-------------------------------------------------------------
     */
    if (menu.height == 0)  points_init();

    nps_plot__clear_view(&menu.view, YES);
    settextjustify(CENTER_TEXT,CENTER_TEXT);

    x_delta = (menu.view.right + 1) / 10;

    for (box = 0, x = menu.view.left;
          box < 10;
          box++, labels++, x += x_delta)
    {
          if (box > 0)  line(x, 0, x, menu.height);

          if ((text = *labels) == NULL)  continue;

          outtextxy(x + x_delta/2, menu.line1_y, text);
          text += strlen(text) + 1;
          outtextxy(x + x_delta/2, menu.line2_y, text);
    }

    nps_plot__main_view();
}
```

```
/*=============================================================
==========================
 *-----------------------------------------------------------*/
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <dir.h>
#include <signal.h>

#include <edit_rt.h>
#include <times.h>

#ifdef DEBUG
#include <edit.h>
#endif DEBUG

#include "nps.h"
#include "nps_m50.h"

/*=============================================================
==========================
 *  Only one each of these structures exists.
 *-----------------------------------------------------------*/
Nps_graphics_info nps_graphics_info =
{
   {0}, /* struct viewporttype main_view */
   0,0,0,        /* int xmax, xmid, c_width              */
   0,0,0,        /* int ymax, ymid, c_height             */
   4,            /* int font_size  32x32 pixels          */
   NO, 0         /* int in_graphics_mode, page           */
};

Emm_info nps_emm_info;

/*=============================================================
==========================
 *  This enumeration defines the various sub-plots of this system.  The static
 * variable holds the current sub-plot, and starts as zero (_main_menu).
 *-----------------------------------------------------------*/
enum nps_plots
{
   NPS_main_menu,
   NPS_spas_gps__orb_gps, NPS_spas_gps__orb_ins,
   NPS_orb_gps__orb_ins, NPS_orb_sawtooth,
   NPS_spas_gps__orb_tgt, NPS_tgt_sawtooth,
   NPS_orb_tgt__orb_gps, NPS_orb_tgt__orb_ins,
   NPS_orb_gps_pitch_yaw, NPS_orb_ins_pitch_yaw,
   NPS_orb_ins_spas_gps_pitch_yaw,
```

```
   NPS_edit_menu,
   NPS_max
};

static enum nps_plots nps_sub_plot = NPS_max;

/*=============================================================
============================
 *  The SPAS vector has two possible sources, as does the Orbiter-GPS.  Some
 * of the sources are WGS84 and need conversion to M50, there are currently two
 * routines for conversion.
 *-------------------------------------------------------------*/
#ifdef NPS_GPS
enum spas_sources { SPAS_spas_gps, SPAS_spas_sv, SPAS_src_max };
enum gps_sources { ORB_GPS_m50, ORB_GPS_ecef, ORB_GPS_src_max };

static Nps_state_vector_info spas_gps  = { YES, NO, SPAS_spas_gps, NPS_M50_from_iload };
static Nps_state_vector_info orb_gps   = { YES, NO, ORB_GPS_m50,   NPS_M50_from_iload };
#endif NPS_GPS

static Nps_state_vector_info orb_ins   = { YES, NO };
static Nps_quaternion_info   orb_quat  = { YES, NO };
static Nps_state_vector_info orb_tgt   = { YES, NO };

/*=============================================================
============================
 *  The perturbations structure is declared far because it's rather large.  It
 * is used for both vehicles: orbiter and spas.  Only the ballistic number needs
 * changing from vehicle to vehicle.
 *-------------------------------------------------------------*/
static Perturbations far nps_P;

static double orb_ballistic_number = 64.0;
static double spas_ballistic_number = 64.0;

/*=============================================================
============================
 *-------------------------------------------------------------*/
double nps_newest_time;

Nps_predictor nps_predictor =
{
   NPS_predict_off, 40, 60.0, 8.0, 0.0, &nps_P, &orb_quat,
   { NO, NO, NO }
};

int nps_dap_on;
Nps_rvplot_info *nps_dap_plot;
```

```
/*=============================================================
==========================
 *  These menus are used by the plots.
 *-------------------------------------------------------------*/
static char
   menu_str_sp[]    = "SPACE\0"     "Alt Menus",
   menu_str_esc[]   = "ESC\0"       "prev scrn",

#  ifdef NPS_GPS
   menu_str_f1[]    = "F1\0"        "Spas/O-GPS",
   menu_str_f2[]    = "F2\0"        "Spas/O-INS",
   menu_str_f3[]    = "F3\0"        "O-GPS/INS",
   menu_str_sf3[]   = "Shft-F3\0"   "Saw - ORB",
   menu_str_f4[]    = "F4\0"        "Spas/TGT",
   menu_str_sf4[]   = "Shft-F4\0"   "Saw - TGT",
   menu_str_f5[]    = "F5\0"        "TGT/O-GPS",
#  endif NPS_GPS
   menu_str_f6[]    = "F6\0"        "TGT/O-INS",
   menu_str_f7[]    = "F7\0"        "Singl Pred",
   menu_str_f8[]    = "F8\0"        "Auto  Pred",
   menu_str_sf6[]   = "Shft-F6\0"   "Rbar-Vbar",
   menu_str_sf7[]   = "Shft-F7\0"   "Vbar-Hbar",
   menu_str_sf8[]   = "Shft-F8\0"   "Rbar-Hbar",
   menu_str_sf9[]   = "Shft-F9\0"   "Hbar on/off",
#  ifdef NPS_GPS
   menu_str_f9[]    = "F9\0"        "Toggle",
#  endif NPS_GPS
   menu_str_f10[]   = "F10\0"       "Settings",

   menu_str_af1[]   = "Alt-F1\0"    "Freeze Scrn",
   menu_str_af5[]   = "Alt-F5\0"    "Pitch/Yaw",
   menu_str_af6[]   = "Alt-F6\0"    "Fast p/y",
   menu_str_af7[]   = "Alt-F7\0"    "Wipe one",
   menu_str_af8[]   = "Alt-F8\0"    "Wipe Plots",
   menu_str_af10[]  = "Alt-F10\0"   "Save plots",

#  ifdef NPS_GPS
   menu_str_cf1[]   = "Cntl-F1\0"   "Filt O-GPS",
   menu_str_cf2[]   = "Cntl-F2\0"   "--> off",
#  ifdef NEW_FILTER_SPAS
   menu_str_cf3[]   = "Cntl-F3\0"   "Filt Spas",
   menu_str_cf4[]   = "Cntl-F4\0"   "--> off",
#  endif NEW_FILTER_SPAS
#  endif NPS_GPS
   menu_str_cf6[]   = "Cntl-F6\0"   "RNDV Pred",
   menu_str_cf7[]   = "Cntl-F7\0"   "DAP mode",
   menu_str_cf8[]   = "Cntl-F8\0"   "Clr thrust",
   menu_str_cf9[]   = "Cntl-F9\0"   "Prop all",
```

```
    menu_str_zoomz[] = "Up-Down\0"    "Zoom Z",
    menu_str_zoomx[] = "Left-Right\0" "Zoom X",
    menu_str_panz[]  = "CNTL up-dn\0" "Pan Z",
    menu_str_panx[]  = "CNTL lf-rt\0" "Pan X",
    menu_str_home[]  = "HOME  find\0" " Target",
    menu_str_end[]   = "END   find\0" " Orbiter";

static char *rv_saw_menu1[] =
{
    menu_str_sp,

#   ifdef NPS_GPS
    menu_str_f1, menu_str_f2, menu_str_f3,
    menu_str_f4, menu_str_f5, menu_str_f6,
#   else
    NULL, NULL, NULL,
    NULL, NULL, menu_str_f6,
#   endif NPS_GPS

    menu_str_f7, menu_str_f8,
    menu_str_f10
};

static char *rv_saw_menu2a[] =
{
    menu_str_sp,
    NULL, NULL,
    NULL, NULL,
    menu_str_sf6, menu_str_sf7,
    menu_str_sf8, menu_str_sf9,
    menu_str_f10
};

#ifdef NPS_GPS
static char *rv_saw_menu2b[] =
{
    menu_str_sp, NULL, NULL,
    menu_str_sf3, menu_str_sf4,
    NULL, NULL,
    menu_str_f9, NULL, menu_str_f10
};
#endif NPS_GPS

static char *rv_saw_menu3[] =
{
    menu_str_sp, NULL, menu_str_af5, menu_str_af6, NULL, NULL,
    menu_str_af7, menu_str_af8, NULL, menu_str_af10
};
```

```
static char *rv_saw_menu4[] =
{
   menu_str_sp,

#  ifdef NPS_GPS
   menu_str_cf1, menu_str_cf2,
#  ifdef NEW_FILTER_SPAS
   menu_str_cf3, menu_str_cf4,
#  else
   NULL, NULL,
#  endif NEW_FILTER_SPAS
#  else
   NULL, NULL,
   NULL, NULL,
#  endif NPS_GPS

#  ifdef NPS_RNDV
   menu_str_cf6,
#  else
   NULL,
#  endif NPS_RNDV
   menu_str_cf7, menu_str_cf8, menu_str_cf9,
   menu_str_f10
};

static char *rv_saw_menu5[] =
{
   menu_str_sp, menu_str_esc, NULL,
   menu_str_zoomz, menu_str_zoomx, menu_str_panz, menu_str_panx,
   menu_str_home, menu_str_end,
   menu_str_f10
};

static char **rv_saw_menu_list[] =
{
   rv_saw_menu1, rv_saw_menu2a,
#  ifdef NPS_GPS
   rv_saw_menu2b,
#  endif NPS_GPS
   rv_saw_menu3, rv_saw_menu4, rv_saw_menu5, NULL
};

static char *py_menu[] =
{
   menu_str_esc,

#  ifdef NPS_GPS
   menu_str_f1, menu_str_f2, menu_str_f3,
```

```
    menu_str_f4, menu_str_f5, menu_str_f6,
    NULL, menu_str_f9,
#   else
    NULL, NULL, NULL,
    NULL, NULL, menu_str_f6,
    NULL, NULL,
#   endif NPS_GPS

    menu_str_f10
};

static char
#   ifdef NPS_GPS
    title_str_orb_gps[]  = "Orb-GPS",
    title_str_spas[10]   = "Spas-GPS",
    *title_str_spas_src[] = { "Spas-GPS", "Spas-SV" },
#   endif NPS_GPS
    title_str_orb_ins[]  = "Orb-INS",
    title_str_orb_tgt[]  = "Targ-INS";

static Nps_log_file_header log_file_header = { "NPS Temporary Plot Data\x1a" };

/*==========================================================================
==========================
 *  I have one of these structures per RVbar plot, the names are composed of
 * target first, chaser second.
 *-------------------------------------------------------------------------*/
#ifdef NPS_GPS
static Nps_rvplot_info spas_gps__orb_gps =
{
    {
        NULL, title_str_spas, title_str_orb_gps,
        { { 0.0, 25.6 }, { 0.0, 25.6/32 }, { 0.0, 25.6 } },
        NPS_axis_Z, NPS_axis_X
    },
    NULL, 0, 0,0,NO,
    rv_saw_menu_list, 0, YES, NPS_rvplot_rv,
    "sgpsogps.nps", &log_file_header, {0},
    &spas_gps, &orb_gps, &nps_predictor, 0.0, 0.0
};

static Nps_rvplot_info spas_gps__orb_ins =
{
    {
        NULL, title_str_spas, title_str_orb_ins,
        { { 0.0, 25.6 }, { 0.0, 25.6/32 }, { 0.0, 25.6 } },
        NPS_axis_Z, NPS_axis_X
    },
```

```
    NULL, 0, 0,0,NO,
    rv_saw_menu_list, 0, YES, NPS_rvplot_rv,
    "sgpsoins.nps", &log_file_header, {0},
    &spas_gps, &orb_ins, &nps_predictor, 0.0, 0.0
};

static Nps_rvplot_info orb_gps__orb_ins =
{
    {
        NULL, title_str_orb_gps, title_str_orb_ins,
        { { 0.0, 25.6/32 }, { 0.0, 25.6/32 }, { 0.0, 25.6/32 } },
        NPS_axis_Z, NPS_axis_X
    },
    NULL, 800, 0,0,NO,
    rv_saw_menu_list, 0, YES, NPS_rvplot_rv,
    "ogpsoins.nps", &log_file_header, {0},
    &orb_gps, &orb_ins, &nps_predictor, 0.0, 0.0
};

static Nps_rvplot_info spas_gps__orb_tgt =
{
    {
        NULL, title_str_spas, title_str_orb_tgt,
        { { 0.0, 25.6/32 }, { 0.0, 25.6/32 }, { 0.0, 25.6/32 } },
        NPS_axis_Z, NPS_axis_X
    },
    NULL, 800, 0,0,NO,
    rv_saw_menu_list, 0, YES, NPS_rvplot_rv,
    "sgpsotgt.nps", &log_file_header, {0},
    &spas_gps, &orb_tgt, &nps_predictor, 0.0, 0.0
};

static Nps_rvplot_info orb_tgt__orb_gps =
{
    {
        NULL, title_str_orb_tgt, title_str_orb_gps,
        { { 0.0, 25.6 }, { 0.0, 25.6/32 }, { 0.0, 25.6 } },
        NPS_axis_Z, NPS_axis_X
    },
    NULL, 800, 0,0,NO,
    rv_saw_menu_list, 0, YES, NPS_rvplot_rv,
    "otgtogps.nps", &log_file_header, {0},
    &orb_tgt, &orb_gps, &nps_predictor, 0.0, 0.0
};
#endif NPS_GPS

static Nps_rvplot_info orb_tgt__orb_ins =
{
```

```
    {
        NULL, title_str_orb_tgt, title_str_orb_ins,
        { { 0.0, 25.6 }, { 0.0, 25.6/32 }, { 0.0, 25.6 } },
        NPS_axis_Z, NPS_axis_X
    },
    NULL, 800, 0,0,NO,
    rv_saw_menu_list, 0, YES, NPS_rvplot_rv,
    "otgtoins.nps", &log_file_header, {0},
    &orb_tgt, &orb_ins, &nps_predictor, 0.0, 0.0
};

/*==========================================================================
==========================
 *  I have one of these structures per difference (sawtooth) plot.
 *-------------------------------------------------------------------------*/
#ifdef NPS_GPS
static Nps_diffplot_info orb_sawtooth =
{
    {   /* Nps_plot_info displays[2] */
        {  "GPS/INS Position Difference vs. Time",
           title_str_orb_gps, title_str_orb_ins,
           { { 0.0, 60000.0 }, { 0.0, 4.0 },   { 0.0, 0.0 } },
           NPS_axis_Y, NPS_axis_X },
        {  "GPS/INS Velocity Difference vs. Time",
           title_str_orb_gps, title_str_orb_ins,
           { { 0.0, 60000.0 }, { 0.0, 0.004 }, { 0.0, 0.0 } },
           NPS_axis_Y, NPS_axis_X }
    },
    NULL, 0, 0,0,NO,
    0.0, rv_saw_menu_list, 0, NPS_diff_pos,
    "orb_saw.nps", &log_file_header, {0},
    &orb_gps, &orb_ins
};

static Nps_diffplot_info tgt_sawtooth =
{
    {   /* Nps_plot_info displays[2] */
        {  "SPAS/TGT Position Difference vs. Time",
           title_str_spas, title_str_orb_tgt,
           { { 0.0, 60000.0 }, { 0.0, 4.0 },   { 0.0, 0.0 } },
           NPS_axis_Y, NPS_axis_X },
        {  "SPAS/TGT Velocity Difference vs. Time",
           title_str_spas, title_str_orb_tgt,
           { { 0.0, 60000.0 }, { 0.0, 0.004 }, { 0.0, 0.0 } },
           NPS_axis_Y, NPS_axis_X }
    },
    NULL, 0, 0,0,NO,
    0.0, rv_saw_menu_list, 0, NPS_diff_pos,
```

```
   "tgt_saw.nps", &log_file_header, {0},
   &spas_gps, &orb_tgt
};
#endif NPS_GPS

/*=============================================================
==========================
 *  I have one of these structures per pitch-yaw plot.
 *-------------------------------------------------------------*/
#ifdef NPS_GPS
static Nps_pyplot_info orb_gps_py =
{
   NULL, title_str_spas, title_str_orb_gps,  py_menu
};
#endif NPS_GPS

static Nps_pyplot_info orb_ins_py =
{
   NULL, title_str_orb_tgt, title_str_orb_ins,  py_menu
};

#ifdef NPS_GPS
static Nps_pyplot_info orb_ins_spas_gps_py =
{
   NULL, title_str_spas, title_str_orb_ins, py_menu
};
#endif NPS_GPS

static int nps_py_plot = NPS_orb_ins_pitch_yaw;

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static double prev_show_time, prev_save_time;
static double sample_rate_show = 0.0, sample_rate_save = 10.0;

static double prop_step = 10.0;            // 10 secs
static double prop_max_time = 1800.0;   // 30 minutes
static int prop_chk_input_time = NO;

static int nps_menu_frozen = NO;

static int gdriver = DETECT, gmode;
static int nps_save_floppy = NO;

/*=============================================================
==========================
 *-------------------------------------------------------------*/
```

```
#ifdef NPS_GPS
static Nps_filter_info orb_gps_filter =
{
   NO, NO, NO,                  /* on, auto_rerun, input_saved          */
   9.0, 300.0, 0.0,       /* save_step, rerun_step, next_rerun       */
   {0},          /* input_vector                                */
   0, 0, NO,              /* prev_sample, next_sample, full          */
   {0}                    /* ring_buffer                             */
};

#ifdef NEW_FILTER_SPAS
static Nps_filter_info spas_gps_filter =
{
   NO, NO, NO,                  /* on, auto_rerun, input_saved          */
   9.0, 300.0, 0.0,       /* save_step, rerun_step, next_rerun       */
   {0},          /* input_vector                                */
   0, 0, NO,              /* prev_sample, next_sample, full          */
   {0}                    /* ring_buffer                             */
};
#endif NEW_FILTER_SPAS
#endif NPS_GPS

/*==========================================================================
==========================
 *-------------------------------------------------------------------------*/
#define RING_MAX_DEFAULT 0x8000

int nps_system_init()
{
   static char
        ring_env_name[] = "NPS_RING_MAX",
#       ifdef NPS_GPS
        iload_env_name[] = "NPS_ILOAD",
        iload_file[] = "tans.ild",
#       endif NPS_GPS
        config_file[] = "nps.cfg",
        fmt1[] = "%i%*[^\n]%*1c";

   FILE *config;
   char *filename, *ring_max;
   ushort ring_max_mem;

   ushort cfg_mem1, cfg_mem2, cfg_mem3, cfg_mem3a, cfg_mem4, cfg_mem4a;
   ushort cfg_mem5, cfg_mem6, cfg_emm, cfg_flp, cfg_year;

   /*-----------------------------------------------------------------------
    *  Blatant kludge until we can find the source of overflow errors.
    */
```

```
    signal(SIGFPE, SIG_IGN);

    /*-----------------------------------------------------------------
     */
    if (bgprop_init() == NO)  return NO;

#   ifdef NPS_GPS
    if ((filename = getenv(iload_env_name)) == NULL)  filename = iload_file;

    if (nps_wgs84__read_iload(filename) == NO)
    {
        printf("Can't open iload file <%s>\n", filename);
        return NO;
    }
#   endif NPS_GPS

    /*-----------------------------------------------------------------
     */
    nps_P.Drag.mode = ON;
    nps_P.Drag.solar_flux = 185.9593403793603;
    nps_P.Drag.mean_solar_flux = 178.6447698684861;
    nps_P.Drag.geomagnetic_index_ap = 19.87811961456716;
    nps_P.Drag.ballistic_number = 64.0;

    nps_P.Gravity.number_zonal    = 4;
    nps_P.Gravity.number_tesseral = 4;
    instantiate_gotpot(1, &nps_P);

    /*-----------------------------------------------------------------
     */
    if ((ring_max = getenv(ring_env_name)) != NULL)
    {
        sscanf(ring_max, "%i", &ring_max_mem);
        if (ring_max_mem < 1024)  ring_max_mem = 1024;
    }
    else  ring_max_mem = RING_MAX_DEFAULT;

    cfg_mem1 = cfg_mem2 =  cfg_mem3 = cfg_mem3a =
    cfg_mem4 = cfg_mem4a = cfg_mem5 = cfg_mem6 = 0;
    cfg_flp = NO;
    cfg_year = NO;
    cfg_emm = YES;

    if ((config = fopen(config_file, "rt")) != NULL)
    {
        fscanf(config, fmt1, &cfg_mem1);        /* Spas-gps vs Orb-gps RV */
        fscanf(config, fmt1, &cfg_mem2);        /* Spas-gps vs Orb-ins RV */
        fscanf(config, fmt1, &cfg_mem3);        /* Orb-gps  vs Orb-ins RV */
```

```
        fscanf(config, fmt1, &cfg_mem3a);       /* Orb-gps  vs  Orb-ins  SAW */
        fscanf(config, fmt1, &cfg_mem4);        /* Spas-gps  vs  Targ-ins RV  */
        fscanf(config, fmt1, &cfg_mem4a);       /* Spas-gps  vs  Targ-ins SAW */
        fscanf(config, fmt1, &cfg_mem5);        /* Targ-ins  vs  Orb-gps  RV  */
        fscanf(config, fmt1, &cfg_mem6);        /* Targ-ins  vs  Orb-ins  RV  */
        fscanf(config, fmt1, &cfg_emm);         /* Expanded memory yes=1/no=0 */
        fscanf(config, fmt1, &cfg_flp);         /* Save to floppy  yes=1/no=0 */
        fscanf(config, fmt1, &cfg_year);  /* Year for propagation math  */
        fclose(config);
    }

    if (cfg_emm)  emm_info_construct(&nps_emm_info);
    nps_save_floppy = cfg_flp;
    if (cfg_year)  bgprop_year(cfg_year);

    /*-----------------------------------------------------------------
     */
#   ifdef NPS_GPS
    if (nps_rvplot__data_init(&spas_gps__orb_gps, ring_max_mem, cfg_mem1,  &nps_emm_info) == NO
     || nps_rvplot__data_init(&spas_gps__orb_ins, ring_max_mem, cfg_mem2,  &nps_emm_info) == NO
     || nps_rvplot__data_init(&orb_gps__orb_ins,  ring_max_mem, cfg_mem3,  &nps_emm_info) == NO
     || nps_diffplot__data_init(&orb_sawtooth,    ring_max_mem, cfg_mem3a, &nps_emm_info) == NO
     || nps_rvplot__data_init(&spas_gps__orb_tgt, ring_max_mem, cfg_mem4,  &nps_emm_info) == NO
     || nps_diffplot__data_init(&tgt_sawtooth,    ring_max_mem, cfg_mem4a, &nps_emm_info) == NO
     || nps_rvplot__data_init(&orb_tgt__orb_gps,  ring_max_mem, cfg_mem5,  &nps_emm_info) == NO
     || nps_rvplot__data_init(&orb_tgt__orb_ins,  ring_max_mem, cfg_mem6,  &nps_emm_info) == NO)
#   else
    if (nps_rvplot__data_init(&orb_tgt__orb_ins,  ring_max_mem, cfg_mem6,  &nps_emm_info) == NO)
#   endif NPS_GPS
    {
        printf("Out of memory, set %s env-variable to < %04X\nOr change file %s\n",
            ring_env_name, ring_max_mem, config_file);
        return NO;
    }

    #ifdef DEBUG
    gdriver = edit_integer("Change gdriver ? ", gdriver);
    gmode = edit_integer("Change gmode ? ", gmode);
    #endif DEBUG

    return YES;
}

static void nps_system_save(void)
{
    if (nps_save_floppy == NO  &&  getdisk() < 2)  return;
#   ifdef NPS_GPS
    nps_rvplot__data_save(&spas_gps__orb_ins);
```

```
    nps_rvplot__data_save(&spas_gps__orb_gps);
    nps_rvplot__data_save(&orb_gps__orb_ins);
    nps_rvplot__data_save(&spas_gps__orb_tgt);
    nps_diffplot__data_save(&orb_sawtooth);
    nps_diffplot__data_save(&tgt_sawtooth);
    nps_rvplot__data_save(&orb_tgt__orb_gps);
#   endif NPS_GPS
    nps_rvplot__data_save(&orb_tgt__orb_ins);
}

void nps_system_stop()
{
#   ifdef NPS_GPS
    nps_rvplot__data_stop(&spas_gps__orb_ins);
    nps_rvplot__data_stop(&spas_gps__orb_gps);
    nps_rvplot__data_stop(&orb_gps__orb_ins);
    nps_rvplot__data_stop(&spas_gps__orb_tgt);
    nps_diffplot__data_stop(&orb_sawtooth);
    nps_diffplot__data_stop(&tgt_sawtooth);
    nps_rvplot__data_stop(&orb_tgt__orb_gps);
#   endif NPS_GPS
    nps_rvplot__data_stop(&orb_tgt__orb_ins);
    nps_system_save();


/*=============================================================================
==========================
 *-----------------------------------------------------------------------*/
int nps_display_init(void)
{
    static int first_time = YES;
    static char blank[] = "";
    static char c_test[] = "W";

    /*-----------------------------------------------------------------------
     */
    if (first_time)
    {
        int errorcode;
        #ifdef DEBUG
        int xmax, ymax;
        #endif DEBUG

        /*-------------------------------------------------------------------
         */
        initgraph(&gdriver, &gmode, blank);
        if (gdriver == CGA)
        {
```

```
    gdriver = ATT400;  gmode = ATT400HI;
    closegraph();
    initgraph(&gdriver, &gmode, blank);
  }

  if ((errorcode = graphresult()) != grOk)
  {
    //closegraph();
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key ...");  key_get();
    return NO;
  }

  /*-----------------------------------------------------------------
   */
  getviewsettings(&nps_graphics_info.main_view);

  #ifdef DEBUG
  closegraph();
  xmax = nps_graphics_info.main_view.right + 1;
  nps_graphics_info.main_view.right = edit_integer("xmax ? ", xmax) - 1;
  ymax = nps_graphics_info.main_view.bottom + 1;
  nps_graphics_info.main_view.bottom = edit_integer("ymax ? ", ymax) - 1;
  initgraph(&gdriver, &gmode, blank);
  #endif DEBUG

  nps_graphics_info.xmax = nps_graphics_info.main_view.right + 1;
  nps_graphics_info.xmid = nps_graphics_info.xmax/2;

  nps_graphics_info.ymax = nps_graphics_info.main_view.bottom + 1;
  nps_graphics_info.ymid = nps_graphics_info.ymax/2;

  settextstyle(SMALL_FONT, HORIZ_DIR, nps_graphics_info.font_size);
  nps_graphics_info.c_width = textwidth(c_test);
  nps_graphics_info.c_height = textheight(c_test);

  #ifdef DEBUG
  nps_graphics_info.c_height = (int)((double)nps_graphics_info.c_height
    * (double)nps_graphics_info.ymax / (double)ymax);
  nps_graphics_info.c_width = (int)((double)nps_graphics_info.c_width
    * (double)nps_graphics_info.xmax / (double)xmax);
  #endif DEBUG

  first_time = NO;
  closegraph();
}

/*-----------------------------------------------------------------
```

```
    */
   nps_graphics_info.in_graphics_mode = NO;

   nps_sub_plot = NPS_main_menu;
   nps_menu__display_init();
   return YES;
}

/*=============================================================
==========================
static void nps_display_graphics(int graph_mode)
 *-------------------------------------------------------------*/
void nps_display_graphics(int graph_mode)
{
   if (nps_sub_plot == NPS_edit_menu)  edit_screen_stop();
   if (graph_mode)
   {
        if (nps_graphics_info.in_graphics_mode)  return;
        initgraph(&gdriver, &gmode, "");
        nps_graphics_info.in_graphics_mode = YES;
   }
   else
   {
        if (nps_graphics_info.in_graphics_mode == NO)  return;
        closegraph();
        nps_graphics_info.in_graphics_mode = NO;
   }
}

void nps_display_stop()
{
   nps_display_graphics(NO);
   nps_sub_plot = NPS_max;
   nps_dap_on = NO;
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
int nps_bgprop(State_vector *sv, double new_time, double input_time, int is_orbiter)
{
   double dt = new_time - sv->time;

   if (prop_chk_input_time  &&  new_time - input_time > prop_max_time)  return NO;

   if (dt > prop_max_time  ||  dt < 0.0)  return NO;

   if (dt > 1.0e-5)
```

```
    {
        nps_P.Drag.ballistic_number =
          (is_orbiter ? orb_ballistic_number : spas_ballistic_number);
        bgprop(sv->time, &sv->pos, &sv->vel, &nps_P, dt, prop_step, &sv->pos, &sv->vel);
        sv->time = new_time;
    }
    return YES;
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static void nps_propagate_all(int find_new_time, double new_time)
{
    /*-------------------------------------------------------------
     */
    if (find_new_time)
    {
        new_time = (orb_ins.sv.time > orb_tgt.sv.time ? orb_ins.sv.time : orb_tgt.sv.time);
#       ifdef NPS_GPS
        if (orb_gps.sv.time > new_time)  new_time = orb_gps.sv.time;
        if (spas_gps.sv.time > new_time) new_time = spas_gps.sv.time;
#       endif NPS_GPS
    }
    nps_newest_time = new_time;

#   ifdef NPS_GPS
        if (nps_bgprop(&orb_gps.sv,  new_time, orb_gps.input.time, YES) == NO)
            orb_gps.got_one  = NO;

        if (nps_bgprop(&spas_gps.sv, new_time, spas_gps.input.time, NO)  == NO)
            spas_gps.got_one = NO;
#   endif NPS_GPS

        if (nps_bgprop(&orb_ins.sv,  new_time, orb_ins.input.time, YES) == NO)
            orb_ins.got_one  = NO;

        if (nps_bgprop(&orb_tgt.sv,  new_time, orb_tgt.input.time, NO)  == NO)
            orb_tgt.got_one  = NO;
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static void nps_data_update_all(void)
{
#   ifdef NPS_GPS
    nps_rvplot__data_update(&spas_gps__orb_gps);
```

```
    nps_rvplot__data_update(&spas_gps__orb_ins);
    nps_rvplot__data_update(&spas_gps__orb_tgt);
    nps_diffplot__data_update(&tgt_sawtooth);
    nps_rvplot__data_update(&orb_gps__orb_ins);
    nps_diffplot__data_update(&orb_sawtooth);
    nps_rvplot__data_update(&orb_tgt__orb_gps);
#   endif NPS_GPS
    nps_rvplot__data_update(&orb_tgt__orb_ins);
}

/*==========================================================
==========================
 *-------------------------------------------------------------------*/
void nps_data_update(State_vector *orb_ins_in,
                     Quaternion *orb_quat_in,
                     State_vector *orb_tgt_in,
                     State_vector *orb_gps_in,
                     State_vector *orb_gps_ecef_in,
                     State_vector *spas_sv_in,
                     State_vector *spas_gps_in)
{
    double new_time;
    int show_me, save_me;

#   ifdef NPS_GPS
    /*-----------------------------------------------------------------
     */
    if (orb_gps.source != ORB_GPS_m50)
        orb_gps_in = orb_gps_ecef_in;
    if (orb_gps.fix_time)  orb_gps_in->time += orb_gps.dtime;

    /*-----------------------------------------------------------------
     */
    if (orb_gps.in_use  &&  orb_gps_in->time > orb_gps.input.time)
    {
        if (orb_gps.source != ORB_GPS_m50)
            nps_wgs84_to_m50(orb_gps_in, orb_gps.converter, YES);

        if (orb_gps_filter.on == NO)  orb_gps.sv = *orb_gps_in;
        orb_gps.input =  *orb_gps_in;
        orb_gps.got_one = YES;

        nps_filter__data_update(&orb_gps_filter, orb_gps_in);
    }

    if (orb_gps_filter.on)
    {
        if (orb_gps_filter.auto_rerun
```

```
        && orb_gps_filter.next_rerun <= orb_gps_filter.input_vector.time)
        {
           nps_P.Drag.ballistic_number = orb_ballistic_number;
           nps_filter__run(&orb_gps_filter, &orb_gps.sv, &nps_P);
        }
   }
#  endif NPS_GPS

   /*-----------------------------------------------------------------
    */
#  ifdef NPS_GPS
   if (spas_gps.source != SPAS_spas_gps)  spas_gps_in = spas_sv_in;

   if (spas_gps.fix_time)  spas_gps_in->time += spas_gps.dtime;

   /*-----------------------------------------------------------------
    */
#  ifdef NEW_FILTER_SPAS
   if (spas_gps.in_use  &&  spas_gps_in->time > spas_gps.input.time)
   {
        if (spas_gps.source == SPAS_spas_gps)
           nps_wgs84_to_m50(spas_gps_in, spas_gps.converter, NO);
        else nps_j2000_to_m50(spas_gps_in);

        if (spas_gps_filter.on == NO)  spas_gps.sv = *spas_gps_in;
        spas_gps.input =  *spas_gps_in;
        spas_gps.got_one = YES;

        nps_filter__data_update(&spas_gps_filter, spas_gps_in);
   }

   if (spas_gps_filter.on)
   {
        if (spas_gps_filter.auto_rerun
         && spas_gps_filter.next_rerun <= spas_gps_filter.input_vector.time)
        {
           nps_P.Drag.ballistic_number = orb_ballistic_number;
           nps_filter__run(&spas_gps_filter, &spas_gps.sv, &nps_P);
        }
   }
#  else

   /*-----------------------------------------------------------------
    */
   if (spas_gps.in_use  &&  spas_gps_in->time > spas_gps.input.time)
   {
        if (spas_gps.source == SPAS_spas_gps)
           nps_wgs84_to_m50(spas_gps_in, spas_gps.converter, NO);
```

```
        else nps_j2000_to_m50(spas_gps_in);
        spas_gps.sv = spas_gps.input = *spas_gps_in;
        spas_gps.got_one = YES;
    }
#   endif NEW_FILTER_SPAS
#   endif NPS_GPS

    /*-----------------------------------------------------------------
     */
    if (orb_ins.fix_time)  orb_ins_in->time += orb_ins.dtime;

    if (orb_ins.in_use  &&  orb_ins_in->time > orb_ins.input.time)
    {
        orb_ins.sv = orb_ins.input = *orb_ins_in;
        orb_ins.got_one = YES;
    }

    if (orb_quat.in_use  &&  orb_quat_in->time > orb_quat.input_time)
    {
        orb_quat.quat = *orb_quat_in;
        orb_quat.input_time = orb_quat_in->time;
        orb_quat.got_one = YES;
    }

    if (orb_tgt.in_use  &&  orb_tgt_in->time > orb_tgt.input.time)
    {
        orb_tgt.sv = *orb_tgt_in;
        orb_tgt.input = *orb_tgt_in;
        orb_tgt.got_one = YES;
    }

    /*-----------------------------------------------------------------
     */
    new_time = (orb_ins_in->time > orb_tgt_in->time ? orb_ins_in->time : orb_tgt_in->time);
#   ifdef NPS_GPS
    if (orb_gps_in->time > new_time)  new_time = orb_gps_in->time;
    if (spas_gps_in->time > new_time) new_time = spas_gps_in->time;
#   endif NPS_GPS

    /*-----------------------------------------------------------------
     */
    if (nps_sub_plot < NPS_edit_menu  &&  new_time > prev_show_time + sample_rate_show)
    {
        show_me = YES;
        prev_show_time = new_time;
    }
    else  show_me = NO;
```

```
    if (new_time > prev_save_time + sample_rate_save)
    {
         save_me = YES;
         prev_save_time = new_time;
    }
    else save_me = NO;

    if (save_me == NO && show_me == NO) return;

    /*-----------------------------------------------------------------
     */
    nps_propagate_all(NO, new_time);

    if (save_me) nps_data_update_all();

    if (show_me == NO) return;

    /*-----------------------------------------------------------------
     */
    if (nps_predictor.on && nps_predictor.next_time <= nps_newest_time)
    {
         nps_predictor.on = NPS_predict_now;
         nps_predictor.next_time = nps_newest_time + nps_predictor.delay;
    }

    switch (nps_sub_plot)
    {
         case NPS_main_menu:
            if (nps_menu_frozen == NO) nps_menu__display_update();
            break;

#        ifdef NPS_GPS
         case NPS_spas_gps__orb_gps:
            nps_rvplot__display_update(&spas_gps__orb_gps, spas_gps_in->time, orb_gps_in->time);
            break;

         case NPS_spas_gps__orb_ins:
            nps_rvplot__display_update(&spas_gps__orb_ins, spas_gps_in->time, orb_ins_in->time);
            break;

         case NPS_orb_gps__orb_ins:
            nps_rvplot__display_update(&orb_gps__orb_ins, orb_gps_in->time, orb_ins_in->time);
            break;

         case NPS_orb_sawtooth:
            nps_diffplot__display_update(&orb_sawtooth, orb_gps_in->time, orb_ins_in->time);
            break;
```

```
        case NPS_spas_gps__orb_tgt:
           nps_rvplot__display_update(&spas_gps__orb_tgt, spas_gps_in->time, orb_tgt_in->time);
           break;

        case NPS_tgt_sawtooth:
           nps_diffplot__display_update(&tgt_sawtooth, spas_gps_in->time, orb_tgt_in->time);
           break;

        case NPS_orb_tgt__orb_gps:
           nps_rvplot__display_update(&orb_tgt__orb_gps, orb_tgt_in->time, orb_gps_in->time);
           break;
#       endif NPS_GPS

        case NPS_orb_tgt__orb_ins:
           nps_rvplot__display_update(&orb_tgt__orb_ins, orb_tgt_in->time, orb_ins_in->time);
           break;

#       ifdef NPS_GPS
        case NPS_orb_gps_pitch_yaw:
           if (spas_gps.got_one & orb_gps.got_one & orb_quat.got_one)
                nps_pyplot__display_update(&orb_gps_py, &spas_gps.sv, &orb_gps.sv,
                  &orb_quat.quat, spas_gps_in->time, orb_gps_in->time);
           break;

        case NPS_orb_ins_spas_gps_pitch_yaw:
           if (orb_ins.got_one & spas_gps.got_one & orb_quat.got_one)
                nps_pyplot__display_update(&orb_ins_spas_gps_py, &spas_gps.sv,
                &orb_ins.sv, &orb_quat.quat, spas_gps_in->time, orb_ins_in->time);
           break;
#       endif NPS_GPS

        case NPS_orb_ins_pitch_yaw:
           if (orb_tgt.got_one & orb_ins.got_one & orb_quat.got_one)
                nps_pyplot__display_update(&orb_ins_py, &orb_tgt.sv, &orb_ins.sv,
                  &orb_quat.quat, orb_tgt_in->time, orb_ins_in->time);
           break;

        default: break;
   }

   if (nps_predictor.on == NPS_predict_now)  nps_predictor.on = NPS_predict_on;
}

/*==========================================================================
==========================
 *-------------------------------------------------------------------------*/
void nps_data_flush()
{
```

```
#   ifdef NPS_GPS
    nps_filter__data_flush(&orb_gps_filter);
#   ifdef NEW_FILTER_SPAS
    nps_filter__data_flush(&spas_gps_filter);
#   endif NEW_FILTER_SPAS
#   endif NPS_GPS

#   ifdef NPS_GPS
    spas_gps.input.time =
    orb_gps.input.time =
#   endif NPS_GPS
    orb_ins.input.time  =
    orb_quat.input_time  =
    prev_show_time  =
    prev_save_time  = 0.0;

#   ifdef NPS_GPS
    nps_rvplot__data_flush(&spas_gps__orb_gps);
    nps_rvplot__data_flush(&spas_gps__orb_ins);
    nps_rvplot__data_flush(&orb_gps__orb_ins);
    nps_rvplot__data_flush(&spas_gps__orb_tgt);
    nps_diffplot__data_flush(&orb_sawtooth);
    nps_diffplot__data_flush(&tgt_sawtooth);
    nps_rvplot__data_flush(&orb_tgt__orb_gps);
#   endif NPS_GPS
    nps_rvplot__data_flush(&orb_tgt__orb_ins);
}

/*==========================================================================
==========================
 *-------------------------------------------------------------------------*/
static void nps_data_flush_one(int plot)
{
    switch (plot)
    {
#       ifdef NPS_GPS
        case NPS_spas_gps__orb_gps:
            nps_rvplot__data_flush(&spas_gps__orb_gps);
            break;
        case NPS_spas_gps__orb_ins:
            nps_rvplot__data_flush(&spas_gps__orb_ins);
            break;
        case NPS_orb_gps__orb_ins:
            nps_rvplot__data_flush(&orb_gps__orb_ins);
            break;
        case NPS_orb_sawtooth:
            nps_diffplot__data_flush(&orb_sawtooth);
            break;
```

```
        case NPS_spas_gps__orb_tgt:
            nps_rvplot__data_flush(&spas_gps__orb_tgt);
            break;
        case NPS_tgt_sawtooth:
            nps_diffplot__data_flush(&tgt_sawtooth);
            break;
        case NPS_orb_tgt__orb_gps:
            nps_rvplot__data_flush(&orb_tgt__orb_gps);
            break;
#       endif NPS_GPS
        case NPS_orb_tgt__orb_ins:
            nps_rvplot__data_flush(&orb_tgt__orb_ins);
            break;
    }
}

/*=============================================================
============================
 * Returns:
 * 0    Didn't use the key
 * 1    Used the key
 * 2    Used the key, My subsystem is done
 *-------------------------------------------------------------*/
int nps_key_handler(int key)
{
    int rc;

    /*-------------------------------------------------------------
     */
    if (nps_dap_on)
    {
        if (nps_dap__key_handler(key) == 2)  nps_dap_on = NO;
        return 1;
    }

    /*-------------------------------------------------------------
     */
    switch (nps_sub_plot)
    {
        case NPS_main_menu:
            rc = NO;
            break;
#       ifdef NPS_GPS
        case NPS_spas_gps__orb_gps:
            rc = nps_rvplot__key_handler(&spas_gps__orb_gps, key);
            break;
        case NPS_spas_gps__orb_ins:
            rc = nps_rvplot__key_handler(&spas_gps__orb_ins, key);
```

```
            break;
        case NPS_orb_gps__orb_ins:
            rc = nps_rvplot__key_handler(&orb_gps__orb_ins, key);
            break;
        case NPS_orb_sawtooth:
            rc = nps_diffplot__key_handler(&orb_sawtooth, key);
            break;
        case NPS_spas_gps__orb_tgt:
            rc = nps_rvplot__key_handler(&spas_gps__orb_tgt, key);
            break;
        case NPS_tgt_sawtooth:
            rc = nps_diffplot__key_handler(&tgt_sawtooth, key);
            break;
        case NPS_orb_tgt__orb_gps:
            rc = nps_rvplot__key_handler(&orb_tgt__orb_gps, key);
            break;
        case NPS_orb_gps_pitch_yaw:
            rc = nps_pyplot__key_handler(&orb_gps_py, key);
            break;
        case NPS_orb_ins_spas_gps_pitch_yaw:
            rc = nps_pyplot__key_handler(&orb_ins_spas_gps_py, key);
            break;
#       endif NPS_GPS
        case NPS_orb_tgt__orb_ins:
            rc = nps_rvplot__key_handler(&orb_tgt__orb_ins, key);
            break;
        case NPS_orb_ins_pitch_yaw:
            rc = nps_pyplot__key_handler(&orb_ins_py, key);
            break;
        case NPS_edit_menu:
            rc = nps_edit_menu_key_handler(key);
            break;
        default: rc = NO;  break;
    }

    /*-----------------------------------------------------------------
     */
    if (rc > 0)
    {
        if (rc == 2)
        {
            nps_display_graphics(NO);
            nps_sub_plot = NPS_main_menu;
            nps_menu__display_init();
        }
        return 1;
    }
```

```
    return nps_menu__key_handler(key);
}

/*=============================================================
============================
 * Returns:
 * 0    Didn't use the key
 * 1    Used the key
 * 2    Used the key, My subsystem is done
 *-------------------------------------------------------------*/
static int nps_menu__key_handler(int key)
{
    nps_menu_frozen = NO;

    switch (key)
    {
        case KEY_esc:
            nps_display_graphics(NO);
            nps_sub_plot = NPS_max;
            return 2;

#       ifdef NPS_GPS
        case KEY_f1:
            if (nps_sub_plot == NPS_spas_gps__orb_gps)  break;
            nps_display_graphics(YES);
            nps_sub_plot = NPS_spas_gps__orb_gps;
            nps_rvplot__display_init(&spas_gps__orb_gps);
            break;

        case KEY_f2:
            if (nps_sub_plot == NPS_spas_gps__orb_ins)  break;
            nps_display_graphics(YES);
            nps_sub_plot = NPS_spas_gps__orb_ins;
            nps_rvplot__display_init(&spas_gps__orb_ins);
            break;

        case KEY_f3:
            if (nps_sub_plot == NPS_orb_gps__orb_ins)  break;
            nps_display_graphics(YES);
            nps_sub_plot = NPS_orb_gps__orb_ins;
            nps_rvplot__display_init(&orb_gps__orb_ins);
            break;

        case KEY_shft_f3:
            if (nps_sub_plot == NPS_orb_sawtooth)  break;
            nps_display_graphics(YES);
            nps_sub_plot = NPS_orb_sawtooth;
            nps_diffplot__display_init(&orb_sawtooth);
```

```
            break;

        case KEY_f4:
            if (nps_sub_plot == NPS_spas_gps__orb_tgt)  break;
            nps_display_graphics(YES);
            nps_sub_plot = NPS_spas_gps__orb_tgt;
            nps_rvplot__display_init(&spas_gps__orb_tgt);
            break;

        case KEY_shft_f4:
            if (nps_sub_plot == NPS_tgt_sawtooth)  break;
            nps_display_graphics(YES);
            nps_sub_plot = NPS_tgt_sawtooth;
            nps_diffplot__display_init(&tgt_sawtooth);
            break;

        case KEY_f5:
            if (nps_sub_plot == NPS_orb_tgt__orb_gps)  break;
            nps_display_graphics(YES);
            nps_sub_plot = NPS_orb_tgt__orb_gps;
            nps_rvplot__display_init(&orb_tgt__orb_gps);
            break;
#       endif NPS_GPS

        case KEY_f6:
            if (nps_sub_plot == NPS_orb_tgt__orb_ins)  break;
            nps_display_graphics(YES);
            nps_sub_plot = NPS_orb_tgt__orb_ins;
            nps_rvplot__display_init(&orb_tgt__orb_ins);
            break;

        case KEY_f7:
            if (nps_sub_plot == NPS_orb_tgt__orb_ins)
                nps_rvplot__display_future(&orb_tgt__orb_ins);
#           ifdef NPS_GPS
            else if (nps_sub_plot == NPS_spas_gps__orb_gps)
                nps_rvplot__display_future(&spas_gps__orb_gps);
            else if (nps_sub_plot == NPS_spas_gps__orb_ins)
                nps_rvplot__display_future(&spas_gps__orb_ins);
            else if (nps_sub_plot == NPS_orb_tgt__orb_gps)
                nps_rvplot__display_future(&orb_tgt__orb_gps);
#           endif NPS_GPS
            break;

        case KEY_f8:
            nps_predictor.on = (nps_predictor.on ? NPS_predict_off : NPS_predict_on);
            nps_predictor.next_time = 0.0;
            break;
```

```
#           ifdef NPS_GPS
          case KEY_f9:
             if (nps_sub_plot != NPS_orb_ins_pitch_yaw
              && nps_sub_plot != NPS_orb_gps_pitch_yaw
              && nps_sub_plot != NPS_orb_ins_spas_gps_pitch_yaw)  break;

             if (nps_py_plot == NPS_orb_gps_pitch_yaw)
                  nps_py_plot = NPS_orb_ins_spas_gps_pitch_yaw;
             else if (nps_py_plot == NPS_orb_ins_spas_gps_pitch_yaw)
                  nps_py_plot = NPS_orb_ins_pitch_yaw;
             else nps_py_plot = NPS_orb_gps_pitch_yaw;
#           endif NPS_GPS

          case KEY_alt_f5:
             if (nps_sub_plot == nps_py_plot)  break;
             nps_display_graphics(YES);
             nps_sub_plot = nps_py_plot;
#           ifdef NPS_GPS
             nps_pyplot__display_init(
              (nps_py_plot == NPS_orb_gps_pitch_yaw
                  ? &orb_gps_py
                  : (nps_py_plot == NPS_orb_ins_pitch_yaw
                    ? &orb_ins_py
                    : &orb_ins_spas_gps_py)));
#           else
             nps_pyplot__display_init(&orb_ins_py);
#           endif NPS_GPS
             break;

          case KEY_alt_f6:
             if (nps_sub_plot == NPS_orb_tgt__orb_ins)
                  nps_pyplot__display_stats(&orb_ins_py, &orb_tgt.sv,
                    &orb_ins.sv, &orb_quat.quat);

#           ifdef NPS_GPS
             else if (nps_sub_plot == NPS_spas_gps__orb_gps)
                  nps_pyplot__display_stats(&orb_gps_py, &spas_gps.sv,
                    &orb_gps.sv, &orb_quat.quat);

             else if (nps_sub_plot == NPS_spas_gps__orb_ins)
                  nps_pyplot__display_stats(&orb_ins_py, &spas_gps.sv,
                    &orb_ins.sv, &orb_quat.quat);

             else if (nps_sub_plot == NPS_orb_tgt__orb_gps)
                  nps_pyplot__display_stats(&orb_gps_py, &orb_tgt.sv,
                    &orb_gps.sv, &orb_quat.quat);
#           endif NPS_GPS
```

```
            break;

         case KEY_alt_f7:
            nps_data_flush_one(nps_sub_plot);
            break;

         case KEY_alt_f8:
            nps_data_flush();
            break;

         case KEY_alt_f10:
            nps_system_save();
            break;

#        ifdef NPS_GPS
         case KEY_cntl_f1:
            if (orb_gps_filter.ring_full == NO)  break;
            orb_gps_filter.on = YES;
            nps_P.Drag.ballistic_number = orb_ballistic_number;
            nps_filter__run(&orb_gps_filter, &orb_gps.sv, &nps_P);
            break;

         case KEY_cntl_f2:
            orb_gps_filter.on = NO;
            break;

#        ifdef NEW_FILTER_SPAS
         case KEY_cntl_f3:
            if (spas_gps_filter.ring_full == NO)  break;
            spas_gps_filter.on = YES;
            nps_P.Drag.ballistic_number = spas_ballistic_number;
            nps_filter__run(&spas_gps_filter, &spas_gps.sv, &nps_P);
            break;

         case KEY_cntl_f4:
            spas_gps_filter.on = NO;
            break;
#        endif NEW_FILTER_SPAS
#        endif NPS_GPS

         case KEY_cntl_f6:
         case KEY_cntl_f7:
            if (nps_sub_plot == NPS_orb_tgt__orb_ins)       nps_dap_plot = &orb_tgt__orb_ins;
#           ifdef NPS_GPS
            else if (nps_sub_plot == NPS_spas_gps__orb_gps) nps_dap_plot = &spas_gps__orb_gps;
            else if (nps_sub_plot == NPS_spas_gps__orb_ins) nps_dap_plot = &spas_gps__orb_ins;
            else if (nps_sub_plot == NPS_orb_tgt__orb_gps)  nps_dap_plot = &orb_tgt__orb_gps;
#           endif NPS_GPS
```

```
            else  break;

            if (key == KEY_cntl_f7)
            {
                 nps_dap_on = YES;
                 nps_dap__display_init();
            }
            else nps_rndv__init();
            break;

         case KEY_cntl_f8:
            nps_predictor.thrust1.on = NO;
            break;

         case KEY_cntl_f9:
            nps_propagate_all(YES, 0.0);
            if (nps_newest_time > prev_save_time)
            {
                 prev_save_time = nps_newest_time;
                 nps_data_update_all();
            }
            break;

         case KEY_f10:
            if (nps_sub_plot == NPS_edit_menu)  break;
            nps_display_graphics(NO);
            nps_sub_plot = NPS_edit_menu;
            nps_edit_menu_display_init();
            break;

         case KEY_alt_f1:
            nps_menu_frozen = (nps_menu_frozen ? NO : YES);
            break;

#        ifdef NPS_SIM
         default:
            return NO;              // NOTE do not leave this in FLIGHT version
#        endif NPS_SIM
      }
      return YES;
}

/*=============================================================================
==========================
 *-----------------------------------------------------------------------*/
static void nps_menu__display_init(void)
{
   static char bar[] = "\xc4\xc4\xc4\xc4\xc4\xc4\xc4\xc4\xc4\xc4"
```

```
                    "\xc4\xc4\xc4\xc4\xc4\xc4\xc4\xc4\xc4\xc4\xc4";

static Edit_label title_labels[] =
{
        { 1, 1, "Naval Postgraduate School - Plotting Functions" },
        { 2,11, "--------Original-------- ------Propagated-----" },
        { 1,62, "Vectors in Kft" },

        { 3, 1, title_str_orb_ins },
        { 3,12, "X" }, { 4,12, "Y" }, { 5,12, "Z" }, { 6,10, "mag" },
        { 9, 1, title_str_orb_tgt },
        { 9,12, "X" }, {10,12, "Y" }, {11,12, "Z" }, {12,10, "mag" },

#       ifdef NPS_GPS
        {15, 1, title_str_orb_gps },
        {15,12, "X" }, {16,12, "Y" }, {17,12, "Z" }, {18,10, "mag" },
        {17, 1, "filter" },
        {21, 1, title_str_spas },
        {21,12, "X" }, {22,12, "Y" }, {23,12, "Z" }, {24,10, "mag" },
#       endif NPS_GPS

        {13,60, bar },
        {14,61, "ESC Return to prev" },
#       ifdef NPS_GPS
        {16,61, "F1  Spas/Orb GPS rv" },
        {17,61, "F2  Spas/Orb INS rv" },
        {18,61, "F3  Orb GPS/INS  rv" },
        {19,61, "F4  Spas/Orb TGT rv" },
        {20,61, "F5  Orb TGT/GPS  rv" },
#       endif NPS_GPS
        {21,61, "F6  Orb TGT/INS  rv" },
        {23,61, "F10 Change Settings" },
        {24,61, "Alt-F1 Freeze page" },
        {25,60, bar }
};
#define TITLE_LABEL_COUNT (sizeof(title_labels)/sizeof(Edit_label))

Edit_label *label;
int cnt;

/*-------------------------------------------------------------------
 */
clrscr();

for (label = title_labels, cnt = TITLE_LABEL_COUNT; cnt; cnt--, label++)
{
        gotoxy(label->col, label->row);
        cputs(label->string);
```

```
    }

    cnt = 13;
    gotoxy(59, cnt);  cputs("\xda");
    while (++cnt < 25) { gotoxy(59, cnt); cputs("\xb3"); }
    gotoxy(59, cnt);  cputs("\xc0");

    nps_menu__display_update();
}

/*=============================================================
=======================
          --------Original-------- ------Propagated------
Orb-INS  X -11111.111  -1111.111   -11111.111  -1111.111     Vectors in Kft
         Y -11111.111  -1111.111   -11111.111  -1111.111
         Z -11111.111  -1111.111   -11111.111  -1111.111
       mag -11111.111  -1111.111   -11111.111  -1111.111
             ddd/hh:mm:ss.sss        ddd/hh:mm:ss.sss
 *-------------------------------------------------------------------*/
static void nps_menu__display_vec(int row, int col, Vector *v, char *fmt)
{
    gotoxy(col, row);    cprintf(fmt, v->x * KILOM_TO_KILOFT);
    gotoxy(col, row+1);  cprintf(fmt, v->y * KILOM_TO_KILOFT);
    gotoxy(col, row+2);  cprintf(fmt, v->z * KILOM_TO_KILOFT);
    gotoxy(col, row+3);  cprintf(fmt, vector_magnitude(v) * KILOM_TO_KILOFT);
}

static void nps_menu__display_sv(int row, int col, State_vector *sv)
{
    static char fmt_pos[] = "%10.3f", fmt_vel[] = "%9.4f";

    double t1;
    char work[20];

    nps_menu__display_vec(row, col,    &sv->pos, fmt_pos);
    nps_menu__display_vec(row, col+12, &sv->vel, fmt_vel);

    t1 = sv->time + 86400;
    time_dbl_to_string(&t1, work);
    gotoxy(col+3, row+4);  cputs(work);
}

static void nps_menu__display_update()
{
    nps_menu__display_sv( 3,14, &orb_ins.input);
    nps_menu__display_sv( 3,36, &orb_ins.sv);

    nps_menu__display_sv( 9,14, &orb_tgt.input);
```

```
   nps_menu__display_sv( 9,36, &orb_tgt.sv);

#  ifdef NPS_GPS
   nps_menu__display_sv(15,14, &orb_gps.input);
   nps_menu__display_sv(15,36, &orb_gps.sv);

   gotoxy( 1,18);
   if (orb_gps_filter.ring_full)  cputs("RDY");
   else                           cprintf("%3d", orb_gps_filter.next_sample);
   gotoxy( 1,19);  cputs((orb_gps_filter.on ? " ON" : "OFF"));
   gotoxy( 5,19);  cputs((orb_gps_filter.auto_rerun ? "AUTO" : "    "));

   nps_menu__display_sv(21,14, &spas_gps.input);
   nps_menu__display_sv(21,36, &spas_gps.sv);
#  endif NPS_GPS
}

/*=============================================================================
============================
 *  These are helper routines for the edit screen below.
 *----------------------------------------------------------------------------*/
#ifdef NPS_GPS
static int filter_rerun_put(char *work, void *data)
{
   edit_field_put_dbl(work, data);
   if (*(double *)data < 60.0)  *(double *)data = 60.0;
   return YES;
}
#endif NPS_GPS

/*=============================================================================
============================
 *  This is the edit-screen for NPS rates & steps, these structures must reside
 * in this file because they reference static variables from this file.  These
 * structures should be static except for the screen structure, which is used in
 * the nps_edit_menu file.
 *
 *--123456789 123456789 123456789 123456789 123456789 123456789
 * 1                Sample and Step Rates
 * 2
 * 3  Sample rates (in secs, 0 = fast)
 * 4    plots  show: X
 * 5    buffer save: X
 * 6
 * 7  Orb-GPS  filter save: X
 * 8    auto-rerun: Y   rerun delay: X
 * 9
 *10  SPAS-GPS filter save: X
```

```
 *11    auto-rerun: Y   rerun delay: X
 *12
 *13   Step size for propagation (in secs): X
 *14   Maximum propagation time  (in secs): X
 *15   Test input vectors for max-prop time: Y
 *16
 *17   Position predictor settings
 *18     Points to display: X
 *19     Step size (secs):  X
 *20     Auto-mode delay:   X
 *-------------------------------------------------------------------------*/
#ifdef NPS_GPS
static char rate_lbl1[] = "auto-rerun: Y   rerun delay:";
#endif NPS_GPS

static Edit_label rate_labels[] =
{
   { 1,20, "Sample and Step Rates"},
   { 3, 4, "Sample rates (in secs, 0 = fast)"},
   { 4, 6, "plots  show:"},
   { 5, 6, "buffer save:"},
#  ifdef NPS_GPS
   { 7, 4, "Orb-GPS  filter save:"},
   { 8, 6, rate_lbl1},
#  ifdef NEW_FILTER_SPAS
   {10, 4, "SPAS-GPS filter save:"},
   {11, 6, rate_lbl1},
#  endif NEW_FILTER_SPAS
#  endif NPS_GPS
   {13, 4, "Step size for propagation (in secs): "},
   {14, 4, "Maximum propagation time  (in secs): "},
   {15, 4, "Test input vectors for max-prop time: "},
   {17, 4, "Position predictor settings"},
   {18, 6, "Points to display:"},
   {19, 6, "Step size (secs):"},
   {20, 6, "Auto-mode delay:"}
};
#define RATE_LABEL_COUNT (sizeof(rate_labels)/sizeof(Edit_label))

static Edit_field rate_fields[] =
{
   { 4,19, edit_field_get_dbl, edit_field_put_dbl, edit_field_key_dbl, &sample_rate_show },
   { 5,19, edit_field_get_dbl, edit_field_put_dbl, edit_field_key_dbl, &sample_rate_save },
#  ifdef NPS_GPS
   { 7,26, edit_field_get_dbl, edit_field_put_dbl, edit_field_key_dbl, &orb_gps_filter.save_step },
   { 8,18, edit_field_get_yesno,edit_field_put_yesno,edit_field_key_yesno,&orb_gps_filter.auto_rerun  },
   { 8,35, edit_field_get_dbl, filter_rerun_put,   edit_field_key_dbl, &orb_gps_filter.rerun_step },
#  ifdef NEW_FILTER_SPAS
```

```
   {10,26, edit_field_get_dbl, edit_field_put_dbl, edit_field_key_dbl, &spas_gps_filter.save_step },
   {11,18, edit_field_get_yesno,edit_field_put_yesno,edit_field_key_yesno,&spas_gps_filter.auto_rerun  },
   {11,35, edit_field_get_dbl, filter_rerun_put,   edit_field_key_dbl, &spas_gps_filter.rerun_step },
#  endif NEW_FILTER_SPAS
#  endif NPS_GPS
   {13,41, edit_field_get_dbl, edit_field_put_dbl, edit_field_key_dbl, &prop_step },
   {14,41, edit_field_get_dbl, edit_field_put_dbl, edit_field_key_dbl, &prop_max_time },
   {15,42, edit_field_get_yesno,edit_field_put_yesno,edit_field_key_yesno, &prop_chk_input_time },
   {18,25, edit_field_get_int, edit_field_put_int, edit_field_key_int, &nps_predictor.steps },
   {19,25, edit_field_get_dbl, edit_field_put_dbl, edit_field_key_dbl, &nps_predictor.step },
   {20,25, edit_field_get_dbl, edit_field_put_dbl, edit_field_key_dbl, &nps_predictor.delay }
};
#define RATE_FIELD_COUNT (sizeof(rate_fields)/sizeof(Edit_field))

Edit_screen nps_rate_screen =
{
   RATE_LABEL_COUNT, rate_labels,
   RATE_FIELD_COUNT, rate_fields,
   NULL, NO, NULL, NULL
};

/*=============================================================
============================
 *  This is the edit-screen for NPS Perturbations, same conditions as above.
 *
 *--123456789 123456789 123456789 123456789 123456789 123456789
 * 1                Propagation Perturbations
 * 2
 * 3  Drag Mode on: Y
 * 4
 * 5  Ballistic numbers
 * 6    Orbiter: X
 * 7    Payload: X
 * 8
 * 9  Drag Info
 *10    Solar Flux: X
 *11    Mean Solar Flux: X
 *12    Geomagnetic Activity Index: X
 *13
 *14  Propagator Info
 *15    Zonal Harmonics: X
 *16    Tesseral Harmonics: X
 *-------------------------------------------------------------*/
static Edit_label pert_labels[] =
{
   { 1,20, "Propagation Perturbations"},
   { 3, 4, "Drag Mode on:"},
   { 5, 4, "Ballistic numbers"},
```

```
    { 6, 6, "Orbiter:"},
    { 7, 6, "Payload:"},
    { 9, 4, "Drag Information"},
    { 10, 6, "Solar Flux:"},
    { 11, 6, "Mean Solar Flux:"},
    { 12, 6, "Geomagnetic Activity Index:"},
    { 14, 4, "Propagator Information"},
    { 15, 6, "Zonal Harmonics:"},
    { 16, 6, "Tesseral Harmonics:"}
};
#define PERT_LABEL_COUNT (sizeof(pert_labels)/sizeof(Edit_label))

static Edit_field pert_fields[] =
{
    { 3,18, edit_field_get_yesno, edit_field_put_yesno, edit_field_key_yesno, &nps_P.Drag.mode },
    { 6,15, edit_field_get_dbl,   edit_field_put_dbl,   edit_field_key_dbl,   &orb_ballistic_number },
    { 7,15, edit_field_get_dbl,   edit_field_put_dbl,   edit_field_key_dbl,   &spas_ballistic_number },
    { 10,18, edit_field_get_dbl,  edit_field_put_dbl,   edit_field_key_dbl,   &nps_P.Drag.solar_flux },
    { 11,23, edit_field_get_dbl,  edit_field_put_dbl,   edit_field_key_dbl,   &nps_P.Drag.mean_solar_flux },
    { 12,34, edit_field_get_dbl,  edit_field_put_dbl,   edit_field_key_dbl,   &nps_P.Drag.mean_solar_flux },
    { 15,23, edit_field_get_int,  edit_field_put_int,   edit_field_key_int,   &nps_P.Gravity.number_zonal },
    { 16,26, edit_field_get_int,  edit_field_put_int,   edit_field_key_int,   &nps_P.Gravity.number_tesseral }
};
#define PERT_FIELD_COUNT (sizeof(pert_fields)/sizeof(Edit_field))

Edit_screen nps_pert_screen =
{
    PERT_LABEL_COUNT, pert_labels,
    PERT_FIELD_COUNT, pert_fields,
    NULL, NO, NULL, NULL
};

/*==========================================================
 *  These are helper routines for the edit screen below.
 *----------------------------------------------------------*/
static Nps_state_vector_info sv_temp;

static void state_vector_scale(State_vector *v, double scale)
{
    v->pos.x *= scale;  v->pos.y *= scale;  v->pos.z *= scale;
    v->vel.x *= scale;  v->vel.y *= scale;  v->vel.z *= scale;
}

static void sv_pre_screen(Edit_screen *screen)
{
    Nps_state_vector_info *sv = (Nps_state_vector_info *)screen->data;
```

```
    if (sv == &orb_ins)
    {
        screen->labels[0].string = "Orbiter Inertial State Vector";
    }
    else if (sv == &orb_tgt)
    {
        screen->labels[0].string = "Orbiter Target State Vector";
    }
#   ifdef NPS_GPS
    else if (sv == &spas_gps)
    {
        screen->labels[0].string = "Spas GPS State Vector";
        screen->labels[10].string =
          "Source for SPAS vector (0 SPAS-GPS, 1 SPAS-SV):";
    }
    else  /* sv == &orb_gps */
    {
        screen->labels[0].string = "Orbiter GPS State Vector";
        screen->labels[10].string =
          "Source for GPS vector (0 GPS-M50, 1 GPS-WGS84):";
    }
#   endif NPS_GPS
    sv_temp = *sv;
    state_vector_scale(&sv_temp.sv, KILOM_TO_KILOFT);
}

static void sv_post_screen(Edit_screen *screen)
{
    Nps_state_vector_info *sv = (Nps_state_vector_info *)screen->data;

    if (screen->changed == NO) return;

    if ((sv->in_use = sv_temp.in_use) == NO)
    {
        state_vector_scale(&sv_temp.sv, KILOFT_TO_KILOM);
        sv->sv = sv_temp.sv;
        sv->got_one = YES;
    }

    if ((sv->fix_time = sv_temp.fix_time) == YES)
        sv->dtime = sv_temp.dtime;

#   ifdef NPS_GPS
    if (sv == &spas_gps)
    {
        if (sv_temp.source < SPAS_src_max) spas_gps.source = sv_temp.source;
        strcpy(title_str_spas, title_str_spas_src[spas_gps.source]);
        if (sv_temp.converter < NPS_M50_cnv_max) spas_gps.converter = sv_temp.converter;
```

```
    }
    else if (sv == &orb_gps)
    {
        if (sv_temp.source < ORB_GPS_src_max)  orb_gps.source = sv_temp.source;
        if (sv_temp.converter < NPS_M50_cnv_max)  orb_gps.converter = sv_temp.converter;
    }
#   endif NPS_GPS
}

/*=============================================================
============================
 *  This is the edit-screen for the State Vectors, same conditions as above.
 *
 *--123456789 123456789 123456789 123456789 123456789 123456789
 * 1                ORBITER STATE VECTOR
 * 2
 * 3   Position x: X
 * 4     Kft   y: X
 * 5           z: X
 * 6
 * 7   Velocity x: X
 * 8     Kft/s y: X
 * 9               z: X
 *10
 *11   Time (ddd/hh:mm:ss): DDD/HH:MM:SS.SSS
 *12
 *13   Use Realtime Vector: Y
 *14
 *15   Apply time correction: Y  (delta in secs): X
 *16
 *17   Source for SPAS vector (0 SPAS-GPS, 1 SPAS-SV): X
 *18
 *19   Converter for WGS84-M50 (0 iload file, 1 computed): X
 *-------------------------------------------------------------------------*/
static Edit_label sv_labels[] =
{
    { 1,20, NULL},
    { 3, 4, "Position x:"},
    { 4, 7, "Kft   y:"},
    { 5,13, "z:"},
    { 7, 4, "Velocity x:"},
    { 8, 7, "Kft/s y:"},
    { 9,13, "z:"},
    { 11, 4, "Time (ddd/hh:mm:ss):"},
    { 13, 4, "Use Realtime Vector:"},
    { 15, 4, "Apply time correction: Y  (delta in secs): X"},
    { 17, 4, NULL},
    { 19, 4, "Converter for WGS84-M50 (0 iload file, 1 computed):"},
```

```
};
#define SV_LABEL_COUNT (sizeof(sv_labels)/sizeof(Edit_label))

static Edit_field sv_fields[] =
{
   { 3,16, edit_field_get_dbl,  edit_field_put_dbl,  edit_field_key_dbl,  &sv_temp.sv.pos.x },
   { 4,16, edit_field_get_dbl,  edit_field_put_dbl,  edit_field_key_dbl,  &sv_temp.sv.pos.y },
   { 5,16, edit_field_get_dbl,  edit_field_put_dbl,  edit_field_key_dbl,  &sv_temp.sv.pos.z },
   { 7,16, edit_field_get_dbl,  edit_field_put_dbl,  edit_field_key_dbl,  &sv_temp.sv.vel.x },
   { 8,16, edit_field_get_dbl,  edit_field_put_dbl,  edit_field_key_dbl,  &sv_temp.sv.vel.y },
   { 9,16, edit_field_get_dbl,  edit_field_put_dbl,  edit_field_key_dbl,  &sv_temp.sv.vel.z },
   {11,25, edit_field_get_dhms1,edit_field_put_dhms1,edit_field_key_dhms, &sv_temp.sv.time },
   {13,25, edit_field_get_yesno,edit_field_put_yesno,edit_field_key_yesno,&sv_temp.in_use },
   {15,27, edit_field_get_yesno,edit_field_put_yesno,edit_field_key_yesno,&sv_temp.fix_time },
   {15,47, edit_field_get_dbl,  edit_field_put_dbl,  edit_field_key_dbl,  &sv_temp.dtime },
   {17,52, edit_field_get_int,  edit_field_put_int,  edit_field_key_int,  &sv_temp.source },
   {19,56, edit_field_get_int,  edit_field_put_int,  edit_field_key_int,  &sv_temp.converter }
};
#define SV_FIELD_COUNT (sizeof(sv_fields)/sizeof(Edit_field))

Edit_screen nps_orb_ins_sv_screen =
{
   SV_LABEL_COUNT - 2, sv_labels,
   SV_FIELD_COUNT - 2, sv_fields,
   (void *)&orb_ins, NO, sv_pre_screen, sv_post_screen
};

Edit_screen nps_orb_tgt_sv_screen =
{
   SV_LABEL_COUNT - 2, sv_labels,
   SV_FIELD_COUNT - 2, sv_fields,
   (void *)&orb_tgt, NO, sv_pre_screen, sv_post_screen
};

#ifdef NPS_GPS
Edit_screen nps_spas_gps_sv_screen =
{
   SV_LABEL_COUNT, sv_labels,
   SV_FIELD_COUNT, sv_fields,
   (void *)&spas_gps, NO, sv_pre_screen, sv_post_screen
};

Edit_screen nps_orb_gps_sv_screen =
{
   SV_LABEL_COUNT, sv_labels,
   SV_FIELD_COUNT, sv_fields,
   (void *)&orb_gps, NO, sv_pre_screen, sv_post_screen
};
```

#endif NPS_GPS

```
/*=============================================================
============ ============
 *-----------------------------------------------------------*/
#ifndef _NPS_
#define _NPS_

#include <stdlib.h>
#include <graphics.h>

#include <vector.h>
#include <keys.h>
#include <orbmech1.h>
#include <emm.h>

#include <nps_if.h>

/*=============================================================
=========================
 *-----------------------------------------------------------*/
extern int
   nps_menu__key_handler(int),
   nps_bgprop(State_vector *sv, double new_time, double input_time, int is_orbiter);

extern void
   nps_menu__display_init(void),
   nps_menu__display_update(void),
   nps_display_graphics(int graph_mode);

extern int
   nps_edit_menu_key_handler(int);
extern void
   nps_edit_menu_display_init(void);

extern int
   nps_dap__key_handler(int);
extern void
   nps_dap__display_init(void),
   nps_rndv__init(void);

/*=============================================================
=========================
 *  Only one of these structures will exist, it will be globally visible, and
 * it will have the same name as the type (but all lower case).
 *  The xmax/ymax members will be the number of pixels, not the largest valid
 * pixel.  For example, if the screen is 640 pixels wide, then xmax will be 640,
 * not 639 (which is the largest valid x-pixel).
 *-----------------------------------------------------------*/
typedef struct
{
   struct viewporttype main_view;
```

```
    int xmax, xmid, c_width;
    int ymax, ymid, c_height;
    int font_size, in_graphics_mode, page;
} Nps_graphics_info;

typedef struct { double origin, range; } Nps_plot_axis;

enum nps_axis { NPS_axis_X, NPS_axis_Y, NPS_axis_Z };

typedef struct
{
    char *title, *target_title, *chaser_title;
    Nps_plot_axis axis[3];
    int axis_vert, axis_horz;
} Nps_plot_info;

typedef struct { int id;  char *text; } Nps_label;

typedef struct { char pattern[24]; } Nps_log_file_header;

/*-------------------------------------------------------------------------*/
extern void
    nps_plot__vlabels(int x, int y, int ystep, Nps_label *label, int cnt),
    nps_plot__clear_view(struct viewporttype *view, int outline),
    nps_plot__set_view(struct viewporttype *view, int outline),
    nps_plot__main_view(void),
    nps_menuplot__init(char **strings);

extern int
    nps_plot__next_page(void);

/*=========================================================================
=========================
 *  These structures help organize all the information known about a given
 * state-vector or quaternion.
 *-------------------------------------------------------------------------*/
typedef struct
{
    int in_use, got_one, source, converter, fix_time;
    double dtime;
    State_vector input, sv;
} Nps_state_vector_info;

typedef struct
{
    int in_use, got_one;
    double input_time;
    Quaternion quat;
} Nps_quaternion_info;
```

```
/*=============================================================
============================
 *   All thrusts are in km/s, just like all the other vectors.
 *-------------------------------------------------------------*/
typedef struct
{
   int on, delayed, in_lvlh;
   double time;
   Vector dv_body;
} Nps_predictor_thrust;

enum nps_predictor_state { NPS_predict_off, NPS_predict_on, NPS_predict_now };

typedef struct
{
   int on, steps;
   double step, delay, next_time;
   Perturbations *perts;
   Nps_quaternion_info *q_btoi;
   Nps_predictor_thrust thrust1;
} Nps_predictor;

/*-------------------------------------------------------------*/
extern int
   nps_predictor__thrust(Nps_predictor *predictor, State_vector *chaser),
   nps_predictor__rndv(Nps_predictor *predictor,
                       State_vector *target, State_vector *chaser,
                       double ttr);

/*=============================================================
============================
 *-------------------------------------------------------------*/
enum nps_rv_type { NPS_rvplot_rv, NPS_rvplot_hv, NPS_rvplot_rh };

typedef struct
{
   Nps_plot_info display;
   RV_vector *ring_buffer;
   ushort ring_max, prev_sample, next_sample, ring_full;

   char ***menu_list;
   int menu_id, hbar_on, plot_type;

   char *log_file_name;
   Nps_log_file_header *log_file_hdr;

   Emm_handle emm;

   Nps_state_vector_info *target, *chaser;
   Nps_predictor *predictor;
```

```
    Vector last_rvdot;
    double crossing_time;
} Nps_rvplot_info;

/*-----------------------------------------------------------------------*/
extern int
    nps_rvplot__data_init(Nps_rvplot_info *info,
                          ushort max_mem, ushort config_mem, Emm_info *emm),
    nps_rvplot__key_handler(Nps_rvplot_info *info, int key);

extern void
    nps_rvplot__data_stop(Nps_rvplot_info *info),
    nps_rvplot__data_save(Nps_rvplot_info *info),
    nps_rvplot__data_flush(Nps_rvplot_info *info),
    nps_rvplot__display_init(Nps_rvplot_info *info),
    nps_rvplot__display_update(Nps_rvplot_info *info, double ttime, double ctime),
    nps_rvplot__data_update(Nps_rvplot_info *info),
    nps_rvplot__display_future(Nps_rvplot_info *info),
    nps_rvplot__menu_off(Nps_rvplot_info *info);

extern void
    nps_rvplot__init(Nps_plot_info *info, RV_vector *prev, Vector *prev_rvdot,
                     char **menu, int type, int hbar, ushort bmax, ushort bcnt),
    nps_rvplot__time(double t1, double t2),
    nps_rvplot__values(double x, double y, double z, double dist,
                     double closure, ushort bcnt),
    nps_rvplot__hbar(double y, double ydot, double crossing_time),
    nps_rvplot__point(Nps_plot_info *info, Vector *pos,
                     double t1, double t2, ushort bcnt,
                     Vector *rvdot, double crossing_time),
    nps_rvplot__point_fast(Nps_plot_info *info, Vector *pos),
    nps_rvplot__point_text(Nps_plot_info *info, Vector *pos, char *text, int bright),
    nps_rvplot__rndv(Nps_plot_info *info, Vector *rndv_dv, double rndv_dist);

/*=======================================================================
=========================
 *-----------------------------------------------------------------------*/
enum nps_diff_type { NPS_diff_pos, NPS_diff_vel };

typedef struct { double diff[2], time; } Diff_vector;

typedef struct
{
    Nps_plot_info displays[2];
    Diff_vector *ring_buffer;
    ushort ring_max, prev_sample, next_sample, ring_full;
    double begin_time;

    char ***menu_list;
    int menu_id, active_plot;
```

```
    char *log_file_name;
    Nps_log_file_header *log_file_hdr;

    Emm_handle emm;

    Nps_state_vector_info *target, *chaser;
} Nps_diffplot_info;

/*-----------------------------------------------------------------*/
extern int
    nps_diffplot__data_init(Nps_diffplot_info *info,
                            ushort max_mem, ushort config_mem, Emm_info *emm),
    nps_diffplot__key_handler(Nps_diffplot_info *info, int key);

extern void
    nps_diffplot__data_stop(Nps_diffplot_info *info),
    nps_diffplot__data_save(Nps_diffplot_info *info),
    nps_diffplot__data_flush(Nps_diffplot_info *info),
    nps_diffplot__display_init(Nps_diffplot_info *info),
    nps_diffplot__display_update(Nps_diffplot_info *info, double ttime, double ctime),
    nps_diffplot__data_update(Nps_diffplot_info *info);

extern void
    nps_diffplot__init(Nps_plot_info *info, int diff_type, char **menu),
    nps_diffplot__time(double t1, double t2),
    nps_diffplot__values(double diff, double dtime, int diff_type),
    nps_diffplot__point(Nps_plot_info *info, double diff, double dtime,
                        int diff_type, double t1, double t2),
    nps_diffplot__point_fast(Nps_plot_info *info, double diff, double dtime);

/*================================================================
==========================
 *-----------------------------------------------------------------*/
typedef struct
{
    char *title, *target_title, *chaser_title, **menu;
} Nps_pyplot_info;

/*-----------------------------------------------------------------*/
extern int
    nps_pyplot__key_handler(Nps_pyplot_info *info, int key);

extern void
    nps_pyplot__display_init(Nps_pyplot_info *info),
    nps_pyplot__display_update(Nps_pyplot_info *info,
                               State_vector *target, State_vector *orbiter,
                               Quaternion *quat, double ttime, double otime),
    nps_pyplot__display_stats(Nps_pyplot_info *info, State_vector *target,
                              State_vector *orbiter, Quaternion *quat);
```

```
extern void
   nps_pyplot__init(Nps_pyplot_info *info),
   nps_pyplot__time(double t1, double t2),
   nps_pyplot__update(Nps_pyplot_info *info, Vector *pyd, double t1, double t2),
   nps_pyplot__stats(Nps_pyplot_info *info, Vector *pyd, double time),
   nps_pitch_yaw(Quaternion *q, State_vector *target, State_vector *orb, Vector *pyd);

/*==========================================================
==========================
 *-----------------------------------------------------------*/
#define NPS_FILTER_SAMPLES 60

typedef struct
{
   int on, auto_rerun, input_saved;
   double save_step, rerun_step, next_rerun;
   State_vector input_vector;
   ushort prev_sample, next_sample, ring_full;
   double ring_buffer[4][NPS_FILTER_SAMPLES];
} Nps_filter_info;

/*-----------------------------------------------------------*/
extern void
   nps_filter__data_flush(Nps_filter_info *info),
   nps_filter__data_save(Nps_filter_info *info),
   nps_filter__data_update(Nps_filter_info *info, State_vector *new_vector),
   nps_filter__run(Nps_filter_info *info, State_vector *out_vector, Perturbations *P);

/*---OLD FILTER
   buffer_raw_orb_gps(State_vector *nps_tans, double tans_buf[4][60], int tans_mat_dat[2]);
---*/

extern int
   ls_filter(State_vector *tans_in, State_vector *tans_out,
             double tans_buf[4][60], int fbi, int point_num, Perturbations *P);

/*==========================================================
==========================
 *-----------------------------------------------------------*/
typedef struct
{
   double a, e, Eo, Mo, Om, i, om, n;
} Orbmech_classicals;

/*-----------------------------------------------------------*/
extern void
   getclas(Vector *pos, Vector *vel, Orbmech_classicals *class, Perturbations *P),
   f_and_g(Vector *pos, Vector *vel, Vector *pos_new, Vector *vel_new,
           Orbmech_classicals *class, double t_dif);
```

```
extern double
   plane_crossing_time(State_vector *chaser, State_vector *target, Perturbations *P);

#endif _NPS_
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#include <stdio.h>

#include <keys.h>
#include <edit_rt.h>

#include "nps.h"

/*=============================================================
============================
 *-------------------------------------------------------------*/
extern Nps_graphics_info nps_graphics_info;

extern double nps_newest_time;
extern Nps_predictor nps_predictor;
extern Nps_rvplot_info *nps_dap_plot;

/*=============================================================
============================
 *-------------------------------------------------------------*/
typedef struct { Vector plus, minus; } Nps_dap_pair;

typedef struct { Nps_dap_pair x, y, z, z_low; } Nps_dap_info;

enum dap_types { DAP_a, DAP_b };
enum dap_axis { DAP_x_plus, DAP_x_minus, DAP_z_plus, DAP_z_minus, DAP_y_plus, DAP_y_minus };
enum dap_keys { DAP_left, DAP_up, DAP_down, DAP_right, DAP_plus, DAP_minus };

/*=============================================================
============================
 *  All DAP thrusts are in ft/sec.  X thrusts have a Z component, forward jets
 * are 9.6 degrees, rear jets are 10 degrees
 *  NOTE: the predictor thrust vector must be in km/sec, not ft/sec.
 *-------------------------------------------------------------*/
static Nps_dap_info dap_a =
{
   {                                    /* X          */
        { 0.056,  0.0,  0.00987 },      /*  plus      */
        { -0.055,  0.0,  0.00947 }      /*  minus     */
   },
   {                                    /* Y          */
        { 0.0,  0.056,  0.0     },      /*  plus      */
        { 0.0, -0.056,  0.0     } /*  minus      */
   },
   {                                    /* Z          */
        { 0.0,    0.0,  0.055   },      /*  plus      */
        { 0.0,    0.0, -0.076   } /*  minus      */
   },
```

```
    {                                   /* Z low        */
        { 0.0,   0.0,  0.052  },        /*  plus        */
        { 0.0,   0.0, -0.076  } /*  minus       */
    }
};

static Nps_dap_info dap_b =
{
    {                                   /* X            */
        { 0.016,  0.0,  0.00282 },      /*  plus        */
        { -0.016,  0.0,  0.00270 }      /*  minus       */
    },
    {                                   /* Y            */
        { 0.0,  0.016,  0.0    },       /*  plus        */
        { 0.0, -0.016,  0.0    } /*  minus       */
    },
    {                                   /* Z            */
        { 0.0,   0.0,  0.024  },        /*  plus        */
        { 0.0,   0.0, -0.034  } /*  minus       */
    },
    {                                   /* Z low        */
        { 0.0,   0.0,  0.013  },        /*  plus        */
        { 0.0,   0.0, -0.034  } /*  minus       */
    },
};

static double launch_time, thrust_time_met;
static int thrust_use_gmt = NO;
static int low_z_mode, dap_type;

static int
    axis_counts[6],
    lvlh_key_to_axis[6] = { DAP_x_plus, DAP_z_minus, DAP_z_plus, DAP_x_minus, DAP_y_plus,
DAP_y_minus },
    body_key_to_axis[6] = { DAP_z_plus, DAP_x_plus, DAP_x_minus, DAP_z_minus, DAP_y_plus,
DAP_y_minus },
    opposite_axis[6]    = { DAP_x_minus, DAP_x_plus, DAP_z_minus, DAP_z_plus, DAP_y_minus,
DAP_y_plus };

static double time_to_rndv = 60.0 * 60.0; // 60 min

/*=============================================================
=========================
 *-------------------------------------------------------------*/
static struct
{
    struct viewporttype view;
    int height, line1_y, line2_y;
    int dap_x, mode_x1, mode_x2, mode_x3;
    int bar_x1, keys1_x1, keys1_x2;
```

```
    int bar_x2, data_x1, data_x2;
    int bar_x3, keys2_x1, keys2_x2, keys2_x3, keys2_x4;
    int a_length, a_height, p_length, p_height;
} menu;

/*==========================================================================
==========================
 *-------------------------------------------------------------------------*/
static void points_init(void)
{
    int xmax = nps_graphics_info.xmax;
    int ymax = nps_graphics_info.ymax;
    int c_width = nps_graphics_info.c_width;
    int c_height = nps_graphics_info.c_height;

    /*-------------------------------------------------------------------
     */
    menu.view.left   = 0;
    menu.view.right  = xmax - 1;
#ifdef DEBUG
    menu.view.top    = ymax - 1 - (menu.height = c_height * 4);
#else
    menu.view.top    = ymax - 1 - (menu.height = c_height * 3 - c_height/4);
#endif DEBUG
    menu.view.bottom = ymax - 1;
    menu.view.clip   = YES;

    menu.line1_y = c_height - c_height/4;
    menu.line2_y = menu.line1_y + c_height + c_height/4;

    /*-------------------------------------------------------------------
     */
    menu.dap_x = c_width;
    menu.mode_x1 = c_width * 2;
    menu.mode_x2 = menu.mode_x1 + c_width * 7;
    menu.mode_x3 = menu.mode_x2 + c_width * 8;

    /*-------------------------------------------------------------------
     */
    menu.bar_x1 = xmax * 0.21;
    menu.keys1_x1 = menu.bar_x1 + c_width;
    menu.keys1_x2 = menu.keys1_x1 + c_width*5;

    /*-------------------------------------------------------------------
     */
    menu.bar_x2 = xmax * 0.42;
    menu.data_x1 = menu.bar_x2 + c_width;
    menu.data_x2 = menu.data_x1 + c_width * 2;

    menu.a_length = c_height + c_height/2;
```

```
    menu.p_length = 3;
    menu.p_height = menu.p_length;
    menu.a_height = menu.p_length * 2;

    /*-----------------------------------------------------------------
     */
    menu.bar_x3 = xmax * 0.56;

    menu.keys2_x1 = menu.bar_x3 + c_width;
    menu.keys2_x2 = menu.keys2_x1 + c_width * 12;
    menu.keys2_x3 = menu.keys2_x2 + c_width * 10;
    menu.keys2_x4 = menu.keys2_x3 + c_width * 12;
}

static void draw_arrow(int x, int y, int a_xdt, int a_ydt)
{
    moveto(x, y);
    linerel( a_xdt * menu.a_length,  a_ydt * menu.a_length);
    linerel(-(a_xdt * menu.p_length + a_ydt * menu.p_height),
            -(a_ydt * menu.p_length + a_xdt * menu.p_height));
    linerel(a_ydt * menu.a_height, a_xdt * menu.a_height);
    linerel(a_xdt * menu.p_length - a_ydt * menu.p_height,
            a_ydt * menu.p_length - a_xdt * menu.p_height);
}

/*==========================================================================
==========================
 *-------------------------------------------------------------------------*/
void nps_dap__menu(void)
{
    static char
        axis_label[4][4] = { "+X", "-X", "+Z", "-Z" },
        fmt1[] = "%s%2d %s";

    Vector dv_ft;
    char work[24];
    int x, y, *key_map, axis;

    /*-----------------------------------------------------------------
     */
    settextstyle(SMALL_FONT, HORIZ_DIR, nps_graphics_info.font_size);
    settextjustify(LEFT_TEXT,CENTER_TEXT);
    nps_plot__clear_view(&menu.view, YES);

    /*-----------------------------------------------------------------
     */
    outtextxy(menu.dap_x, menu.line1_y, (dap_type ? "DAP-B" : "DAP-A"));

    outtextxy(menu.mode_x1, menu.line2_y, "Modes:");
```

```
if (nps_predictor.thrust1.delayed)
      outtextxy(menu.mode_x2, menu.line2_y, "Delayed");
if (nps_predictor.thrust1.in_lvlh)
      outtextxy(menu.mode_x2, menu.line1_y, "in-LVLH");
if (low_z_mode)
      outtextxy(menu.mode_x3, menu.line2_y, "low-Z");

/*-----------------------------------------------------------------
 */
line(menu.bar_x1, 0, menu.bar_x1, menu.height);

outtextxy(menu.keys1_x2, menu.line1_y, "ESC, SPACE, Z,L");
outtextxy(menu.keys1_x2, menu.line2_y, "A,B, ENTER  + -");

x  = menu.keys1_x1;   y  = menu.line2_y;     draw_arrow(x,y,  0,-1);
x += menu.a_height;   y -= menu.a_length;    draw_arrow(x,y,  0, 1);
y += menu.a_length;
x += menu.p_height+2; y -= menu.p_height; draw_arrow(x,y,  1, 0);
x += menu.a_length;   y -= menu.a_height;    draw_arrow(x,y, -1, 0);

/*-----------------------------------------------------------------
 */
line(menu.bar_x2, 0, menu.bar_x2, menu.height);

dv_ft.x = nps_predictor.thrust1.dv_body.x * KILOM_TO_FT;
dv_ft.z = nps_predictor.thrust1.dv_body.z * KILOM_TO_FT;

sprintf(work, "%9.6lf", dv_ft.x);
outtextxy(menu.data_x1, menu.line1_y, "X:");
outtextxy(menu.data_x2, menu.line1_y, work);

sprintf(work, "%9.6lf", dv_ft.z);
outtextxy(menu.data_x1, menu.line2_y,  "Z:");
outtextxy(menu.data_x2, menu.line2_y,  work);

/*-----------------------------------------------------------------
 */
line(menu.bar_x3, 0, menu.bar_x3, menu.height);

outtextxy(menu.keys2_x1, menu.line2_y,
  (nps_predictor.thrust1.in_lvlh ? "(LVLH)" : "(Body)"));

key_map = (nps_predictor.thrust1.in_lvlh ? lvlh_key_to_axis : body_key_to_axis);

x = menu.keys2_x1 + menu.a_length;  y = menu.line1_y;
draw_arrow(x,y, -1, 0);
axis = key_map[DAP_left];
sprintf(work, fmt1, axis_label[axis], axis_counts[axis], "out");
outtextxy(x+4, menu.line1_y, work);
```

```
    x = menu.keys2_x2;  y = menu.line1_y + menu.a_length/2 + 2;
    draw_arrow(x,y,  0,-1);
    axis = key_map[DAP_up];
    sprintf(work, fmt1, axis_label[axis], axis_counts[axis], "up");
    outtextxy(x+5, menu.line1_y, work);

    x = menu.keys2_x2;  y = menu.line2_y - menu.a_length/2 - 2;
    draw_arrow(x,y,  0, 1);
    axis = key_map[DAP_down];
    sprintf(work, fmt1, axis_label[axis], axis_counts[axis], "down");
    outtextxy(x+5, menu.line2_y, work);

    x = menu.keys2_x3;  y = menu.line1_y;
    draw_arrow(x,y,  1, 0);
    axis = key_map[DAP_right];
    sprintf(work, fmt1, axis_label[axis], axis_counts[axis], "in");
    outtextxy(x+4+menu.a_length, menu.line1_y, work);

    if (nps_predictor.thrust1.in_lvlh == NO)
    {
        x = menu.keys2_x4;
        outtextxy(x, menu.line1_y, "Labels valid");
        outtextxy(x, menu.line2_y, "for pitch 90");
    }

    nps_plot__main_view();
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
void nps_dap__display_init()
{
    if (menu.height == 0)  points_init();

    nps_rvplot__menu_off(nps_dap_plot);
    nps_dap__menu();
}

void nps_dap__clear(int new_type)
{
    nps_predictor.thrust1.on = NO;
    nps_predictor.thrust1.dv_body.x =
    nps_predictor.thrust1.dv_body.y =
    nps_predictor.thrust1.dv_body.z = 0.0;

    axis_counts[DAP_x_plus]  =
    axis_counts[DAP_x_minus] =
    axis_counts[DAP_y_plus]  =
    axis_counts[DAP_y_minus] =
```

```
    axis_counts[DAP_z_plus]  =
    axis_counts[DAP_z_minus] = 0;

    dap_type = new_type;
}

/*=============================================================================
============================
 * Returns:
 *  0    Didn't use the key
 *  1    Used the key
 *  2    Used the key, My subsystem is done
 *-----------------------------------------------------------------------*/
int nps_dap__key_handler(int key)
{
    int axis, opp_axis, sign;
    Nps_dap_info *dap;
    Vector *dv;

    /*-------------------------------------------------------------------
     */
    switch (key)
    {
        case KEY_esc:
            nps_rvplot__display_init(nps_dap_plot);
            return 2;

        case KEY_space:
            nps_dap__clear(dap_type);
            nps_rvplot__display_init(nps_dap_plot);
            nps_dap__menu();
            return YES;

        case KEY_cr:
            nps_rvplot__display_future(nps_dap_plot);
            nps_dap__menu();
            return YES;

        case 'Z':
        case 'z':
            low_z_mode = (low_z_mode ? NO : YES);
            nps_dap__menu();
            return YES;

        case 'L':
        case 'l':
            nps_predictor.thrust1.in_lvlh = (nps_predictor.thrust1.in_lvlh ? NO : YES);
            nps_dap__menu();
            return YES;
```

```
    case 'A':
    case 'a':
       if (dap_type != DAP_a) { nps_dap__clear(DAP_a);  nps_dap__menu(); }
       return YES;

    case 'B':
    case 'b':
       if (dap_type != DAP_b) { nps_dap__clear(DAP_b);  nps_dap__menu(); }
       return YES;

    case KEY_up:    key = DAP_up;    break;
    case KEY_down:  key = DAP_down;  break;
    case KEY_left:  key = DAP_left;  break;
    case KEY_right: key = DAP_right; break;
    case '-':       key = DAP_minus; break;
    case '+':       key = DAP_plus;  break;

    default: return YES;
}

/*-------------------------------------------------------------------
 *  Check if there have been pulses in the opposite direction.  If so then
 * we'll use the opposite key and sutract delta-v, otherwise we want to use
 * this key and add delta-v.
 */
axis = (nps_predictor.thrust1.in_lvlh ? lvlh_key_to_axis : body_key_to_axis)[key];

opp_axis = opposite_axis[axis];

if (axis_counts[opp_axis]) { sign = -1; axis = opp_axis; }
else                       sign = 1;

/*-------------------------------------------------------------------
 */
axis_counts[axis] += sign;

dap = (dap_type == DAP_a ? &dap_a : &dap_b);

switch (axis)
{
    case DAP_x_plus:  dv = &dap->x.plus;  break;
    case DAP_x_minus: dv = &dap->x.minus; break;
    case DAP_z_plus:  dv = (low_z_mode ? &dap->z_low.plus : &dap->z.plus);   break;
    case DAP_z_minus: dv = (low_z_mode ? &dap->z_low.minus : &dap->z.minus); break;
    case DAP_y_plus:  dv = &dap->y.plus;  break;
    case DAP_y_minus: dv = &dap->y.minus; break;
}

/*-------------------------------------------------------------------
 */
```

```
    nps_predictor.thrust1.dv_body.x += dv->x * FT_TO_KILOM * sign;
    nps_predictor.thrust1.dv_body.y += dv->y * FT_TO_KILOM * sign;
    nps_predictor.thrust1.dv_body.z += dv->z * FT_TO_KILOM * sign;
    nps_predictor.thrust1.on = YES;

    nps_rvplot__display_future(nps_dap_plot);
    nps_dap__menu();

    return YES;
}

/*==========================================================
==========================
 *-------------------------------------------------------------------------*/
void nps_rndv__init()
{
#ifdef NPS_RNDV
    nps_predictor__rndv(&nps_predictor, &nps_dap_plot->target->sv,
      &nps_dap_plot->chaser->sv, time_to_rndv);
    nps_rvplot__display_future(nps_dap_plot);
#endif NPS_RNDV
}

/*==========================================================
==========================
 *-------------------------------------------------------------------------*/
static int edit_field_get_dbl9(char *work, void *data)
{
    sprintf(work, "%9.6lf", *(double *)data);
    return -1;
}

static int edit_field_get_dbl5(char *work, void *data)
{
    sprintf(work, "%5.2lf", *(double *)data);
    return -1;
}

/*==========================================================
==========================
 *--0123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
 * 1                Thrust information (ft/s)
 * 2
 * 3                  X-comp      Z-comp
 * 4  DAP-A   +x up    -0.000000   -0.000000   10  deg pitch
 * 5          -x down  -0.000000   -0.000000   9.6 deg pitch
 * 6          +z out               -0.000000          +y -0.000000
 * 7          -z in                -0.000000          -y -0.000000
 * 8  (low-z) +z out               -0.000000
 * 9          -z in                -0.000000
```

```
*10
*11 DAP-B   +x up    -0.000000   -0.000000
*12         -x down  -0.000000   -0.000000
*13         +z out               -0.000000          +y -0.000000
*14         -z in                -0.000000          -y -0.000000
*15 (low-z) +z out               -0.000000
*16         -z in                -0.000000
*17
*18 Future thrust: Y              Note: future thrusts use the current attitude,
*19  GMT-L  DDD/HH:MM:SS.SSS      so answers are valid only during inertial hold,
*20  + MET  DDD/HH:MM:SS.SSS      the alternative is to apply thrust in LVLH
*21  or GMT DDD/HH:MM:SS.SSS Y
*22
*23 Thrust: enabled: Y                      TTR (NO whole orbit intervals!)
*24         in LVLH: Y                        DDD/HH:MM:SS.SSS
*25 (XYZ):  -0.00  -0.00  -0.00
*-------------------------------------------------------------------------*/
static char
   lbl_xp[] = "+x up",
   lbl_xm[] = "-x down",
   lbl_yp[] = "+y",
   lbl_ym[] = "-y",
   lbl_zp[] = "+z out",
   lbl_zm[] = "-z in",
   lbl_zl[] = "(low-z)";

static Edit_label thrust_labels[] =
{
   { 1,18, "Thrust information (ft/s)"},
   { 3,20, "X-comp      Z-comp"},

   { 4, 1, "DAP-A"},
   { 4, 9, lbl_xp}, { 5, 9, lbl_xm},
   { 6, 9, lbl_zp}, { 7, 9, lbl_zm},
   { 6,49, lbl_yp}, { 7,49, lbl_ym},
   { 8, 9, lbl_zp}, { 9, 9, lbl_zm}, { 8, 1, lbl_zl},
   { 4,42, "10  deg pitch"}, { 5,42, "9.6 deg pitch"},

   {11, 1, "DAP-B"},
   {11, 9, lbl_xp}, {12, 9, lbl_xm},
   {13, 9, lbl_zp}, {14, 9, lbl_zm},
   {13,49, lbl_yp}, {14,49, lbl_ym},
   {15, 9, lbl_zp}, {16, 9, lbl_zm}, {15, 1, lbl_zl},

   {18, 1, "Future thrust:"},
   {19, 2, "GMT-L"},
   {20, 2, "+ MET"},
   {21, 2, "or GMT"},

   {18,27, "Note: future thrusts use the current attitude,"},
```

```
   {19,29, "so answers are valid only during inertial hold,"},
   {20,29, "the alternative is to apply thrust in LVLH"},

   {23, 1, "Thrust: enabled:"},
   {24, 9, "in LVLH:"},
   {25, 1, "(XYZ):"},
#ifdef NPS_RNDV
   {23,44, "TTR (NO whole orbit intervals!)"}
#endif NPS_RNDV
};
#define THRUST_LABEL_COUNT (sizeof(thrust_labels)/sizeof(Edit_label))

static Edit_field thrust_fields[] =
{
   { 4,18, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_a.x.plus.x },
   { 4,30, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_a.x.plus.z },
   { 5,18, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_a.x.minus.x },
   { 5,30, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_a.x.minus.z },
   { 6,30, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_a.z.plus.z },
   { 7,30, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_a.z.minus.z },
   { 8,30, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_a.z_low.plus.z },
   { 9,30, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_a.z_low.minus.z },
   { 6,52, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_a.y.plus.y },
   { 7,52, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_a.y.minus.y },

   {11,18, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_b.x.plus.x },
   {11,30, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_b.x.plus.z },
   {12,18, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_b.x.minus.x },
   {12,30, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_b.x.minus.z },
   {13,30, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_b.z.plus.z },
   {14,30, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_b.z.minus.z },
   {15,30, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_b.z_low.plus.z },
   {16,30, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_b.z_low.minus.z },
   {13,52, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_b.y.plus.y },
   {14,52, edit_field_get_dbl9, edit_field_put_dbl,  edit_field_key_dbl,  &dap_b.y.minus.y },

   {18,16, edit_field_get_yesno,edit_field_put_yesno,edit_field_key_yesno,&nps_predictor.thrust1.delayed },
   {19, 9, edit_field_get_dhms0,edit_field_put_dhms0,edit_field_key_dhms,  &launch_time },
   {20, 9, edit_field_get_dhms0,edit_field_put_dhms0,edit_field_key_dhms,  &thrust_time_met },
   {21, 9, edit_field_get_dhms1,edit_field_put_dhms1,edit_field_key_dhms,  &nps_predictor.thrust1.time },
   {21,26, edit_field_get_yesno,edit_field_put_yesno,edit_field_key_yesno,&thrust_use_gmt },

   {23,18, edit_field_get_yesno,edit_field_put_yesno,edit_field_key_yesno,&nps_predictor.thrust1.on },
   {24,18, edit_field_get_yesno,edit_field_put_yesno,edit_field_key_yesno,&nps_predictor.thrust1.in_lvlh },
   {25, 9, edit_field_get_dbl5, edit_field_put_dbl,  edit_field_key_dbl,  &nps_predictor.thrust1.dv_body.x },
   {25,16, edit_field_get_dbl5, edit_field_put_dbl,  edit_field_key_dbl,  &nps_predictor.thrust1.dv_body.y },
   {25,23, edit_field_get_dbl5, edit_field_put_dbl,  edit_field_key_dbl,  &nps_predictor.thrust1.dv_body.z },

#ifdef NPS_RNDV
   {24,48, edit_field_get_dhms0,edit_field_put_dhms0,edit_field_key_dhms,  &time_to_rndv }
```

```
#endif NPS_RNDV
};
#define THRUST_FIELD_COUNT (sizeof(thrust_fields)/sizeof(Edit_field))

/*=============================================================
=======================
 *-------------------------------------------------------------*/
static void vector_scale(Vector *v, double scale)
{
   v->x *= scale;  v->y *= scale;  v->z *= scale;
}

static void thrust_pre_screen(Edit_screen *screen)
{
   if (nps_predictor.thrust1.time < nps_newest_time)
        nps_predictor.thrust1.time = nps_newest_time;

   vector_scale(&nps_predictor.thrust1.dv_body, KILOM_TO_FT);
}

static void thrust_post_screen(Edit_screen *screen)
{
   if (thrust_use_gmt)
        thrust_time_met = nps_predictor.thrust1.time - launch_time;
   else
        nps_predictor.thrust1.time = launch_time + thrust_time_met;

   vector_scale(&nps_predictor.thrust1.dv_body, FT_TO_KILOM);
   if (time_to_rndv <= 60.0)  time_to_rndv = 60.0;
}

Edit_screen nps_thrust_screen =
{
   THRUST_LABEL_COUNT, thrust_labels,
   THRUST_FIELD_COUNT, thrust_fields,
   NULL, NO, thrust_pre_screen, thrust_post_screen
};
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>

#include <keys.h>
#include <edit_rt.h>

/*=============================================================
============================
 *-------------------------------------------------------------*/
extern Edit_screen
   nps_rate_screen,
   nps_pert_screen,
   nps_orb_ins_sv_screen,
   nps_orb_tgt_sv_screen,
#  ifdef NPS_GPS
   nps_orb_gps_sv_screen,
   nps_spas_gps_sv_screen,
#  endif NPS_GPS
   nps_thrust_screen;

/*=============================================================
============================
 *-------------------------------------------------------------*/
static Edit_screen *screens[] =
{
   &nps_rate_screen, &nps_pert_screen,
   &nps_orb_ins_sv_screen, &nps_orb_tgt_sv_screen,
#  ifdef NPS_GPS
   &nps_orb_gps_sv_screen, &nps_spas_gps_sv_screen,
#  endif NPS_GPS
   &nps_thrust_screen
};
#define MAX_SCREENS (sizeof(screens)/sizeof(Edit_screen *))

static int current_screen;

/*=============================================================
============================
 * Returns:
 *  0    Didn't use the key
 *  1    Used the key
 *  2    Used the key, My subsystem is done
 *-------------------------------------------------------------*/
int nps_edit_menu_key_handler(int key)
{
   int rc = edit_screen_key_handler(screens[current_screen], key);
```

```
    if (rc > 0)  return rc;

    /*-----------------------------------------------------------------
     */
    switch (key)
    {
         case KEY_pgup:
            if (--current_screen < 0)  current_screen = MAX_SCREENS - 1;
            break;

         case KEY_pgdn:
            if (++current_screen >= MAX_SCREENS)  current_screen = 0;
            break;

         default:  return NO;
    }

    edit_screen_init(screens[current_screen]);
    return YES;
}

/*=================================================================
==========================
 *-----------------------------------------------------------------*/
void nps_edit_menu_display_init()
{
    edit_screen_init(screens[current_screen]);
}
```

```
/*=============================================================
==========================
 *-------------------------------------------------------------*/
#include <stddef.h>

#include "nps.h"

/*=============================================================
==========================
 *-------------------------------------------------------------*/
void nps_filter__data_flush(Nps_filter_info *this)
{
   this->ring_full = NO;
   this->prev_sample = this->next_sample = 0;
   this->ring_buffer[3][0] = 0.0;
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
void nps_filter__data_save(Nps_filter_info *this)
{
   /*-----------------------------------------------------------
    */
   if (this->input_saved) return;

   this->ring_buffer[0][this->next_sample] = this->input_vector.pos.x;
   this->ring_buffer[1][this->next_sample] = this->input_vector.pos.y;
   this->ring_buffer[2][this->next_sample] = this->input_vector.pos.z;
   this->ring_buffer[3][this->next_sample] = this->input_vector.time;

   this->prev_sample = this->next_sample;
   this->next_sample++;
   if (this->next_sample >= NPS_FILTER_SAMPLES)
   {
        this->next_sample = 0;
        this->ring_full = YES;
   }
   this->input_saved = YES;
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
void nps_filter__data_update(Nps_filter_info *this, State_vector *new_vector)
{
   /*-----------------------------------------------------------
    */
   this->input_saved = NO;
   this->input_vector = *new_vector;
```

```
    if (this->input_vector.time - this->ring_buffer[3][this->prev_sample] >= this->save_step)
        nps_filter__data_save(this);
}

/*==========================================================
==========================
 *-----------------------------------------------------------*/
void nps_filter__run(Nps_filter_info *this, State_vector *out_vector, Perturbations *P)
{
    nps_filter__data_save(this);

    ls_filter(&this->input_vector, out_vector,
      this->ring_buffer, this->ring_full, this->prev_sample, P);

    if (this->auto_rerun)
        this->next_rerun = this->input_vector.time + this->rerun_step;
}
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#include <stdio.h>

#include "nps.h"
#include "nps_m50.h"

/*=============================================================
============================
 *-------------------------------------------------------------*/
static State_vector_transform m50_ecef;

#define M50_TO_WGS84 0
#define WGS84_TO_M50 1

/*=============================================================
============================
 *-------------------------------------------------------------*/
int nps_wgs84__read_iload(char *filename)
{
   FILE *iload;

   /*---------------------------------------------------------
    */
   if ((iload = fopen(filename, "r")) == NULL)  return NO;

   fscanf(iload, "%le", &m50_ecef.time);
   m50_ecef.time -= 86400.0;              // Make time rel-secs, like bgprop()

   fscanf(iload, "%le %le %le %le %le %le %le %le %le",
     &m50_ecef.matrix[0][0], &m50_ecef.matrix[0][1], &m50_ecef.matrix[0][2],
     &m50_ecef.matrix[1][0], &m50_ecef.matrix[1][1], &m50_ecef.matrix[1][2],
     &m50_ecef.matrix[2][0], &m50_ecef.matrix[2][1], &m50_ecef.matrix[2][2]);

   fclose(iload);

   return YES;
}

/*=============================================================
============================
 *   Note: Velocity may be an inertial velocity expressed in ECEF coordinates.
 * Otherwise it is an ECEF velocity, expressed in ECEF coordinates.
 *-------------------------------------------------------------*/
void nps_wgs84_to_m50(State_vector *vector, int converter, int vel_is_ecef)
{
   State_vector m50;

   /*---------------------------------------------------------
```

```
    */
   if (converter == NPS_M50_from_iload)
        state_vector__ecef_to_m50(vector, &m50, &m50_ecef, vel_is_ecef);
   else if (converter == NPS_M50_computed)
        m50_wgs84_m50(vector, &m50, WGS84_TO_M50, NULL, vel_is_ecef);
   else return;

   m50.time = vector->time;
   *vector = m50;
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
void nps_m50_to_wgs84(State_vector *vector, int converter, int vel_is_ecef)
{
   State_vector ecef;

   /*-------------------------------------------------------------
    */
   if (converter == NPS_M50_from_iload)
        state_vector__m50_to_ecef(vector, &ecef, &m50_ecef, vel_is_ecef);
   else if (converter == NPS_M50_computed)
        m50_wgs84_m50(vector, &ecef, M50_TO_WGS84, NULL, vel_is_ecef);
   else return;

   ecef.time = vector->time;
   *vector = ecef;
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
void nps_j2000_to_m50(State_vector *vector)
{
   State_vector m50;

   /*-------------------------------------------------------------
    */
   state_vector__j2000_to_m50(vector, &m50);

   m50.time = vector->time;
   *vector = m50;
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
void nps_m50_to_j2000(State_vector *vector)
{
```

```
    State_vector j2000;

    /*----------------------------------------------------------------------
     */
    state_vector__m50_to_j2000(vector, &j2000);

    j2000.time = vector->time;
    *vector = j2000;
}
```

```
/*=============================================================
=========================
 *-------------------------------------------------------------*/
#ifndef _NPS_M50_
#define _NPS_M50_

/*=============================================================
=========================
 *-------------------------------------------------------------*/
enum nps_m50_converters { NPS_M50_from_iload, NPS_M50_computed, NPS_M50_cnv_max };

/*-------------------------------------------------------------*/
extern int
   nps_wgs84__read_iload(char *filename);

extern void
   nps_wgs84_to_m50(State_vector *vector, int converter, int vel_is_ecef),
   nps_m50_to_wgs84(State_vector *vector, int converter, int vel_is_ecef),
   nps_m50_to_j2000(State_vector *vector),
   nps_j2000_to_m50(State_vector *vector);

#endif _NPS_M50_
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#include "nps_if.h"

/*=============================================================
============================
 *-------------------------------------------------------------*/
int nps_system_init() { return YES; }

int nps_key_handler(int key) { return NO; }
int nps_display_init() { return NO; }

void nps_data_flush() {}
void nps_system_stop() {}

void nps_data_update(State_vector *orb_ins_m50,
                     Quaternion *orb_quat,
                     State_vector *orb_targ_m50,
                     State_vector *orb_gps_m50,
                     State_vector *orb_gps_ecef,
                     State_vector *spas_sv_ecef,
                     State_vector *spas_gps_ecef)
{
}
```

```
/*=============================================================
=========================
 *-------------------------------------------------------------*/
#include "nps.h"

/*=============================================================
=========================
 *-------------------------------------------------------------*/
extern Nps_graphics_info nps_graphics_info;

/*=============================================================
=========================
 *-------------------------------------------------------------*/
void nps_plot__vlabels(int x, int y, int ystep, Nps_label *label, int cnt)
{
   for ( ; cnt > 0; cnt--, label++)
        outtextxy(x, y + label->id * ystep, label->text);
}

//int nps_plot__next_page()
//{
//   return nps_graphics_info.page = (nps_graphics_info.page ? 0 : 1);
//}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
void nps_plot__clear_view(struct viewporttype *view, int outline)
{
   setcolor(0);
   setviewport(view->left, view->top, view->right, view->bottom, view->clip);
   clearviewport();
   setcolor(7);
   if (outline)  rectangle(0, 0, view->right - view->left, view->bottom - view->top);
}

void nps_plot__set_view(struct viewporttype *view, int outline)
{
   setcolor(7);
   setviewport(view->left, view->top, view->right, view->bottom, view->clip);
   if (outline)  rectangle(0, 0, view->right - view->left, view->bottom - view->top);
}

void nps_plot__main_view()
{
   setviewport(nps_graphics_info.main_view.left, nps_graphics_info.main_view.top,
     nps_graphics_info.main_view.right, nps_graphics_info.main_view.bottom,
     nps_graphics_info.main_view.clip);
}
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#include <stddef.h>
#include <math.h>

#include "nps.h"

/*=============================================================
============================
 *-------------------------------------------------------------*/
static void quick_prop(State_vector *old, double new_time, Perturbations *perts)
{
   State_vector new;
   double dt;
   Orbmech_classicals class;

   getclas(&old->pos, &old->vel, &class, perts);
   f_and_g(&old->pos, &old->vel, &new.pos, &new.vel, &class, new_time - old->time);
   new.time = new_time;  *old = new;
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
int nps_predictor__thrust(Nps_predictor *predictor, State_vector *chaser)
{
   double body_to_m50[3][3];
   Vector dv_m50;

   /*-----------------------------------------------------------
    */
   if (predictor->thrust1.on == NO)  return NO;

   if (predictor->thrust1.delayed)
   {
        if (predictor->thrust1.time < chaser->time)
           return (predictor->thrust1.on =  predictor->thrust1.delayed =  NO);

        quick_prop(chaser, predictor->thrust1.time, predictor->perts);
   }

   if (predictor->thrust1.in_lvlh)
   {
        if (state_vector__itol_matrix(chaser, body_to_m50) == NO)  return NO;
        matrix_transpose(body_to_m50, body_to_m50);
   }
   else
   {
        if (predictor->q_btoi->got_one == NO)  return NO;
```

```
        quaternion_to_matrix(&predictor->q_btoi->quat, body_to_m50);
    }

    /*-----------------------------------------------------------------
     */
    vector_transform(&predictor->thrust1.dv_body, &dv_m50, body_to_m50);

    chaser->vel.x += dv_m50.x;
    chaser->vel.y += dv_m50.y;
    chaser->vel.z += dv_m50.z;

    return YES;
}

/*=================================================================
==========================
 *  The vectors must already have matching time tags.
 *-----------------------------------------------------------------*/
#ifdef NPS_RNDV
int nps_predictor__rndv(Nps_predictor *predictor, State_vector *target_in,
                        State_vector *chaser_in, double ttr)
{
    State_vector target = *target_in;
    State_vector chaser = *chaser_in;
    double mu = predictor->perts->Gravity.GMD.planet_mu;

    Vector xtxvt, hhat, vb, pos_dif, ro, rdot, omega, wxr_term;
    Vector dv1_lvlh, dv1_m50_1, dv1_m50;

    double mag_tpos, mag_tvel, magxxv, magvbar, lcg[3][3], gcl[3][3];
    double T, w, delta;

    /*-----------------------------------------------------------------
     */
    if (predictor->thrust1.delayed)
    {
        quick_prop(&chaser, predictor->thrust1.time, predictor->perts);
        quick_prop(&target, predictor->thrust1.time, predictor->perts);
    }

    /*-----------------------------------------------------------------
     */
    mag_tpos = vector_magnitude(&target.pos);
    mag_tvel = vector_magnitude(&target.vel);

    T = 2*PI*sqrt(mu*mu/(pow(2*mu/mag_tpos - mag_tvel*mag_tvel,3)));
    w = 2*PI/T;

    /*-----------------------------------------------------------------
     */
```

```
lcg[1][0] = target.pos.x / mag_tpos;
lcg[1][1] = target.pos.y / mag_tpos;
lcg[1][2] = target.pos.z / mag_tpos;

vector_cross(&target.pos, &target.vel, &xtxvt);
magxxv = vector_magnitude(&xtxvt);

lcg[2][0] =  hhat.x =  xtxvt.x / magxxv;
lcg[2][1] =  hhat.y =  xtxvt.y / magxxv;
lcg[2][2] =  hhat.z =  xtxvt.z / magxxv;

vector_cross(&target.pos, &hhat, &vb);

magvbar = vector_magnitude(&vb);

lcg[0][0] = vb.x / magvbar;
lcg[0][1] = vb.y / magvbar;
lcg[0][2] = vb.z / magvbar;

pos_dif.x = chaser.pos.x - target.pos.x;
pos_dif.y = chaser.pos.y - target.pos.y;
pos_dif.z = chaser.pos.z - target.pos.z;

vector_transform(&pos_dif, &ro, lcg);

/*-----------------------------------------------------------------
 */
rdot.x = ro.x*sin(w*ttr) + ro.y*(6*w*ttr*sin(w*ttr)-14*(1 - cos(w*ttr)));
delta = 3*w*ttr*sin(w*ttr) - 8*(1 - cos(w*ttr));
rdot.x /= delta;
rdot.y = ro.x+(4*rdot.x-6*ro.y)*sin(w*ttr)+(6*ro.y-3*rdot.x)*w*ttr;
rdot.y = rdot.y/(-2)/(1-cos(w*ttr));
rdot.x *= w;
rdot.y *= w;

rdot.z = -ro.z*w/tan(w*ttr);

omega.x = 0.0;
omega.y = 0.0;
omega.z = w;

vector_cross(&omega, &ro, &wxr_term);

dv1_lvlh.x = rdot.x + wxr_term.x;
dv1_lvlh.y = rdot.y + wxr_term.y;
dv1_lvlh.z = rdot.z + wxr_term.z;

/*-----------------------------------------------------------------
 */
matrix_transpose(lcg, gcl);
```

```
    vector_transform(&dv1_lvlh, &dv1_m50_1, gcl);

    dv1_m50.x = target.vel.x + dv1_m50_1.x - chaser.vel.x;
    dv1_m50.y = target.vel.y + dv1_m50_1.y - chaser.vel.y;
    dv1_m50.z = target.vel.z + dv1_m50_1.z - chaser.vel.z;

    /*-------------------------------------------------------------
     */
    predictor->thrust1.on = YES;

    if (predictor->thrust1.in_lvlh)
    {
        state_vector__itol_matrix(&chaser, lcg);
    }
    else
    {
        quaternion_to_matrix(&predictor->q_btoi->quat, gcl);
        matrix_transpose(gcl, lcg);
    }
    vector_transform(&dv1_m50, &predictor->thrust1.dv_body, lcg);

    return YES;
}
#endif NPS_RNDV
```

```
/*==========================================================================
==========================
 *-------------------------------------------------------------------------*/
#include <math.h>

#include "nps.h"

/*==========================================================================
==========================
 *  This function determines the time of the next plane crossing. It becomes
 * increasingly accurate as the time gets smaller.
 *-------------------------------------------------------------------------*/
double plane_crossing_time(State_vector *chaser, State_vector *target, Perturbations *P)
{
   Vector ref_sv_pos;
   Orbmech_classicals clas_t, clas_c, clas_big, clas_sml;
   double mu, delOM;
   double deli, sin_wpnu, cos_wpnu, wpnu, nu1_sml, nu2_sml, p, e, n;
   double nuo, r1, r2, ro, g1, g2, E1, E2, time, t2;

   /*-----------------------------------------------------------------------
    */
   mu = P->Gravity.GMD.planet_mu;

   getclas(&chaser->pos, &chaser->vel, &clas_c, P);
   getclas(&target->pos, &target->vel, &clas_t, P);

   if (clas_c.i < clas_t.i)
   {
        clas_big = clas_t;
        clas_sml = clas_c;
        ref_sv_pos = chaser->pos;
   }
   else
   {
        clas_big = clas_c;
        clas_sml = clas_t;
        ref_sv_pos = target->pos;
   }

   /*-----------------------------------------------------------------------
    */
   delOM = fabs(clas_t.Om - clas_c.Om);
   deli = fabs(clas_t.i - clas_c.i);
   sin_wpnu = sin(clas_big.i)*sin(delOM)/sin(deli);
   cos_wpnu = (cos(clas_sml.i) - cos(deli)*cos(clas_big.i)) /sin(deli)/sin(clas_big.i);
   wpnu = atan2(sin_wpnu,cos_wpnu);

   nu1_sml = wpnu - clas_sml.om;
   nu2_sml = nu1_sml + PI;
```

```
    /*-----------------------------------------------------------------
     */
    p = clas_sml.a*(1.0 - clas_sml.e*clas_sml.e);
    e = clas_sml.e;
    n = clas_sml.n;
    nuo = 2.0*atan(sqrt((1+e)/(1-e))*tan(clas_sml.Eo/2));
    if (nu1_sml-nuo < 0.0)  nu1_sml += 2.0*PI;
    if (nu2_sml-nuo < 0.0)  nu2_sml += 2.0*PI;

    r1 = p/(1.0+e*cos(nu1_sml));
    r2 = p/(1.0+e*cos(nu2_sml));
    ro = vector_magnitude(&ref_sv_pos);

    g1 = r1*ro*sin(nu1_sml - nuo)/sqrt(mu*p);
    g2 = r2*ro*sin(nu2_sml - nuo)/sqrt(mu*p);

    E1 = 2*atan(sqrt((1-e)/(1+e))*tan(nu1_sml/2));
    E2 = 2*atan(sqrt((1-e)/(1+e))*tan(nu2_sml/2));

    time = g1 - (sin(E1 - clas_sml.Eo) - (E1 - clas_sml.Eo))/n;
    if (time < 0.0)  time += 2*PI/n;

    t2   = g2 - (sin(E2 - clas_sml.Eo) - (E2 - clas_sml.Eo))/n;
    if (t2 < 0.0)   t2 += 2*PI/n;
    if (t2 < time)  time = t2;

    /*-----------------------------------------------------------------
     */
    return time;
}
```

```
/*=============================================================================
=============================
 *-----------------------------------------------------------------------*/
#include <math.h>

#include "nps.h"

/*=============================================================================
=============================
 *-----------------------------------------------------------------------*/
void nps_pyplot__display_update(Nps_pyplot_info *this,
                                State_vector *target, State_vector *orbiter,
                                Quaternion *quat, double t_time, double o_time)
{
   Vector pyd;

   nps_pitch_yaw(quat, target, orbiter, &pyd);
   nps_pyplot__update(this, &pyd, t_time, o_time);
}

void nps_pyplot__display_stats(Nps_pyplot_info *this,
                               State_vector *target, State_vector *orbiter,
                               Quaternion *quat)
{
   Vector pyd;

   nps_pitch_yaw(quat, target, orbiter, &pyd);
   nps_pyplot__stats(this, &pyd, orbiter->time);
}


/*=============================================================================
=============================
 * Returns:
 * 0    Didn't use the key
 * 1    Used the key
 * 2    Used the key, My subsystem is done
 *-----------------------------------------------------------------------*/
int nps_pyplot__key_handler(Nps_pyplot_info *this, int key)
{
   return (key == KEY_esc ? 2 : NO);
}

/*=============================================================================
=============================
 *-----------------------------------------------------------------------*/
void nps_pyplot__display_init(Nps_pyplot_info *this)
{
   nps_pyplot__init(this);
}
```

```
/*=============================================================================
=============================
 *  This function is designed to compute the Pitch/Yaw angles from the orbiter
 * to the target.
 *
 *  Note: I assume both state vectors and the quaternion have been propagated
 * and have matching times.
 *-----------------------------------------------------------------------*/
void nps_pitch_yaw(Quaternion *q_btoi, State_vector *target, State_vector *orb, Vector *pyd)
{
   Quaternion q_itob;
   double m50_to_body[3][3];
   Vector diff, vec_to_target;
   double rho;

   /*-----------------------------------------------------------------------
    */
   q_itob = *q_btoi;
   quaternion_conjugate(&q_itob);
   quaternion_to_matrix(&q_itob, m50_to_body);

   /*-----------------------------------------------------------------------
    */
   diff.x = target->pos.x - orb->pos.x;
   diff.y = target->pos.y - orb->pos.y;
   diff.z = target->pos.z - orb->pos.z;

   vector_transform(&diff, &vec_to_target, m50_to_body);

   rho = sqrt(vec_to_target.x * vec_to_target.x + vec_to_target.z * vec_to_target.z);

   pyd->x = atan2(-vec_to_target.z, vec_to_target.x);      // pitch
   pyd->y = atan2(vec_to_target.y, rho);                   // yaw
   pyd->z = vector_magnitude(&diff);                       // distance(km)
}
```

```
/*=============================================================
==========================
 *-------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>

#include <times.h>
#include "nps.h"

/*=============================================================
==========================
 *-------------------------------------------------------------*/
extern Nps_graphics_info nps_graphics_info;

/*=============================================================
==========================
 *  I use these structures to avoid extra floating-point work during realtime.
 *-------------------------------------------------------------*/
static struct pointtype axis;
static int label_y, clear_pitch_yaw;
static double radius = 150.0;

static struct                          // Plot times
{
   struct viewporttype view;
   int line1_y, line2_y;
} time;

static struct                          // Stats (for other plots)
{
   struct viewporttype view;
   int line1_y, line2_y, line3_y;
} stats;

static struct                          // Pitch info
{
   struct viewporttype view, data_view;
   int center_x;
   struct pointtype view_origin, prev;
} pitch;

static struct                          // Yaw info
{
   struct viewporttype view, data_view;
   int center_x;
   struct pointtype view_origin, prev;
} yaw;

static struct                          // Distance info
{
```

```
    struct viewporttype data_view;
    int center_x;
} dist;

/*=============================================================
=========================
 *-------------------------------------------------------------*/
static void points_init(void)
{
    int xmax = nps_graphics_info.xmax;
    int ymax = nps_graphics_info.ymax;
    int xmid = nps_graphics_info.xmid;
    int c_width = nps_graphics_info.c_width;
    int c_height = nps_graphics_info.c_height;

    /*-------------------------------------------------------------
     */
    time.view.left   = c_width * 8;                    // Top left corner
    time.view.right  = time.view.left + c_width * 16;
    time.view.top    = c_height/8;
    time.view.bottom = time.view.top + c_height * 3;
    time.view.clip   = YES;
    if (time.view.right >= xmax)  time.view.right = xmax - 1;
    if (time.view.bottom >= ymax) time.view.bottom = ymax - 1;

    time.line1_y = c_height;
    time.line2_y = (c_height * 5)/2;

    /*-------------------------------------------------------------
     */
    stats.view.left   = c_width * 3;                   // Top left corner
    stats.view.right  = stats.view.left + c_width * 14;
    stats.view.top    = c_height * 5;
    stats.view.bottom = stats.view.top + c_height * 5;
    stats.view.clip   = YES;
    if (stats.view.right >= xmax)  stats.view.right = xmax - 1;
    if (stats.view.bottom >= ymax) stats.view.bottom = ymax - 1;

    stats.line1_y = c_height;
    stats.line2_y = (c_height * 5)/2;
    stats.line3_y = c_height * 4;

    /*-------------------------------------------------------------
     */
    axis.x = xmid;
    axis.y = ymax * 0.70;

    label_y = axis.y + c_height * 2;

    /*-------------------------------------------------------------
```

```
    */
    pitch.center_x = xmid / 2;
    pitch.data_view.left   = pitch.center_x - c_width * 4;
    pitch.data_view.right  = pitch.center_x + c_width * 4;
    pitch.data_view.top    = label_y + c_height;
    pitch.data_view.bottom = pitch.data_view.top + c_height * 2;

    pitch.view.left   = 20;
    pitch.view.right  = axis.x - 20;
    pitch.view.top    = 35;
    pitch.view.bottom = axis.y - 20;
    pitch.view.clip   = YES;
    pitch.view_origin.x = (pitch.view.right - pitch.view.left) / 2;
    pitch.view_origin.y = (pitch.view.bottom - pitch.view.top) / 2;

    /*-----------------------------------------------------------------
    */
    yaw.center_x = xmid + xmid / 2;
    yaw.data_view.left   = yaw.center_x - c_width * 4;
    yaw.data_view.right  = yaw.center_x + c_width * 4;
    yaw.data_view.top    = label_y + c_height;
    yaw.data_view.bottom = yaw.data_view.top + c_height * 2;

    yaw.view.left   = axis.x + 20;
    yaw.view.right  = xmax - 20;
    yaw.view.top    = 35;
    yaw.view.bottom = axis.y - 20;
    yaw.view.clip   = YES;
    yaw.view_origin.x = (yaw.view.right - yaw.view.left) / 2;
    yaw.view_origin.y = (yaw.view.bottom - yaw.view.top) / 2;

    /*-----------------------------------------------------------------
    */
    dist.center_x = xmid;
    dist.data_view.left   = dist.center_x - c_width * 5;
    dist.data_view.right  = dist.center_x + c_width * 5;
    dist.data_view.top    = label_y + c_height;
    dist.data_view.bottom = dist.data_view.top + c_height * 2;
}

/*=================================================================
=======================
 *-----------------------------------------------------------------*/
typedef struct { struct pointtype p1, p2; } Draw_point;
typedef struct { struct pointtype p1; int a1, a2, r; } Draw_arc;

static void draw_points(Draw_point *this, int count, int x_origin, int y_origin)
{
    for ( ; count > 0; count--, this++)
        line(this->p1.x + x_origin, this->p1.y + y_origin,
```

```
            this->p2.x + x_origin, this->p2.y + y_origin);
}

static void draw_arcs(Draw_arc *this, int count, int x_origin, int y_origin)
{
   for ( ; count > 0; count--, this++)
        arc(this->p1.x + x_origin, this->p1.y + y_origin, this->a1,this->a2, this->r);
}

/*=============================================================
==========================
 *-----------------------------------------------------------*/
static Draw_point pitch_points[] =
{
   {{ 85, 18},{-27, 18}},          // bottom
   {{-33, 12},{-15, -2}},          // slant to canopy
   {{-10, -7},{  0, -7}},  // top
   {{  0, -7},{  0,  0}},  // payload bay (origin)
   {{  0,  0},{ 65,  0}},
   {{ 65,  0},{ 65, -7}},
   {{ 65, -7},{ 80,-27}},          // tail
   {{ 80,-27},{ 90,-27}},
   {{ 90,-27},{ 85, -7}},
   {{ 85, -7},{ 85, 18}}
};
#define PITCH_POINTS (sizeof(pitch_points)/sizeof(Draw_point))

static Draw_arc pitch_arcs[] =
{
   {{-28, 13},170,270,5},          // nose
   {{-10, -2}, 90,180,5} // canopy
};
#define PITCH_ARCS (sizeof(pitch_arcs)/sizeof(Draw_arc))

/*=============================================================
==========================
 *-----------------------------------------------------------*/
static Draw_point yaw_points[] =
{
   {{-40, 69},{-10, 72}},          // left wing tip to eng
   {{-10, 72},{-10, 79}},
   {{-10, 79},{ 10, 79}},          // engines
   {{ 10, 79},{ 10, 72}},
   {{ 10, 72},{ 40, 69}},          // eng to right tip
   {{ -7, 60},{-14, 76}},          // engine outline
   {{-14, 76},{ 14, 76}},
   {{ 14, 76},{  7, 60}},
   {{ 40, 69},{ 40, 61}},          // right side, tail to nose
   {{ 40, 61},{ 37, 54}},
   {{ 37, 54},{ 20, 34}},
```

```
    {{ 20, 34},{ 15, 29}},
    {{ 15, 29},{ 10,  5}},
    {{ 10,  5},{  7,-25}},
    {{  7,-25},{  5,-31}},
    {{ -5,-31},{ -7,-25}}, // left side, nose to tail
    {{ -7,-25},{-10,  5}},
    {{-10,  5},{-15, 29}},
    {{-15, 29},{-20, 34}},
    {{-20, 34},{-37, 54}},
    {{-37, 54},{-40, 61}},
    {{-40, 61},{-40, 69}},
    {{ -8, 60},{  8, 60}}, // payload bay
    {{ -8, 45},{  8, 45}},
    {{ -8, 30},{  8, 30}},
    {{ -8, 15},{  8, 15}},
    {{ -8,  0},{  8,  0}},  // (origin)
    {{ -8, 60},{ -8,-15}},
    {{  8, 60},{  8,-15}}
};
#define YAW_POINTS (sizeof(yaw_points)/sizeof(Draw_point))

static Draw_arc yaw_arcs[] =
{
    {{  0,-29}, 30,150,5} // nose
};
#define YAW_ARCS (sizeof(yaw_arcs)/sizeof(Draw_arc))

/*=============================================================
==========================
 *-------------------------------------------------------------*/
void nps_pyplot__init(Nps_pyplot_info *info)
{
    // int xmax = nps_graphics_info.xmax;
    int c_width = nps_graphics_info.c_width;

    int x1;

    /*-------------------------------------------------------------
     */
    if (time.view.bottom == 0)  points_init();

    cleardevice();

    settextstyle(SMALL_FONT, HORIZ_DIR, nps_graphics_info.font_size);
    settextjustify(CENTER_TEXT,CENTER_TEXT);

    clear_pitch_yaw = NO;

    /*-------------------------------------------------------------
     *   Draw the dividers, labels, and outlines.
```

```
    */
    // line(0,axis.y, xmax,axis.y);
    // line(axis.x,0, axis.x,axis.y);

    outtextxy(pitch.center_x, label_y, "Pitch");
    outtextxy(yaw.center_x, label_y, "Yaw");
    outtextxy(axis.x, label_y, "Distance");

    nps_plot__clear_view(&pitch.view, YES);
    draw_points(pitch_points, PITCH_POINTS, pitch.view_origin.x, pitch.view_origin.y);
    draw_arcs(pitch_arcs, PITCH_ARCS, pitch.view_origin.x, pitch.view_origin.y);
    nps_plot__main_view();

    nps_plot__clear_view(&yaw.view, YES);
    draw_points(yaw_points, YAW_POINTS, yaw.view_origin.x, yaw.view_origin.y);
    draw_arcs(yaw_arcs, YAW_ARCS, yaw.view_origin.x, yaw.view_origin.y);
    nps_plot__main_view();

    /*-----------------------------------------------------------------
     *  Draw time labels and menu bar.
     */
    settextjustify(LEFT_TEXT,CENTER_TEXT);

    x1 = time.view.left - c_width * 8;
    outtextxy(x1, time.view.top + time.line1_y, info->target_title);
    outtextxy(x1, time.view.top + time.line2_y, info->chaser_title);

    nps_menuplot__init(info->menu);
}

/*=================================================================
==========================
 *-----------------------------------------------------------------*/
void nps_pyplot__time(double t1, double t2)
{
    int c_width = nps_graphics_info.c_width;
    char work[20];

    /*-----------------------------------------------------------------
     */
    nps_plot__clear_view(&time.view, NO);
    settextjustify(LEFT_TEXT,CENTER_TEXT);

    t1 += 86400.0;
    time_dbl_to_string(&t1, work);
    outtextxy(c_width/2, time.line1_y, work);

    t2 += 86400.0;
    time_dbl_to_string(&t2, work);
    outtextxy(c_width/2, time.line2_y, work);
```

```
    nps_plot__main_view();
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
void nps_pyplot__update(Nps_pyplot_info *info, Vector *pyd, double t1, double t2)
{
    static char fmt_py[] = "%5.1f", fmt_dist[] = "%6.1f Kft";

    int c_width = nps_graphics_info.c_width;
    int c_height = nps_graphics_info.c_height;

    int x, y;
    char work[24];

    /*-------------------------------------------------------------
     *  Draw the pitch line, clear the previous line first (unless the plot was
     * just initialized).
     */
    nps_plot__set_view(&pitch.view, YES);

    if (clear_pitch_yaw)
    {
        setcolor(0);
        line(pitch.view_origin.x,pitch.view_origin.y, pitch.prev.x,pitch.prev.y);
        setcolor(7);
    }
    draw_points(pitch_points, PITCH_POINTS, pitch.view_origin.x, pitch.view_origin.y);
    draw_arcs(pitch_arcs, PITCH_ARCS, pitch.view_origin.x, pitch.view_origin.y);

    x = pitch.prev.x = pitch.view_origin.x - (int)(radius * cos(pyd->x));
    y = pitch.prev.y = pitch.view_origin.y - (int)(radius * sin(pyd->x));
    line(pitch.view_origin.x,pitch.view_origin.y, x,y);

    nps_plot__main_view();

    /*-------------------------------------------------------------
     *  Draw the yaw line.
     */
    nps_plot__set_view(&yaw.view, YES);

    if (clear_pitch_yaw)
    {
        setcolor(0);
        line(yaw.view_origin.x,yaw.view_origin.x, yaw.prev.x,yaw.prev.y);
        setcolor(7);
    }
    draw_points(yaw_points, PITCH_POINTS, yaw.view_origin.x, yaw.view_origin.y);
    draw_arcs(yaw_arcs, PITCH_ARCS, yaw.view_origin.x, yaw.view_origin.y);
```

```
    x = yaw.prev.x = yaw.view_origin.x + (int)(radius * sin(pyd->y));
    y = yaw.prev.y = yaw.view_origin.y - (int)(radius * cos(pyd->y));
    line(yaw.view_origin.x,yaw.view_origin.x, x,y);

    nps_plot__main_view();

    /*-----------------------------------------------------------------
     *  Draw the pitch/yaw/distance numbers.
     */
    settextjustify(LEFT_TEXT,CENTER_TEXT);

    nps_plot__clear_view(&pitch.data_view, NO);
    sprintf(work, fmt_py, pyd->x * RAD_TO_DEG);
    outtextxy(c_width, c_height, work);
    nps_plot__main_view();

    nps_plot__clear_view(&yaw.data_view, NO);
    sprintf(work, fmt_py, pyd->y * RAD_TO_DEG);
    outtextxy(c_width, c_height, work);
    nps_plot__main_view();

    nps_plot__clear_view(&dist.data_view, NO);
    pyd->z *= KILOM_TO_KILOFT;
    if (pyd->z < 1.0)    { fmt_dist[6] = ' ';  pyd->z *= 1000.0; }
    else                 { fmt_dist[6] = 'K'; }
    sprintf(work, fmt_dist, pyd->z);
    outtextxy(c_width, c_height, work);
    nps_plot__main_view();

    /*-----------------------------------------------------------------
     */
    nps_pyplot__time(t1, t2);

    clear_pitch_yaw = YES;
}

/*=================================================================
=======================
 *-----------------------------------------------------------------*/
void nps_pyplot__stats(Nps_pyplot_info *info, Vector *pyd, double time)
{
    static char fmt_p[] = "Pitch %6.1f", fmt_y[] = "Yaw   %6.1f";

    int c_width = nps_graphics_info.c_width;

    char work[20];

    /*-----------------------------------------------------------------
     */
    if (stats.view.bottom == 0)  points_init();
```

```
    settextjustify(LEFT_TEXT,CENTER_TEXT);

    nps_plot__clear_view(&stats.view, NO);

    sprintf(work, fmt_p, pyd->x * RAD_TO_DEG);
    outtextxy(c_width, stats.line1_y, work);

    sprintf(work, fmt_y, pyd->y * RAD_TO_DEG);
    outtextxy(c_width, stats.line2_y, work);

    time += 86400.0;
    time_dbl_to_string(&time, work);  work[12] = 0;
    outtextxy(c_width, stats.line3_y, work);

    nps_plot__main_view();
}
```

```
/*=============================================================
 *-----------------------------------------------------------*/
#include <alloc.h>
#include <mem.h>
#include <io.h>
#include <fcntl.h>
#include <stdio.h>

#include "nps.h"

/*=============================================================
 *-----------------------------------------------------------*/
int nps_rvplot__data_init(Nps_rvplot_info *this,
                          ushort max_mem, ushort config_mem, Emm_info *emm)
{
   int handle;
   Nps_log_file_header header;

   /*-------------------------------------------------------------
    *  Try to get EMM memory first, and 64K of it.  Otherwise use normal mem.
    */
   if (emm_info_exists(emm))  emm_construct_new(&this->emm, emm, 4);

   if (emm_exists(&this->emm))
   {
        char name[10];

        memcpy(name, this->log_file_name, 8);
        emm_name_set(&this->emm, name);

        this->ring_max = 0xffff / sizeof(RV_vector);
        max_mem = sizeof(RV_vector) * this->ring_max;
        this->ring_buffer = MK_FP(emm->phys.segments[0], 0);

        emm_map_multi(&this->emm, EMM_phys_page, 4, emm->map_0123);
   }

   /*-------------------------------------------------------------
    *  Ring_max may already be initialized, if not then compute it from the
    * max_mem argument.  Allocate the ring buffer.
    */
   else
   {
        if (config_mem > 1024)
           this->ring_max = config_mem / sizeof(RV_vector);

        if (this->ring_max == 0)
           this->ring_max = max_mem / sizeof(RV_vector);
```

```
        this->ring_buffer = malloc(max_mem = sizeof(RV_vector) * this->ring_max);

        if (this->ring_buffer == NULL)  return NO;
    }

    /*-----------------------------------------------------------------
     *  Clear the ring buffer, and try to open my log-file.
     */
    memset(this->ring_buffer, 0, max_mem);

    if ((handle = open(this->log_file_name, O_RDONLY|O_BINARY)) < 0)  return YES;

    /*-----------------------------------------------------------------
     *  Read and verify the log-file header pattern, then fill my ring buffer
     * from the file.  I assume all samples are in time order in the file.
     */
    if (read(handle, &header, sizeof(header)) == sizeof(header)
     && memcmp(&header, this->log_file_hdr, sizeof(header)) == 0)
    {
        this->next_sample = read(handle, this->ring_buffer, max_mem) / sizeof(RV_vector);
        this->prev_sample = this->next_sample - 1;

        if (this->next_sample >= this->ring_max)
        {
           this->next_sample = 0;
           this->ring_full = YES;
        }
        else if (this->next_sample <= 0)
           this->prev_sample =  this->next_sample = 0;
    }

    close(handle);
    return YES;
}

/*================================================================
==========================
 *----------------------------------------------------------------*/
void nps_rvplot__data_stop(Nps_rvplot_info *this)
{
    if (emm_exists(&this->emm))
        emm_destroy(&this->emm);
}

void nps_rvplot__data_save(Nps_rvplot_info *this)
{
    int handle;
    ushort size;

    /*-----------------------------------------------------------------
```

```
    *  Try to open/create the log-file.  Write the header pattern.
    */
   if ((handle = open(this->log_file_name, O_RDWR|O_BINARY|O_CREAT, 0666)) < 0)  return;

   write(handle, this->log_file_hdr, sizeof(Nps_log_file_header));

   /*-----------------------------------------------------------------
    *  I must write the samples in time order.
    */
   if (emm_exists(&this->emm))
        emm_map_multi(&this->emm,  EMM_phys_page, 4, this->emm.info->map_0123);

   if (this->ring_full)
   {
        size = (this->ring_max - this->next_sample) * sizeof(RV_vector);
        if (size)  write(handle, this->ring_buffer + this->next_sample, size);
   }
   size = this->next_sample * sizeof(RV_vector);
   if (size)  write(handle, this->ring_buffer, size);

   _write(handle, 0, 0);  // Truncate file to current size
   close(handle);
}

/*=================================================================
=======================
 *-----------------------------------------------------------------*/
void nps_rvplot__data_flush(Nps_rvplot_info *this)
{
   this->ring_full = NO;
   this->prev_sample =  this->next_sample =  0;
}

/*=================================================================
=======================
 *-----------------------------------------------------------------*/
void nps_rvplot__data_update(Nps_rvplot_info *this)
{
   Nps_state_vector_info *target = this->target;
   Nps_state_vector_info *chaser = this->chaser;
   RV_vector *rv = this->ring_buffer + this->next_sample;

   /*-----------------------------------------------------------------
    */
   if (target->got_one == NO  ||  chaser->got_one == NO)  return;

   if (emm_exists(&this->emm))
        emm_map_multi(&this->emm,  EMM_phys_page, 4, this->emm.info->map_0123);

   vector_rvbar(&target->sv, &chaser->sv, &rv->pos, &this->last_rvdot);
```

```
    rv->time = chaser->sv.time;

    this->prev_sample = this->next_sample;
    this->next_sample++;
    if (this->next_sample >= this->ring_max)
    {
        this->next_sample = 0;
        this->ring_full = YES;
    }
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
void nps_rvplot__display_update(Nps_rvplot_info *this, double t_time, double c_time)
{
    Nps_state_vector_info *target = this->target;
    Nps_state_vector_info *chaser = this->chaser;
    Vector pos;

    /*-----------------------------------------------------------
     */
    if (target->got_one == NO || chaser->got_one == NO)
    {
        nps_rvplot__time(t_time, c_time);
        return;
    }

    /*-----------------------------------------------------------
     */
    if (this->predictor->on == NPS_predict_now) nps_rvplot__display_future(this);

    vector_rvbar(&target->sv, &chaser->sv, &pos, &this->last_rvdot);
    this->crossing_time = plane_crossing_time(&chaser->sv, &target->sv, this->predictor->perts);

    nps_rvplot__point(&this->display, &pos, t_time, c_time,
        (this->ring_full ? this->ring_max : this->next_sample),
        &this->last_rvdot, this->crossing_time);
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static void nps_rvplot__find_origin(Nps_plot_axis *axis, double point)
{
    double step = axis->range/2.0;
    int steps;

    point *= KILOM_TO_KILOFT;
    if (point < axis->origin - step) step = -step;
```

```
    else if (point <= axis->origin + step)  return;

    steps = (int)((point - axis->origin) / step);
    axis->origin += step * steps;
}

/*=============================================================
=========================
 * Returns:
 * 0    Didn't use the key
 * 1    Used the key
 * 2    Used the key, My subsystem is done
 *-------------------------------------------------------------*/
int nps_rvplot__key_handler(Nps_rvplot_info *this, int key)
{
    int rc = YES, refresh = YES;

    Nps_plot_axis *axis_horz = this->display.axis + this->display.axis_horz;
    Nps_plot_axis *axis_vert = this->display.axis + this->display.axis_vert;
    Nps_plot_axis *axis;

    double range_delta;
    RV_vector *sample;

    /*-------------------------------------------------------------
     */
    switch (key)
    {
        case KEY_esc:  return 2;

        case KEY_cr:  break;

        /*-------------------------------------------------------------
         */
        case KEY_up:    axis_vert->range *= 2;  break;
        case KEY_down:  axis_vert->range /= 2;  break;
        case KEY_left:  axis_horz->range /= 2;  break;
        case KEY_right: axis_horz->range *= 2;  break;

        /*-------------------------------------------------------------
         */
        case KEY_cntl_up:
        case KEY_cntl_down:
        case KEY_cntl_left:
        case KEY_cntl_right:
            axis = (key == KEY_cntl_up  ||  key == KEY_cntl_down ? axis_vert : axis_horz);
            range_delta = axis->range/4.0;

            if (key == KEY_cntl_up  ||  key == KEY_cntl_right)
            {
```

```
        if (axis->origin > 0 && axis->origin <= range_delta)
           axis->origin = 0.0;
        else axis->origin -= range_delta;
    }
    else
    {
        if (axis->origin < 0 && -axis->origin <= range_delta)
           axis->origin = 0.0;
        else axis->origin += range_delta;
    }
    break;

/*-----------------------------------------------------------------
 */
case KEY_home:
    this->display.axis[NPS_axis_X].origin =
    this->display.axis[NPS_axis_Y].origin =
    this->display.axis[NPS_axis_Z].origin = 0.0;
    break;

case KEY_end:
    sample = this->ring_buffer + this->prev_sample;
    nps_rvplot__find_origin(&this->display.axis[NPS_axis_X], sample->pos.x);
    nps_rvplot__find_origin(&this->display.axis[NPS_axis_Y], sample->pos.y);
    nps_rvplot__find_origin(&this->display.axis[NPS_axis_Z], sample->pos.z);
    break;

/*-----------------------------------------------------------------
 */
case KEY_space:
    if (this->menu_list[this->menu_id] == NULL) this->menu_id = 0;
    else                                        this->menu_id++;
    break;

/*-----------------------------------------------------------------
 */
case KEY_shft_f6:
    this->plot_type = NPS_rvplot_rv;
    this->display.axis_vert = NPS_axis_Z;
    this->display.axis_horz = NPS_axis_X;
    break;

case KEY_shft_f7:
    this->plot_type = NPS_rvplot_hv;
    this->display.axis_vert = NPS_axis_Y;
    this->display.axis_horz = NPS_axis_X;
    break;

case KEY_shft_f8:
    this->plot_type = NPS_rvplot_rh;
```

```
            this->display.axis_vert = NPS_axis_Z;
            this->display.axis_horz = NPS_axis_Y;
            break;

        case KEY_shft_f9:  this->hbar_on = (this->hbar_on ? NO : YES);  break;

        default:  rc = NO;  break;
    }

    /*-----------------------------------------------------------------
     */
    if (rc == YES  &&  refresh == YES)  nps_rvplot__display_init(this);

    return rc;
}

/*=================================================================
=========================
 *-----------------------------------------------------------------*/
void nps_rvplot__display_init(Nps_rvplot_info *this)
{
    RV_vector *sample;
    ushort samples;

    /*-----------------------------------------------------------------
     */
    if (emm_exists(&this->emm))
        emm_map_multi(&this->emm,  EMM_phys_page, 4, this->emm.info->map_0123);

    sample = this->ring_buffer + this->prev_sample;
    nps_rvplot__init(&this->display, sample, &this->last_rvdot,
      this->menu_list[this->menu_id],  this->plot_type, this->hbar_on,
      this->ring_max, (this->ring_full ? this->ring_max : this->next_sample));

    /*-----------------------------------------------------------------
     *  Figure out how many RV samples are in the ring buffer, then process all
     * the valid samples.
     */
    if (this->ring_full == YES)  samples = this->ring_max;
    else                         samples = this->next_sample;

    /*-----------------------------------------------------------------
     */
    for (sample = this->ring_buffer;  samples > 0;  samples--, sample++)
        nps_rvplot__point_fast(&this->display, &sample->pos);
}

void nps_rvplot__menu_off(Nps_rvplot_info *this)
{
    for (this->menu_id = 0;  this->menu_list[this->menu_id] != NULL;  this->menu_id++);
```

```
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
void nps_rvplot__display_future(Nps_rvplot_info *this)
{
   Nps_state_vector_info *target = this->target;
   Nps_state_vector_info *chaser = this->chaser;
   Nps_predictor *predictor = this->predictor;

   Orbmech_classicals chaser_clas, target_clas;
   int cnt;

   int thrust_on, thrust_waiting, got_rndv_dv;
   State_vector chaser2;
   Orbmech_classicals chaser2_clas;
   double thrust_delay, rndv_dist;
   Vector rndv_dv;

   /*-------------------------------------------------------------
    */
   if (target->got_one == NO || chaser->got_one == NO) return;

   nps_rvplot__display_init(this);

   getclas(&chaser->sv.pos, &chaser->sv.vel, &chaser_clas, predictor->perts);
   getclas(&target->sv.pos, &target->sv.vel, &target_clas, predictor->perts);

   /*-------------------------------------------------------------
    */
   if (predictor->thrust1.on)
   {
        chaser2 = chaser->sv;
        thrust_on = nps_predictor__thrust(predictor, &chaser2);
   }
   else thrust_on = NO;

   if (thrust_on)
   {
        getclas(&chaser2.pos, &chaser2.vel, &chaser2_clas, predictor->perts);
        thrust_delay = chaser2.time - chaser->sv.time;
        thrust_waiting = YES;
        got_rndv_dv = NO; rndv_dist = 800.0 * FT_TO_KILOM;
   }

   /*-------------------------------------------------------------
    */
   for (cnt = 1; cnt <= predictor->steps; cnt++)
   {
```

```
    double dt = cnt * predictor->step;
    State_vector target_new, chaser_new;
    Vector rv_pos, rvdot;
    char work[8];
    double dist;
    int bright = YES;

    /*-----------------------------------------------------------
     */
    if (cnt <= 10  ||  (cnt % 10) == 0)     sprintf(work, "%d", cnt);
    else                                    bright = NO, work[0] = '+', work[1] = 0;

    f_and_g(&chaser->sv.pos, &chaser->sv.vel, &chaser_new.pos, &chaser_new.vel, &chaser_clas, dt);
    f_and_g(&target->sv.pos, &target->sv.vel, &target_new.pos, &target_new.vel, &target_clas, dt);

    vector_rvbar(&target_new, &chaser_new, &rv_pos, &rvdot);
    nps_rvplot__point_text(&this->display, &rv_pos, work, bright);

    /*-----------------------------------------------------------
     */
    if (thrust_on == NO)  continue;

    if (thrust_waiting)
    {
       if (dt < thrust_delay)  continue;
       thrust_waiting = NO;
    }

    if (work[0] == '+')  work[0] = 'o';

    dt -= thrust_delay;
    f_and_g(&chaser2.pos, &chaser2.vel, &chaser_new.pos, &chaser_new.vel, &chaser2_clas, dt);

    vector_rvbar(&target_new, &chaser_new, &rv_pos, &rvdot);
    nps_rvplot__point_text(&this->display, &rv_pos, work, bright);

    /*-----------------------------------------------------------
     */
    if ((dist = vector_magnitude(&rv_pos)) < rndv_dist)
    {
       rndv_dist = dist;
       got_rndv_dv = YES;
       vector_diff(&chaser_new.vel, &target_new.vel, &rndv_dv);
    }
}

/*-----------------------------------------------------------
 */
if (thrust_on  &&  got_rndv_dv)
    nps_rvplot__rndv(&this->display, &rndv_dv, rndv_dist);
```

```
}
```

```
/*=============================================================
==========================
 *
 *  Plot   Vertical   Horizontal
 *  ----   --------   ----------
 *   RV    Rbar z     Vbar x
 *   HV    Hbar y     Vbar x
 *   RH    Rbar z     Hbar y
 *
 *-------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>

#include <times.h>
#include "nps.h"

/*=============================================================
==========================
 *-------------------------------------------------------------*/
extern Nps_graphics_info nps_graphics_info;

/*=============================================================
==========================
 *  I use these structures to avoid extra floating-point work during realtime.
 *-------------------------------------------------------------*/
static struct                           // Plot times
{
   struct viewporttype view;
   int line1_y, line2_y;
   int line3_y, line4_y;
   int desc_color;
} time;

static struct                           // Rbar Vbar Hbar numbers
{
   struct viewporttype view;
   int line1_y, line2_y, line3_y, line4_y, line5_y;
   int label_color, value_color;
} rvh_bar;

static struct                           // Buffer numbers
{
   struct viewporttype view;
   int line1_y, line2_y;
} buffer;

static struct                           // Hbar 'wander' box
{
   struct viewporttype view;
   int hline_x1, hline_x2, hline_y;
```

```
    int vline_x, vline_y1, vline_y2;
    int ns_delta_x, hbar_y;
    int kft_x, kft_y1, kft_y2;
    int cross_x, cross_xt;
    int is_on;
} h_bar;

static struct
{
    int rline_y2;
    struct pointtype label_vert, label_horz;
    int label_color;
} axis;

static struct                                   // Rndv numbers
{
    struct viewporttype view;
    int line1_y, line2_y;
} rndv;

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static int plot_type;

typedef struct                                  // Labels Etc. for all three plots
{
    char label_vert[5], label_horz[5], title[20];
} My_plot_stuff;

static My_plot_stuff plot_stuff[3] =
{
    /* RV */ { "Rbar", "Vbar", "View from the Side" },
    /* HV */ { "Hbar", "Vbar", "View from Below" },
    /* RH */ { "Rbar", "Hbar", "View from Front" }
};

/*=============================================================
========================
 *-------------------------------------------------------------*/
static void points_init(void)
{
    int xmax = nps_graphics_info.xmax;
    int ymax = nps_graphics_info.ymax;
    int xmid = nps_graphics_info.xmid;
    int ymid = nps_graphics_info.ymid;
    int c_width = nps_graphics_info.c_width;
    int c_height = nps_graphics_info.c_height;
    int c_drop = c_height + c_height/2;
    int xsize, ysize;
```

```
/*-------------------------------------------------------------------
 */
// time.view.left   = xmax * 0.05;                    // Bottom left corner
// time.view.top    = ymax * 0.833;

time.view.left   = c_width * 8;                       // Top left corner
time.view.right  = time.view.left + c_width * 16;
time.view.top    = c_height/8;
time.view.bottom = time.view.top + c_height * 3;
time.view.clip   = YES;
if (time.view.right >= xmax)  time.view.right = xmax - 1;
if (time.view.bottom >= ymax) time.view.bottom = ymax - 1;

time.line1_y = c_height;
time.line2_y = time.line1_y + c_drop;
time.line3_y = time.line2_y + c_height*2;
time.line4_y = time.line3_y + c_drop;

time.desc_color = 11;

/*-------------------------------------------------------------------
 */
rvh_bar.view.left   = xmax * 0.875;                   // Bottom right corner
rvh_bar.view.right  = rvh_bar.view.left + c_width * 14;
rvh_bar.view.top    = ymax * 0.790;
if (ymax < 480) rvh_bar.view.top -= c_height * 2;
rvh_bar.view.bottom = rvh_bar.view.top + c_height * 8;
rvh_bar.view.clip   = YES;
if (rvh_bar.view.right >= xmax)  rvh_bar.view.right = xmax - 1;

rvh_bar.line1_y = c_height;
rvh_bar.line2_y = rvh_bar.line1_y + c_drop;
rvh_bar.line3_y = rvh_bar.line2_y + c_drop;
rvh_bar.line4_y = rvh_bar.line3_y + c_drop;
rvh_bar.line5_y = rvh_bar.line4_y + c_drop;

rvh_bar.label_color = 7;
rvh_bar.value_color = 14;

/*-------------------------------------------------------------------
 */
buffer.view.left   = c_width * 5;                     // Bottom left corner
buffer.view.right  = buffer.view.left + c_width * 6;
buffer.view.top    = ymax * 0.865;
buffer.view.bottom = buffer.view.top + c_height * 2;
buffer.view.clip   = YES;
if (buffer.view.right >= xmax)  buffer.view.right = xmax - 1;

buffer.line1_y = c_height;
buffer.line2_y = buffer.line1_y + c_drop;
```

```
/*-----------------------------------------------------------
 */
rndv.view.left   = buffer.view.right + c_width * 8;     // Bottom left
rndv.view.right  = rndv.view.left + c_width * 14;
rndv.view.top    = buffer.view.top;
rndv.view.bottom = rndv.view.top + c_height * 3;
rndv.view.clip   = YES;

rndv.line1_y = buffer.line1_y;
rndv.line2_y = buffer.line2_y;

/*-----------------------------------------------------------
 */
xsize = xmax * 0.25;
ysize = ymax * 0.15;

h_bar.view.left   = xmax - 1 - xsize;          // Top right corner
h_bar.view.right  = xmax - 1;
h_bar.view.top    = 0;
h_bar.view.bottom = ysize;
h_bar.view.clip   = YES;

h_bar.hline_x1  = xsize * 0.0625;                    // 6.25% of box width
h_bar.hline_x2  = h_bar.hline_x1 + xsize * 0.875;
h_bar.hline_y   = ysize * 0.40;

h_bar.vline_x   = xsize * 0.50;
h_bar.vline_y1  = ysize * 0.21;
h_bar.vline_y2  = h_bar.vline_y1 + ysize * 0.25;

h_bar.ns_delta_x = xsize * 0.03125;

h_bar.hbar_y   = ysize * 0.10;
h_bar.kft_y1   = ysize * 0.82 - c_height;
h_bar.kft_y2   = h_bar.kft_y1 + 10;
h_bar.kft_x    = c_width;
h_bar.cross_x  = c_width * 16;
h_bar.cross_xt = h_bar.cross_x + c_width * 4;

/*-----------------------------------------------------------
 */
axis.rline_y2 = ymax * 0.92;                  // R axis is 92% ymax

axis.label_vert.y = axis.rline_y2;
axis.label_horz.x = c_width * 2;
// axis.label_vert.x = xmid + c_width;
// axis.label_horz.y = ymid - c_height;
axis.label_color = 12;
}
```

```
/*=============================================================================
==========================
 *-----------------------------------------------------------------*/
void nps_rvplot__init(Nps_plot_info *info, RV_vector *prev, Vector *prev_rvdot,
                      char **menu, int type, int hbar_on,
                      ushort buffer_max, ushort buffer_cnt)
{
   static char fmt1[] = "%3.1fK", fmt2[] = "%5.3fK";

   int xmax = nps_graphics_info.xmax;
   int xmid = nps_graphics_info.xmid;
   int ymid = nps_graphics_info.ymid;
   int c_width = nps_graphics_info.c_width;
   int c_height = nps_graphics_info.c_height;

   Nps_plot_axis *axis_horz = info->axis + info->axis_horz;
   Nps_plot_axis *axis_vert = info->axis + info->axis_vert;

   int x_origin, y_origin, x_axis_show, y_axis_show;
   int x1, x2, x3, y1, y2, y3, cnt;

   char *fmt, work[32];
   double step, point;
   int x_range_is_ft, y_range_is_ft;

   /*-----------------------------------------------------------------
    */
   if (time.view.bottom == 0)  points_init();

   cleardevice();

   settextstyle(SMALL_FONT, HORIZ_DIR, nps_graphics_info.font_size);

   plot_type = type;

   /*-----------------------------------------------------------------
    *  Draw the Wander box.
    */
   h_bar.is_on = (type == NPS_rvplot_rv  &&  hbar_on == YES);

   if (h_bar.is_on)
   {
        nps_plot__clear_view(&h_bar.view, YES);

        line(h_bar.hline_x1, h_bar.hline_y, h_bar.hline_x2, h_bar.hline_y);
        line(h_bar.vline_x, h_bar.vline_y1, h_bar.vline_x, h_bar.vline_y2);

        settextjustify(CENTER_TEXT,CENTER_TEXT);
        outtextxy(h_bar.hline_x1 + h_bar.ns_delta_x, h_bar.hline_y + c_height, "N");
        outtextxy(h_bar.hline_x2 - h_bar.ns_delta_x, h_bar.hline_y + c_height, "S");
```

```
        outtextxy(h_bar.vline_x, h_bar.hbar_y, "hbar");

        settextjustify(LEFT_TEXT,CENTER_TEXT);
        outtextxy(h_bar.cross_x, h_bar.kft_y1, "crossover");
        outtextxy(h_bar.cross_x, h_bar.kft_y2, " in");

        nps_plot__main_view();
    }

    /*-----------------------------------------------------------------
     *  Draw horizontal/vertical axis.
     *
     *  I pretend I'm drawing in a 1x1 window, but the y-axis is clipped more
     * than the x-axis, thus all scaling is done with the x-axis size.
     *  The X origin determines the location of the Y axis and visa-versa.
     */
    setcolor(7);
    settextjustify(CENTER_TEXT,CENTER_TEXT);

    x_axis_show =  y_axis_show =  YES;

//GIGO
    x_origin = (int)(axis_horz->origin * (double)xmid / axis_horz->range) + xmid;
    if (x_origin >= nps_graphics_info.xmax)
    {
        if (x_origin > nps_graphics_info.xmax)  y_axis_show = NO;
        x_origin = nps_graphics_info.xmax - 1;
    }
    else if (x_origin < 0)  { x_origin = 0;  y_axis_show = NO; }

    y_origin = (int)(-axis_vert->origin * (double)xmid / axis_vert->range) + ymid;
    if (y_origin >= nps_graphics_info.ymax)
    {
        if (y_origin > nps_graphics_info.ymax)  x_axis_show = NO;
        y_origin = nps_graphics_info.ymax - 1;
    }
    else if (y_origin < 0)  { y_origin = 0;  x_axis_show = NO; }

    setlinestyle((y_axis_show ? SOLID_LINE : DOTTED_LINE), 0, NORM_WIDTH);
    line(x_origin, 0, x_origin, axis.rline_y2);

    setlinestyle((x_axis_show ? SOLID_LINE : DOTTED_LINE), 0, NORM_WIDTH);
    line(0, y_origin, xmax-1, y_origin);

    setlinestyle(SOLID_LINE, 0, NORM_WIDTH);

    /*-----------------------------------------------------------------
     *  Draw horizontal axis ticks .
     *
     *  The range tells me how much to show on one axis (from center to edge),
```

```
    * and is divided into four sections.  "Step" is the value of one section.
    *   The range is always in kilo-feet (as are the data points), but I switch
    * the labels to feet when the range grows too small.
    *   The loop makes tick-points and labels travelling out from the center.
    */
//GIGO
   settextjustify(LEFT_TEXT,CENTER_TEXT);

   step = axis_horz->range / 4.0;
   if (step <= 1.0)
   {
        x_range_is_ft = YES;
        fmt = (fabs(axis_horz->origin) <= axis_horz->range ? fmt1 : fmt2);
        fmt[5] = 0;
   }
   else { x_range_is_ft = NO;  fmt = fmt1;  fmt[5] = 'K'; }

   y1 = y_origin - 5;  y2 = y_origin + 5;
   if (y_origin > nps_graphics_info.ymax - 20)    y3 = y1 - c_height;
   else                                           y3 = y2 + c_height;
   setcolor(axis.label_color);
   outtextxy(axis.label_horz.x, y3, plot_stuff[plot_type].label_horz);
   setcolor(7);

   settextjustify(CENTER_TEXT,CENTER_TEXT);

   for (cnt = 0;  cnt <= 3;  cnt++)
   {
        x3 = xmid/4 * cnt;

        if ((x1 = xmid - x3) != x_origin)
        {
           line(x1, y1, x1, y2);
           point = step * cnt + axis_horz->origin;
           if (x_range_is_ft)  point *= 1000.0;
           sprintf(work, fmt, point);  outtextxy(x1, y3, work);
        }

        if (cnt == 0)  continue;

        if ((x2 = xmid + x3) != x_origin)
        {
           line(x2, y1, x2, y2);
           point = step * -cnt + axis_horz->origin;
           if (x_range_is_ft)  point *= 1000.0;
           sprintf(work, fmt, point);  outtextxy(x2, y3, work);
        }
   }

   /*-----------------------------------------------------------------
```

```
    *  Draw vertical axis ticks.
    *
    *  The range on this axis is divided into four sections per side, same as
    * the horizontal axis, but only three of them fit on the screen.
    *  Don't forget that we're pretending to draw in a 1x1 window, but the
    * actual x-axis is larger, so we must scale the points using the x-size.
    */
//GIGO
   settextjustify(LEFT_TEXT,CENTER_TEXT);

   step = axis_vert->range / 4.0;
   if (step < 1.0)
   {
        y_range_is_ft = YES;
        fmt = (fabs(axis_vert->origin) <= axis_vert->range ? fmt1 : fmt2);
        fmt[5] = 0;
   }
   else { y_range_is_ft = NO;  fmt = fmt1;  fmt[5] = 'K'; }

   x1 = x_origin - 5;  x2 = x_origin + 5;
   if (x_origin > nps_graphics_info.xmax - 20)    x3 = x1 - c_width * 6;
   else                                           x3 = x2 + c_width;
   setcolor(axis.label_color);
   outtextxy(x3, axis.label_vert.y, plot_stuff[plot_type].label_vert);
   setcolor(7);

   for (cnt = 0;  cnt <= 2;  cnt++)
   {
        y3 = xmid/4 * cnt;

        if ((y1 = ymid - y3) != y_origin)
        {
           line(x1, y1, x2, y1);
           point = step * -cnt + axis_vert->origin;
           if (y_range_is_ft)  point *= 1000.0;
           sprintf(work, fmt, point);  outtextxy(x3, y1, work);
        }

        if (cnt == 0)  continue;

        if ((y2 = ymid + y3) != y_origin)
        {
           line(x1, y2, x2, y2);
           point = step * cnt + axis_vert->origin;
           if (y_range_is_ft)  point *= 1000.0;
           sprintf(work, fmt, point);  outtextxy(x3, y2, work);
        }
   }

   /*-----------------------------------------------------------------
```

```
    *  Draw number box, time labels, and menu bar.
    */
   settextjustify(LEFT_TEXT,CENTER_TEXT);

   x1 = rvh_bar.view.left - c_width * 4;
   setcolor(rvh_bar.label_color);
   outtextxy(x1, rvh_bar.view.top + rvh_bar.line1_y, "vbar");
   outtextxy(x1, rvh_bar.view.top + rvh_bar.line2_y, "rbar");
   outtextxy(x1, rvh_bar.view.top + rvh_bar.line3_y, "hbar");
   outtextxy(x1, rvh_bar.view.top + rvh_bar.line4_y, "dist");
   outtextxy(x1, rvh_bar.view.top + rvh_bar.line5_y, "rate");
   setcolor(7);

   x1 = buffer.view.left - c_width * 5;
   outtextxy(x1, buffer.view.top + buffer.line1_y, "bufr");
   outtextxy(x1, buffer.view.top + buffer.line2_y, " of ");
   x1 = buffer.view.left + c_width/2;
   sprintf(work, "%4u", buffer_max);
   outtextxy(x1, buffer.view.top + buffer.line2_y, work);

   x1 = time.view.left - c_width * 8;
   outtextxy(x1, time.view.top + time.line1_y, info->target_title);
   outtextxy(x1, time.view.top + time.line2_y, info->chaser_title);

   setcolor(time.desc_color);
   x1 += c_width * 4;
   outtextxy(x1, time.view.top + time.line3_y, plot_stuff[plot_type].title);
   setcolor(7);

   if (menu != NULL)  nps_menuplot__init(menu);

   /*-------------------------------------------------------------------
    *  Draw the previous known time and data values.
    */
   nps_rvplot__values(prev->pos.x * KILOM_TO_KILOFT, prev->pos.y * KILOM_TO_KILOFT,
     prev->pos.z * KILOM_TO_KILOFT, prev_rvdot->x * KILOM_TO_KILOFT,
     prev_rvdot->z * KILOM_TO_KILOFT, buffer_cnt);

   nps_rvplot__time(prev->time, prev->time);
}

/*==========================================================================
==========================
 *-------------------------------------------------------------------------*/
void nps_rvplot__time(double t1, double t2)
{
   int c_width = nps_graphics_info.c_width;
   char work[20];

   /*-------------------------------------------------------------------
```

```
    */
   nps_plot__clear_view(&time.view, NO);
   settextjustify(LEFT_TEXT,CENTER_TEXT);

   t1 += 86400.0;
   time_dbl_to_string(&t1, work);
   outtextxy(c_width/2, time.line1_y, work);

   t2 += 86400.0;
   time_dbl_to_string(&t2, work);
   outtextxy(c_width/2, time.line2_y, work);

   nps_plot__main_view();
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
void nps_rvplot__values(double x, double y, double z,
                        double d, double rdot, ushort buffer_cnt)
{
   static char fmt_rv[] = "%8.3f Kft/s";

   int c_width = nps_graphics_info.c_width;

   char work[32];

   /*-------------------------------------------------------------
    */
   nps_plot__clear_view(&rvh_bar.view, NO);
   settextjustify(LEFT_TEXT,CENTER_TEXT);
   setcolor(rvh_bar.value_color);

   /*-------------------------------------------------------------
    */
   fmt_rv[9] = 0;

   if (fabs(x) < 1.0)     { fmt_rv[6] = ' '; x *= 1000.0; }
   else                   { fmt_rv[6] = 'K'; }
   sprintf(work, fmt_rv, x);
   outtextxy(c_width/2, rvh_bar.line1_y, work);

   if (fabs(z) < 1.0)     { fmt_rv[6] = ' '; z *= 1000.0; }
   else                   { fmt_rv[6] = 'K'; }
   sprintf(work, fmt_rv, z);
   outtextxy(c_width/2, rvh_bar.line2_y, work);

   if (fabs(y) < 1.0)     { fmt_rv[6] = ' '; y *= 1000.0; }
   else                   { fmt_rv[6] = 'K'; }
   sprintf(work, fmt_rv, y);
```

```
    outtextxy(c_width/2, rvh_bar.line3_y, work);

    if (fabs(d) < 1.0)      { fmt_rv[6] = ' '; d *= 1000.0; }
    else                    { fmt_rv[6] = 'K'; }
    sprintf(work, fmt_rv, d);
    outtextxy(c_width/2, rvh_bar.line4_y, work);

    /*-----------------------------------------------------------------
     */
    fmt_rv[9] = '/';

    if (fabs(rdot) < 1.0)  { fmt_rv[6] = ' '; rdot *= 1000.0; }
    else                   { fmt_rv[6] = 'K'; }
    sprintf(work, fmt_rv, rdot);
    outtextxy(c_width/2, rvh_bar.line5_y, work);

    /*-----------------------------------------------------------------
     */
    setcolor(7);
    nps_plot__clear_view(&buffer.view, NO);

    sprintf(work, "%4u", buffer_cnt);
    outtextxy(c_width/2, buffer.line1_y, work);

    nps_plot__main_view();
}

/*=================================================================
=======================
 *-----------------------------------------------------------------*/
void nps_rvplot__hbar(double y, double ydot, double crossing_time)
{
    static char fmt_h[] = "%8.3f Kft/s";
    static char ft[20], fps[24], time[20];
    static int old_y;

    int c_height = nps_graphics_info.c_height;
    int mins, secs;

    /*-----------------------------------------------------------------
     */
    setviewport(h_bar.view.left, h_bar.view.top, h_bar.view.right,
      h_bar.view.bottom, h_bar.view.clip);

    settextjustify(CENTER_TEXT,CENTER_TEXT);

    setcolor(0);
    outtextxy(h_bar.vline_x + old_y, h_bar.hline_y - c_height, "|");

    setcolor(7);
```

```
    old_y = (int)(y * 100);
    outtextxy(h_bar.vline_x + old_y, h_bar.hline_y - c_height, "|");

    /*-----------------------------------------------------------------
     */
    settextjustify(LEFT_TEXT,CENTER_TEXT);

    setcolor(0);
    outtextxy(h_bar.kft_x, h_bar.kft_y1, ft);
    outtextxy(h_bar.kft_x, h_bar.kft_y2, fps);
    outtextxy(h_bar.cross_xt, h_bar.kft_y2, time);

    setcolor(7);

    fmt_h[9] = 0;
    if (fabs(y) < 1.0)      { fmt_h[6] = ' '; y *= 1000.0; }
    else                    { fmt_h[6] = 'K'; }
    sprintf(ft, fmt_h, y);
    outtextxy(h_bar.kft_x, h_bar.kft_y1, ft);

    fmt_h[9] = '/';
    if (fabs(ydot) < 1.0) { fmt_h[6] = ' '; ydot *= 1000.0; }
    else                  { fmt_h[6] = 'K'; }
    sprintf(fps, fmt_h, ydot);
    outtextxy(h_bar.kft_x, h_bar.kft_y2, fps);

    /*-----------------------------------------------------------------
     */
    mins = crossing_time / 60.0;
    secs = crossing_time - (mins * 60);
    sprintf(time, "%02.2d:%02.2d", mins, secs);
    outtextxy(h_bar.cross_xt, h_bar.kft_y2, time);

    nps_plot__main_view();
}

/*=================================================================
=======================
 *-----------------------------------------------------------------*/
void nps_rvplot__point(Nps_plot_info *info, Vector *pos,
                       double t1, double t2, ushort buffer_cnt,
                       Vector *rvdot, double crossing_time)
{
//GIGO
    int xmid = nps_graphics_info.xmid;
    int ymid = nps_graphics_info.ymid;

    double x = pos->x * KILOM_TO_KILOFT;
    double y = pos->y * KILOM_TO_KILOFT;
    double z = pos->z * KILOM_TO_KILOFT;
```

```
    double dist = rvdot->x * KILOM_TO_KILOFT;
    double ydot = rvdot->y * KILOM_TO_KILOFT;
    double rdot = rvdot->z * KILOM_TO_KILOFT;
    double horz, vert;

    Nps_plot_axis *axis_horz = info->axis + info->axis_horz;
    Nps_plot_axis *axis_vert = info->axis + info->axis_vert;

    int xpoint, ypoint;

    /*-------------------------------------------------------------------
     */
    nps_rvplot__values(x, y, z, dist, rdot, buffer_cnt);

    nps_rvplot__time(t1, t2);

    if (h_bar.is_on)  nps_rvplot__hbar(y, ydot, crossing_time);

    /*-------------------------------------------------------------------
     */
    horz = (plot_type == NPS_rvplot_rh ? y : x);
    vert = (plot_type == NPS_rvplot_hv ? y : z);

    xpoint = (int)(-(horz-axis_horz->origin) * (double)xmid / axis_horz->range) + xmid;
    ypoint = (int)((vert-axis_vert->origin) * (double)xmid / axis_vert->range) + ymid;
    if (xpoint >= 0  &&  xpoint < nps_graphics_info.xmax
     && ypoint >= 0  &&  ypoint < nps_graphics_info.ymax)  putpixel(xpoint, ypoint, 7);
}

/*=========================================================================
==========================
 *-------------------------------------------------------------------*/
void nps_rvplot__point_fast(Nps_plot_info *info, Vector *pos)
{
    int xmid = nps_graphics_info.xmid;
    int ymid = nps_graphics_info.ymid;

    double x = pos->x * KILOM_TO_KILOFT;
    double y = pos->y * KILOM_TO_KILOFT;
    double z = pos->z * KILOM_TO_KILOFT;
    double horz, vert;

    Nps_plot_axis *axis_horz = info->axis + info->axis_horz;
    Nps_plot_axis *axis_vert = info->axis + info->axis_vert;

    int xpoint, ypoint;

    /*-------------------------------------------------------------------
     */
    settextjustify(CENTER_TEXT,CENTER_TEXT);
```

```
    setcolor(7);

    horz = (plot_type == NPS_rvplot_rh ? y : x);
    vert = (plot_type == NPS_rvplot_hv ? y : z);

    xpoint = (int)(-(horz-axis_horz->origin) * (double)xmid / axis_horz->range) + xmid;
    ypoint = (int)((vert-axis_vert->origin) * (double)xmid / axis_vert->range) + ymid;
    if (xpoint >= 0 && xpoint < nps_graphics_info.xmax
     && ypoint >= 0 && ypoint < nps_graphics_info.ymax)  putpixel(xpoint, ypoint, 7);
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
void nps_rvplot__point_text(Nps_plot_info *info, Vector *pos, char *text, int bright)
{
    int xmid = nps_graphics_info.xmid;
    int ymid = nps_graphics_info.ymid;

    double x = pos->x * KILOM_TO_KILOFT;
    double y = pos->y * KILOM_TO_KILOFT;
    double z = pos->z * KILOM_TO_KILOFT;
    double horz, vert;

    Nps_plot_axis *axis_horz = info->axis + info->axis_horz;
    Nps_plot_axis *axis_vert = info->axis + info->axis_vert;

    int xpoint, ypoint;

    /*-------------------------------------------------------------
     */
    settextstyle(SMALL_FONT, HORIZ_DIR, 2);
    settextjustify(CENTER_TEXT,CENTER_TEXT);
    setcolor(bright ? rvh_bar.value_color : 7);

    horz = (plot_type == NPS_rvplot_rh ? y : x);
    vert = (plot_type == NPS_rvplot_hv ? y : z);

    xpoint = (int)(-(horz-axis_horz->origin) * (double)xmid / axis_horz->range) + xmid;
    ypoint = (int)((vert-axis_vert->origin) * (double)xmid / axis_vert->range) + ymid;
    if (xpoint >= 0 && xpoint < nps_graphics_info.xmax
     && ypoint >= 0 && ypoint < nps_graphics_info.ymax)
    {
        outtextxy(xpoint, ypoint, text);
    }
    settextstyle(SMALL_FONT, HORIZ_DIR, nps_graphics_info.font_size);
    setcolor(7);
}

/*=============================================================
```

```
=========================
 *-------------------------------------------------------------------*/
void nps_rvplot__rndv(Nps_plot_info *info, Vector *rndv_dv, double rndv_dist)
{
   static char fmt_dv[] = "%7.3f Kft/s";

   int c_width = nps_graphics_info.c_width;

   double dv;
   char work[32];

   /*-------------------------------------------------------------
    */
   outtextxy(rndv.view.left - c_width * 3, rndv.view.top + rndv.line1_y, "RNDV");

   nps_plot__clear_view(&rndv.view, NO);
   settextjustify(LEFT_TEXT,CENTER_TEXT);
   setcolor(7);

   rndv_dist *= KILOM_TO_KILOFT;  fmt_dv[9] = 0;
   if (fabs(rndv_dist) < 1.0)      { fmt_dv[6] = ' ';  rndv_dist *= 1000.0; }
   else                            { fmt_dv[6] = 'K';  }
   sprintf(work, fmt_dv, rndv_dist);
   outtextxy(c_width/2, rndv.line1_y, work);

   dv = vector_magnitude(rndv_dv) * KILOM_TO_KILOFT;  fmt_dv[9] = '/';
   if (fabs(dv) < 1.0)             { fmt_dv[6] = ' ';  dv *= 1000.0; }
   else                            { fmt_dv[6] = 'K';  }
   sprintf(work, fmt_dv, dv);
   outtextxy(c_width/2, rndv.line2_y, work);

   nps_plot__main_view();
}
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#include <stdio.h>
#include <mem.h>

#include <edit.h>
#include <com_port.h>
#include <packet.h>
#include "sim.h"

/*=============================================================
============================
 *-----------------------------------------------------------*/
static int port_id, send_orb_gps;

/*=============================================================
============================
 *-----------------------------------------------------------*/
int gen_init()
{
   ushort tx_size = 512;
   char *answer;

   /*-------------------------------------------------------------
    */
   port_id = edit_integer("Enter output port id ", 1);
   /* tx_size = edit_integer("       Enter tx size ", 512); */

   if (com_port_start(port_id, 0, tx_size) == NO)
   {
        printf("Can't start port %d\n", port_id);
        port_id = 0;
        return NO;
   }

   /*-------------------------------------------------------------
    */
   answer = edit_text("Send Orbiter-GPS packets too ? ", "y");
   send_orb_gps = (answer[0] == 'y' || answer[0] == 'Y');

   return YES;
}

/*=============================================================
============================
 *-----------------------------------------------------------*/
void gen_stop()
{
   if (port_id)  com_port_stop(port_id);
```

```
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
static void state_vector_scale(State_vector *in, State_vector *out, double scale, double dtime)
{
   out->pos.x = in->pos.x * scale;
   out->pos.y = in->pos.y * scale;
   out->pos.z = in->pos.z * scale;
   out->vel.x = in->vel.x * scale;
   out->vel.y = in->vel.y * scale;
   out->vel.z = in->vel.z * scale;
   out->time = in->time + dtime;
}

/*=============================================================
=========================
 *  The packet has start sync eb 90, stop sync BE EF, and types 1 and 2.
 *-------------------------------------------------------------*/
void gen_save_sample(Sim_sample *sample)
{
   struct
   {
        Orb_packet orb;
        Spas_packet spas;
        Orb_gps_packet gps;
   } packet;

   ubyte *where = (ubyte *)&packet;
   ushort bytes = sizeof(packet);

   /*-----------------------------------------------------------------
    */
   memset(&packet, 0, sizeof(packet));

   packet.orb.header.type = PKT_orbiter;
   packet.orb.header.sync_start[0] = 0xeb;
   packet.orb.header.sync_start[1] = 0x90;
   packet.orb.trailer.sync_stop[0] = 0xbe;
   packet.orb.trailer.sync_stop[1] = 0xef;

   state_vector_scale(&sample->orb_ins, &packet.orb.orbiter_vector, KILOM_TO_FT, 86400.0);
   state_vector_scale(&sample->orb_targ, &packet.orb.target_vector, KILOM_TO_FT, 86400.0);
   packet.orb.orbiter_quat = sample->orb_quat;
   packet.orb.orbiter_quat.time += 86400.0;

   /*-----------------------------------------------------------------
    */
   packet.spas.header.type = PKT_spas;
```

```
    packet.spas.header.sync_start[0] = 0xeb;
    packet.spas.header.sync_start[1] = 0x90;
    packet.spas.trailer.sync_stop[0] = 0xbe;
    packet.spas.trailer.sync_stop[1] = 0xef;

    state_vector_scale(&sample->spas_sv, &packet.spas.spas_vector, KILOM_TO_FT, 86400.0);
    state_vector_scale(&sample->spas_gps, &packet.spas.gps_vector, KILOM_TO_FT, 86400.0);

    /*-----------------------------------------------------------------
     */
    if (send_orb_gps)
    {
        packet.gps.header.type = PKT_gps;
        packet.gps.header.sync_start[0] = 0xeb;
        packet.gps.header.sync_start[1] = 0x90;
        packet.gps.trailer.sync_stop[0] = 0xbe;
        packet.gps.trailer.sync_stop[1] = 0xef;

        state_vector_scale(&sample->orb_gps_ecef, &packet.gps.gps_vector, 1000.0, 86400.0);
    }
    else bytes -= sizeof(Orb_gps_packet);

    /*-----------------------------------------------------------------
     */
    while (bytes > 0)
    {
        if (com_port_put(port_id, *where) == NO) continue;
        bytes--; where++;
        /* if (kbhit()) break; */
    }
}
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#include <stdio.h>
#include <mem.h>
#include <io.h>
#include <fcntl.h>

#include <edit.h>
#include <packet.h>
#include "sim.h"

/*=============================================================
============================
 *-------------------------------------------------------------*/
static int port_file, port_status, send_orb_gps;

#define FILE_IS_DEVICE      0x0080
#define FILE_IS_RAW   0x0020
#define FILE_SET_BITS 0x00ff

/*=============================================================
============================
 *-------------------------------------------------------------*/
int gen_init()
{
   char *filename, *answer;

   /*-------------------------------------------------------------
    */
   filename = edit_text("Enter output port/file:", "com1");

   if ((port_file = open(filename, O_RDWR|O_BINARY|O_CREAT, 0666)) < 0)
   {
        printf("Can't open port/file %s\n", filename);
        return NO;
   }

   /*-------------------------------------------------------------
    *  If output file is a device, place that device in "raw" mode.
    */
   port_status = ioctl(port_file, 0, 0,0);

   if (port_status & FILE_IS_DEVICE)
   {
        port_status = (port_status & FILE_SET_BITS) | FILE_IS_RAW;
        ioctl(port_file, 1, port_status,0);
   }

   /*-------------------------------------------------------------
```

```
    */
    answer = edit_text("Send Orbiter-GPS packets too ? ", "y");
    send_orb_gps = (answer[0] == 'y' || answer[0] == 'Y');

    return YES;
}

/*==========================================================================
==========================
 *-------------------------------------------------------------------------*/
void gen_stop()
{
    /*-----------------------------------------------------------
     *  If output file is a device, place that device back in "cooked" mode.
     */
    if (port_status & FILE_IS_DEVICE)
    {
        port_status &= ~FILE_IS_RAW;
        ioctl(port_file, 1, port_status,0);
    }

    close(port_file);
}

/*==========================================================================
==========================
 *-------------------------------------------------------------------------*/
static void state_vector_scale(State_vector *in, State_vector *out, double scale, double dtime)
{
    out->pos.x = in->pos.x * scale;
    out->pos.y = in->pos.y * scale;
    out->pos.z = in->pos.z * scale;
    out->vel.x = in->vel.x * scale;
    out->vel.y = in->vel.y * scale;
    out->vel.z = in->vel.z * scale;
    out->time = in->time + dtime;
}

/*==========================================================================
==========================
 *  The packet has start sync eb 90, stop sync BE EF, and types 1 and 2.
 *-------------------------------------------------------------------------*/
void gen_save_sample(Sim_sample *sample)
{
    struct
    {
        Orb_packet orb;
        Spas_packet spas;
        Orb_gps_packet gps;
    } packet;
```

```
ubyte *where = (ubyte *)&packet;
ushort bytes = sizeof(packet);
int sent;

/*-----------------------------------------------------------------
 */
memset(&packet, 0, sizeof(packet));

packet.orb.header.type = PKT_orbiter;
packet.orb.header.sync_start[0] = 0xeb;
packet.orb.header.sync_start[1] = 0x90;
packet.orb.trailer.sync_stop[0] = 0xbe;
packet.orb.trailer.sync_stop[1] = 0xef;

state_vector_scale(&sample->orb_ins, &packet.orb.orbiter_vector, KILOM_TO_FT, 86400.0);
state_vector_scale(&sample->orb_targ, &packet.orb.target_vector, KILOM_TO_FT, 86400.0);
packet.orb.orbiter_quat = sample->orb_quat;
packet.orb.orbiter_quat.time += 86400.0;

/*-----------------------------------------------------------------
 */
packet.spas.header.type = PKT_spas;
packet.spas.header.sync_start[0] = 0xeb;
packet.spas.header.sync_start[1] = 0x90;
packet.spas.trailer.sync_stop[0] = 0xbe;
packet.spas.trailer.sync_stop[1] = 0xef;

state_vector_scale(&sample->spas_sv, &packet.spas.spas_vector, KILOM_TO_FT, 86400.0);
state_vector_scale(&sample->spas_gps, &packet.spas.gps_vector, KILOM_TO_FT, 86400.0);

/*-----------------------------------------------------------------
 */
if (send_orb_gps)
{
    packet.gps.header.type = PKT_gps;
    packet.gps.header.sync_start[0] = 0xeb;
    packet.gps.header.sync_start[1] = 0x90;
    packet.gps.trailer.sync_stop[0] = 0xbe;
    packet.gps.trailer.sync_stop[1] = 0xef;

    state_vector_scale(&sample->orb_gps_ecef, &packet.gps.gps_vector, 1000.0, 86400.0);
}
else bytes -= sizeof(Orb_gps_packet);

/*-----------------------------------------------------------------
 */
while (bytes > 0)
{
    sent = _write(port_file, where, bytes);
    if (sent < 0) break;
```

```
        if (sent > 0) { bytes -= sent;  where += sent; }
        /* if (kbhit())  break; */
    }
}
```

```
/*=============================================================
=========================
 *-------------------------------------------------------------*/
#include <stdio.h>

#include <edit.h>
#include "sim.h"

/*=============================================================
=========================
 *-------------------------------------------------------------*/
static FILE *outfile;

/*=============================================================
=========================
 *-------------------------------------------------------------*/
int gen_init()
{
   char *filename;

   /*-----------------------------------------------------------
    */
   filename = edit_text("Enter output filename:", "sim.dat");

   if ((outfile = fopen(filename, "wb")) == NULL)
   {
        printf("Can't open %s\n", filename);
        return NO;
   }

   return YES;
}

void gen_stop()
{
   fclose(outfile);
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
void gen_save_sample(Sim_sample *sample)
{
   fwrite(sample, sizeof(Sim_sample),1, outfile);
}
```

```
/*=============================================================
==========================
 *-------------------------------------------------------------*/
#include "sim.h"

/*=============================================================
==========================
 *-------------------------------------------------------------*/
int gen_init() { return YES; }

void gen_stop() { }

void gen_save_sample(Sim_sample  *sample) { }
```

```
/*=============================================================
=========================
 *-------------------------------------------------------------*/
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <bios.h>
#include <time.h>

#include "sim.h"
#include <keys.h>
#include <nps_if.h>

extern unsigned _stklen = 15000U;                    //larger stack

/*=============================================================
=========================
 *-------------------------------------------------------------*/
static int
   main_key_handler(int key);

static void
   key_process(void),
   main_display_init(void),
   main_display_status(void);

/*=============================================================
=========================
 *-------------------------------------------------------------*/
static int (*my_key_handler)(int) = main_key_handler;

static int terminate = NO;

/*=============================================================
=========================
 *-------------------------------------------------------------*/
void main(int argc, char **argv)
{
   /*-----------------------------------------------------------
    */
   clrscr();
   if (sim_init(argc, argv) == NO)
   {
        puts("sim_init() failed during init");
        return;
   }

   if (nps_system_init() == NO)
   {
        puts("nps_system_init() failed during init");
```

```
        return;
    }

    if (nps_display_init() == YES)
        my_key_handler = nps_key_handler;
    else main_display_init();

    /*-----------------------------------------------------------------
     */
    for (;;)
    {
        static Sim_sample sample;
        int rc;

        /*-----------------------------------------------------------------
         *  Check for key hit, process the key, see if somebody wants to stop
         * the program.
         */
        if (key_hit())
        {
           key_process();
           if (terminate)  break;
        }

        /*-----------------------------------------------------------------
         *  Get next sample, call all data update routines.  The _next_sample
         * routine must return one of these values:
         *        -1  Still trying to get a sample, call me again
         *         0  Done, no more samples available (usually from disk_sim)
         *         1  Ok, got the sample
         */
        if ((rc = sim_next_sample(&sample)) < 1)  continue;

        nps_data_update(&sample.orb_ins, &sample.orb_quat, &sample.orb_targ,
          &sample.orb_gps_m50, &sample.orb_gps_ecef,
          &sample.spas_sv, &sample.spas_gps);
    }

    nps_system_stop();
    sim_stop();
}

/*=================================================================
=======================
 *-----------------------------------------------------------------*/
static void key_process()
{
    int rc, key = key_get();

    rc = (*my_key_handler)(key);
```

```
    if (rc == 2)
    {
        my_key_handler = main_key_handler;
        main_display_init();
    }
    if (rc == 0)  sim_key_handler(key);
}

/*=============================================================
============================
 * Returns:
 * 0    Didn't use the key
 * 1    Used the key
 * 2    Used the key, My subsystem is done
 *-------------------------------------------------------------*/
static int main_key_handler(int key)
{
    if (key == 'y'  ||  key == 'Y')  terminate = YES;
    else if (nps_display_init() == YES)  my_key_handler = nps_key_handler;
    return YES;
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
typedef struct { int col, row; char *text; } Label;

static void main_display_init()
{
    clrscr();
    gotoxy(10, 6);
    cputs("Exit program (y/n) ? ");
}
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#include <stddef.h>

#include <packet.h>
#include <pkt_if.h>
#include "sim.h"
#include "../nps/nps_m50.h"

/*=============================================================
============================
 *  These structures are owned by the orb_port_process() routine, and are in
 * the original units: orbiter vectors are M50 in ft and ft/s, spas vectors are
 * WGS84 in ft and ft/s.
 *
 *  The orbiter time is absolute, 001/00:00:01.000 == 1.0 sec + 86400 secs (one
 * day).  The spas time is also absolute.
 *-----------------------------------------------------------*/
extern Orb_packet orbiter_packet;
extern Spas_packet spas_packet;

/*=============================================================
============================
 *  This structure is also owned by the orb_port_process() routine, and is only
 * used for testing.  The vector is WGS84 in m and m/s.  Its time is absolute.
 *-----------------------------------------------------------*/
extern Orb_gps_packet orbiter_gps_packet;

/*=============================================================
============================
 *-----------------------------------------------------------*/
int sim_init(int argc, char **argv)
{
   return (argc > 1  &&  argv[1] != NULL
     ? orb_port_init_byname(argv[1])
     : orb_port_init());
}

/*=============================================================
============================
 *-----------------------------------------------------------*/
void sim_stop() { orb_port_stop(); }

/*=============================================================
============================
 *-----------------------------------------------------------*/
static void state_vector_scale(State_vector  *in, State_vector *out, double scale, double dtime)
{
   out->pos.x = in->pos.x * scale;
```

```
    out->pos.y = in->pos.y * scale;
    out->pos.z = in->pos.z * scale;
    out->vel.x = in->vel.x * scale;
    out->vel.y = in->vel.y * scale;
    out->vel.z = in->vel.z * scale;
    out->time = in->time + dtime;
}

/*=============================================================
============================
 *  I try to fill in my packet buffer, piece by piece if necessary.  I may not
 * get all the bytes I need in one read, so I must wait for them to arrive.
 *
 *  But I should not hold up other processing just because I can't get enough
 * bytes from my port.  So I tell my caller that I'm not done yet, and that I
 * must be called again.
 *
 *  I return one of these values:
 *       -1  Still trying to get a sample, call me again
 *        1  Ok, got the sample
 *-------------------------------------------------------------------------*/
int sim_next_sample(Sim_sample *sample)
{
    int type;
    if ((type = orb_port_process()) == NO) return -1;

    if (type == PKT_orbiter)
    {
        state_vector_scale(&orbiter_packet.orbiter_vector,  &sample->orb_ins, FT_TO_KILOM, -86400.0);
        state_vector_scale(&orbiter_packet.target_vector,  &sample->orb_targ, FT_TO_KILOM, -86400.0);
        sample->orb_quat = orbiter_packet.orbiter_quat;
        sample->orb_quat.time -= 86400.0;
    }
#   ifdef NPS_GPS
    else if (type == PKT_spas)
    {
        state_vector_scale(&spas_packet.spas_vector,  &sample->spas_sv, FT_TO_KILOM, -86400.0);
        state_vector_scale(&spas_packet.gps_vector,  &sample->spas_gps, FT_TO_KILOM, -86400.0);
    }
    else  /* type == PKT_gps */
    {
        state_vector_scale(&orbiter_gps_packet.gps_vector,  &sample->orb_gps_ecef, 0.001, -86400.0);
        state_vector_scale(&orbiter_gps_packet.gps_vector,  &sample->orb_gps_m50,  0.001, -86400.0);
        nps_wgs84_to_m50(&sample->orb_gps_m50, NPS_M50_computed, YES);
    }
#   endif NPS_GPS
    return YES;
}

/*=============================================================
```

```
==========================*
 *-------------------------------------------------------------*/
int sim_key_handler(int key) { return YES; }

int sim_edit_display_init() { return NO; }
int sim_edit_key_handler(int key) { return YES; }
```

```
/*=============================================================
==========================
 *-----------------------------------------------------------*/
#include <stdlib.h>

#include <edit_rt.h>
#include <times.h>
#include <keys.h>

#include <packet.h>
#include <pkt_if.h>
#include "sim.h"
#include "../nps/nps_m50.h"

/*=============================================================
==========================
 *  These structures are owned by the orb_port_process() routine, and are in
 * the original units: orbiter vectors are M50 in ft and ft/s, spas vectors are
 * WGS84 in ft and ft/s.
 *
 *  The orbiter time is absolute, 001/00:00:01.000 == 1.0 sec + 86400 secs (one
 * day).  The spas time is also absolute.
 *-----------------------------------------------------------*/
extern Orb_packet orbiter_packet;
extern Spas_packet spas_packet;

/*=============================================================
==========================
 *  This structure is also owned by the orb_port_process() routine, and is only
 * used for testing.  The vector is WGS84 in m and m/s.  Its time is absolute.
 *  Note: this structure is also filled in by the tans_port_process() routine.
 *-----------------------------------------------------------*/
extern Orb_gps_packet orbiter_gps_packet;

/*=============================================================
==========================
 *-----------------------------------------------------------*/
static int sim_delay_flag = 1, sim_delay_ms = 1000, sim_delay_save = 1000;

typedef struct { int got, send; double time; } Packet_info;

static Packet_info orb, gps;

/*=============================================================
==========================
 *-----------------------------------------------------------*/
int sim_init(int argc, char **argv)
{
   /*-------------------------------------------------------
    */
```

```
    argv++; argc--;

    if (argc > 0)
    {
        if (orb_port_init_byname(*argv++) == NO)  return NO;
        argc--;
    }
    else if (orb_port_init() == NO)  return NO;

    /*-----------------------------------------------------------------
     */
#   ifdef NPS_GPS
    if (argc > 0)
    {
        if (tans_port_init_byname(*argv++) == NO)  return NO;
        argc--;
    }
    else if (tans_port_init() == NO)  return NO;
#   endif NPS_GPS

    /*-----------------------------------------------------------------
     */
    if (argc > 0)  sim_delay_ms =  sim_delay_save =  atoi(*argv);

    return YES;
}

/*==========================================================
==========================
 *-----------------------------------------------------------------*/
void sim_stop()
{
    orb_port_stop();
#   ifdef NPS_GPS
    tans_port_stop();
#   endif NPS_GPS
}

/*==========================================================
==========================
 *-----------------------------------------------------------------*/
static void state_vector_scale(State_vector  *in, State_vector *out, double scale, double dtime)
{
    out->pos.x = in->pos.x * scale;
    out->pos.y = in->pos.y * scale;
    out->pos.z = in->pos.z * scale;
    out->vel.x = in->vel.x * scale;
    out->vel.y = in->vel.y * scale;
    out->vel.z = in->vel.z * scale;
    out->time = in->time + dtime;
```

```
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
static void sim_copy_packet(int type, Sim_sample *sample)
{
   /*-------------------------------------------------------
    */
   if (type == PKT_orbiter)
   {
        state_vector_scale(&orbiter_packet.orbiter_vector, &sample->orb_ins, FT_TO_KILOM, -86400.0);
        state_vector_scale(&orbiter_packet.target_vector, &sample->orb_targ, FT_TO_KILOM, -86400.0);
        sample->orb_quat = orbiter_packet.orbiter_quat;
        sample->orb_quat.time -= 86400.0;
   }
#  ifdef NPS_GPS
   else if (type == PKT_spas)
   {
        state_vector_scale(&spas_packet.spas_vector, &sample->spas_sv, FT_TO_KILOM, -86400.0);
        state_vector_scale(&spas_packet.gps_vector, &sample->spas_gps, FT_TO_KILOM, -86400.0);
   }
   else if (type == PKT_gps)
   {
        state_vector_scale(&orbiter_gps_packet.gps_vector, &sample->orb_gps_ecef, 0.001, -86400.0);
        state_vector_scale(&orbiter_gps_packet.gps_vector, &sample->orb_gps_m50, 0.001, -86400.0);
        nps_wgs84_to_m50(&sample->orb_gps_m50, NPS_M50_computed, YES);
   }
#  endif NPS_GPS
}

/*=============================================================
============================
 *  I return one of these values:
 *       -1  Still trying to get a sample, call me again
 *        1  Ok, got the sample
 *-------------------------------------------------------------*/
int sim_next_sample(Sim_sample *sample)
{
   /*-------------------------------------------------------
    */
   if (sim_delay_ms)
   {
        if (sim_delay_flag == 0) return -1;
        sim_delay_flag = 0;
        rtc_timer_set(sim_delay_ms, &sim_delay_flag);
   }

   /*-------------------------------------------------------
    */
```

```
   if (orb.got == NO)
   {
        orb.got = orb_port_process();

#       ifdef NPS_GPS
        if (orb.got == PKT_orbiter)
           orb.time = orbiter_packet.orbiter_vector.time;

        else if (orb.got == PKT_spas)
           orb.time = spas_packet.gps_vector.time;
#       endif NPS_GPS
   }

#  ifdef NPS_GPS
   if (gps.got == NO)
   {
        gps.got = tans_port_process();

        if (gps.got == PKT_gps)
           gps.time = orbiter_gps_packet.gps_vector.time;
   }
#  endif NPS_GPS

   /*-----------------------------------------------------------------
    */
   orb.send =  gps.send =  NO;

#  ifdef NPS_GPS
   if (orb.got  &&  gps.got)
   {
        if (orb.time < gps.time) orb.send = YES;
        else                                gps.send = YES;
   }
   else
   {
        if (gps.got)       gps.send = YES;
        if (orb.got)       orb.send = YES;
   }

#  else
        if (orb.got)       orb.send = YES;
#  endif NPS_GPS

   /*-----------------------------------------------------------------
    */
   if (orb.send == NO  &&  gps.send == NO)  return -1;

   if (orb.send)
   {
        sim_copy_packet(orb.got, sample);
```

```
        orb.got = NO;
    }

#   ifdef NPS_GPS
    if (gps.send)
    {
        sim_copy_packet(gps.got, sample);
        gps.got = NO;
    }
#   endif NPS_GPS

    return YES;
}

/*=============================================================
============================
 * Returns:
 * 0    Didn't use the key
 * 1    Used the key
 * 2    Used the key, My subsystem is done
 *-------------------------------------------------------------*/
int sim_key_handler(int key)
{
    if (key == KEY_alt_w)       sim_delay_ms = 0;
    else if (key == KEY_alt_t)  sim_delay_ms = sim_delay_save;
    return YES;
}

/*=============================================================
============================
 *--123456789 123456789 123456789 123456789 123456789 123456789
 * 1                SIM Settings
 * 2
 * 3  Realtime delay (ms): X
 * 4
 * 5  ALT-W  Warp mode (ignore delay)
 * 6  ALT-T  Timed mode (use delay)
 *-------------------------------------------------------------*/
static int temp_delay_ms;

static void screen1_pre(Edit_screen *screen)
{
    temp_delay_ms = sim_delay_save;
}

static void screen1_post(Edit_screen *screen)
{
    if (sim_delay_ms == sim_delay_save)  sim_delay_ms = temp_delay_ms;
    sim_delay_save = temp_delay_ms;
}
```

```
static Edit_label labels1[] =
{
   { 1,20, "SIM Settings"},
   { 3, 4, "Realtime delay (ms):"},
   { 5, 4, "ALT-W  Warp mode (ignore delay)"},
   { 6, 4, "ALT-T  Timed mode (use delay)"}
};
#define LABEL1_COUNT (sizeof(labels1)/sizeof(Edit_label))

static Edit_field fields1[] =
{
   { 3,25, edit_field_get_int,   edit_field_put_int,   edit_field_key_int,   &temp_delay_ms }
};
#define FIELD1_COUNT (sizeof(fields1)/sizeof(Edit_field))

Edit_screen screen1 =
{
   LABEL1_COUNT, labels1,
   FIELD1_COUNT, fields1,
   NULL, NO, screen1_pre, screen1_post
};

/*=============================================================
============================
 *-----------------------------------------------------------*/
static Edit_screen *screens[] =
{
   &screen1
};
#define MAX_SCREENS (sizeof(screens)/sizeof(Edit_screen *))

static int current_screen;

/*=============================================================
============================
 *-----------------------------------------------------------*/
int sim_edit_display_init()
{
   edit_screen_init(screens[current_screen]);
   return YES;
}

/*=============================================================
============================
 *-----------------------------------------------------------*/
int sim_edit_key_handler(int key)
{
   int rc = edit_screen_key_handler(screens[current_screen],  key);

   if (rc > 0)  return rc;
```

```
    /*-----------------------------------------------------------
     */
    switch (key)
    {
         case KEY_pgup:
            if (--current_screen < 0)  current_screen = MAX_SCREENS - 1;
            break;

         case KEY_pgdn:
            if (++current_screen >= MAX_SCREENS)  current_screen = 0;
            break;

         default:  return NO;
    }

    edit_screen_init(screens[current_screen]);
    return YES;
}
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#include <stdio.h>
#include <io.h>
#include <fcntl.h>
#include <conio.h>
#include <mem.h>

#include <edit.h>
#include <packet.h>
#include "sim.h"

/*=============================================================
============================
 *-----------------------------------------------------------*/
static int port_file, port_status;

#define FILE_IS_DEVICE      0x0080
#define FILE_NOT_EOF        0x0040
#define FILE_IS_RAW   0x0020
#define FILE_SET_BITS 0x00ff

static int check_sync(ubyte *buffer, int bytes, ubyte *sync);

/*=============================================================
============================
 *-----------------------------------------------------------*/
int sim_init()
{
   char *filename;

   /*-------------------------------------------------------------
    */
   filename = edit_text("Enter input port/file:", "com1");

   if ((port_file = open(filename, O_RDONLY|O_BINARY)) < 0)
   {
        printf("Can't open port/file %s\n", filename);
        return NO;
   }

   /*-------------------------------------------------------------
    *  If input file is a device, place that device in "raw" mode.
    */
   port_status = ioctl(port_file, 0, 0,0);

   if (port_status & FILE_IS_DEVICE)
   {
        port_status = (port_status & FILE_SET_BITS) | FILE_IS_RAW | FILE_NOT_EOF;
```

```
        ioctl(port_file, 1, port_status,0);
    }

    return YES;
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
void sim_stop()
{
    /*-------------------------------------------------------------
     *  If input file is a device, place that device back in "cooked" mode.
     */
    if (port_status & FILE_IS_DEVICE)
    {
        port_status &= ~FILE_IS_RAW;
        ioctl(port_file, 1, port_status,0);
    }

    close(port_file);
}

/*=============================================================
============================
 *  DOS sucks.  DOS has a bug regarding devices in raw mode, this piece of code
 * fixes the error caused by the bug.  The bug occurs under these conditions:
 *      - You are reading from a device in raw mode.
 *      - The memory you are reading into just happens to fall on a paragraph
 *      boundary (i.e. the address is divisible by 16).
 *      - The device driver doesn't have any bytes ready for you.
 *
 *  Under these conditions, DOS will erroneously mark your file with an EOF
 * flag, after which read() will always fail, even if the device has some bytes
 * ready.  This kludge marks the file as not-EOF.
 *-------------------------------------------------------------*/
static void fix_dos_eof_bug(void *location)
{
    if (((ushort)location & 15) == 0  &&  port_status & FILE_IS_DEVICE)
    {
        port_status |= FILE_NOT_EOF;
        ioctl(port_file, 1, port_status,0);
    }
}

/*=============================================================
============================
 *  I try to fill in my packet buffer, piece by piece if necessary.  I may not
 * get all the bytes I need in one read, so I must wait for them to arrive.
 *
```

```
 *  But I should not hold up other processing just because I can't get enough
 * bytes from my port.  So I tell my caller that I'm not done yet, and that I
 * must be called again.
 *
 *  I return one of these values:
 *       -1  Still trying to get a sample, call me again
 *        1  Ok, got the sample
 *-----------------------------------------------------------------------*/
int sim_next_sample(Sim_sample *sample)
{
   static union
   {
        Packet_header header;
        Orb_packet orb;
        Spas_packet spas;
        ubyte buffer[48];
   } data;

   enum my_states { STATE_find_header, STATE_read_packet };

   static ubyte *where = data.buffer;
   static int bytes = sizeof(Packet_header);
   static int state = STATE_find_header;

   static ubyte sync1[2] = {0xeb,0x90}, sync2[2] = {0xbe,0xef};

   /*-----------------------------------------------------------------
    *  Loop until I'm done or I can't read any bytes.
    */
   for (;;)
   {
        int got;

        /*-------------------------------------------------------------
         *  Try to read as many bytes as I need to complete the buffer.  If I
         * get any bytes at all, then bump the buffer pointer and drop the
         * bytes-remaining count.  Return if I can't get enough on this pass.
         */
        if (bytes)
        {
           if ((got = read(port_file, where, bytes)) <= 0)
           {
                fix_dos_eof_bug(where);
                break;
           }

           where += got;  bytes -= got;

           if (bytes > 0)  break;
        }
```

```
/*-----------------------------------------------------------
 *  If I'm in the _find_header state, then check the leading sync and
 * type, and get the remaining packet size.
 */
if (state == STATE_find_header)
{
   /*-----------------------------------------------------
    *  The sync is there, figure out how many more bytes to read, and
    * switch to the next state.
    */
   if (((ushort *)data.header.sync_start)[0] == ((ushort *)sync1)[0])
   {
        bytes = (data.header.type == PKT_orbiter
          ? sizeof(Orb_packet)
          : sizeof(Spas_packet));
        bytes -= (where - data.buffer);
        state = STATE_read_packet;
   }

   /*-----------------------------------------------------------
    *  The sync is not there, I'll search the byte buffer for sync and
    * try to recover a partial header.
    *  The find_sync() routine searches for sync and saves the bytes
    * that are valid, it returns the number of valid bytes.
    */
   else
   {
        got = check_sync(data.buffer, where - data.buffer, sync1);
        where = data.buffer + got;
        if (got)
        {
           bytes = sizeof(Packet_header) - got;
           if (bytes < 0)  bytes = 0;
        }
        else bytes = sizeof(Packet_header);
   }
   continue;
}

/*-----------------------------------------------------------
 *  If I'm in the _read_packet state, then I've got the entire packet.
 * Check the trailing sync, extract the usefull data and return OK.
 */
else
{
   Packet_trailer *trailer;
   int pkt_size;

   /*-----------------------------------------------------------
    *  The trailing sync location depends on the packet type.
```

```
 */
if (data.header.type == PKT_orbiter)
{
     trailer = &data.orb.trailer;
     pkt_size = sizeof(Orb_packet);
}
else
{
     trailer = &data.spas.trailer;
     pkt_size = sizeof(Spas_packet);
}

state = STATE_find_header;

/*-----------------------------------------------------------------
 *  The sync is there, extract data depending on the packet type.
 */
if (((ushort *)trailer->sync_stop)[0] == ((ushort *)sync2)[0])
{
     static int first_time = YES;

     if (data.header.type == PKT_orbiter)
     {
        sample->ins.pos.x = data.orb.orbiter_vector.pos.x * FT_TO_KILOM;
        sample->ins.pos.y = data.orb.orbiter_vector.pos.y * FT_TO_KILOM;
        sample->ins.pos.z = data.orb.orbiter_vector.pos.z * FT_TO_KILOM;
        sample->ins.vel.x = data.orb.orbiter_vector.vel.x * FT_TO_KILOM;
        sample->ins.vel.y = data.orb.orbiter_vector.vel.y * FT_TO_KILOM;
        sample->ins.vel.z = data.orb.orbiter_vector.vel.z * FT_TO_KILOM;
        sample->ins.time = data.orb.orbiter_vector.time - 86400.0;

        sample->quat = data.orb.orbiter_quat;

        if (first_time)
        {
             sample->tans = sample->ins;
             sample->spas = sample->ins;
             first_time = NO;
        }
     }
     else
     {
        sample->spas.pos.x = data.spas.spas_vector.pos.x * FT_TO_KILOM;
        sample->spas.pos.y = data.spas.spas_vector.pos.y * FT_TO_KILOM;
        sample->spas.pos.z = data.spas.spas_vector.pos.z * FT_TO_KILOM;
        sample->spas.vel.x = data.spas.spas_vector.vel.x * FT_TO_KILOM;
        sample->spas.vel.y = data.spas.spas_vector.vel.y * FT_TO_KILOM;
        sample->spas.vel.z = data.spas.spas_vector.vel.z * FT_TO_KILOM;
        sample->spas.time = data.spas.spas_vector.time - 86400.0;
```

```
                if (first_time)
                {
                    sample->tans = sample->spas;
                    sample->ins = sample->spas;
                    sample->quat.time = sample->spas.time;
                    first_time = NO;
                }
            }
            where = data.buffer;
            bytes = sizeof(Packet_header);
            return 1;
        }

        /*-----------------------------------------------------
         *  The sync is not there, I'll search the byte buffer for sync and
         * try to recover a partial packet.
         */
        got = check_sync(data.buffer, pkt_size, sync1);
        if (got  &&  got != pkt_size)
        {
            where = data.buffer + got;
            bytes = sizeof(Packet_header) - got;
            if (bytes < 0)  bytes = 0;
        }
        else
        {
            where = data.buffer;
            bytes = sizeof(Packet_header);
        }
    }
}

return -1;
}

/*=============================================================
=========================
 *  I search for sync and save the bytes that are valid, I return the number of
 * valid bytes.  I assume sync is two bytes.
 *-------------------------------------------------------------*/
static int check_sync(ubyte *buffer, int bytes, ubyte *sync)
{
    ubyte *where;

    for (where = buffer;  bytes > 1;  where++, bytes--)
    {
        if (((ushort *)where)[0] == ((ushort *)sync)[0])
        {
            memcpy(buffer, where, bytes);
            return bytes;
```

```
        }
    }

    if (*where == *sync)
    {
        *buffer = *where;
        return bytes;
    }

    return 0;
}

/*=============================================================================
============================
 *-----------------------------------------------------------------------------*/
int sim_key_handler(int key) { return YES; }

int sim_edit_display_init() { return NO; }
int sim_edit_key_handler(int key) { return YES; }
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#include <stdio.h>

#include <edit.h>
#include <edit_rt.h>
#include <times.h>
#include <keys.h>
#include "sim.h"

/*=============================================================
============================
 *-----------------------------------------------------------*/
static FILE *infile;

static int sim_delay_flag = 1, sim_delay_ms = 0;

/*=============================================================
============================
 *-----------------------------------------------------------*/
int sim_init(int argc, char **argv)
{
   char *filename;

     ? orb_port_init_byname(argv[1])
     : orb_port_init());

   filename = (argc > 1  &&  argv[1] != NULL
     ? filename = argv[1]
     : edit_text("Enter input filename:", "sim.dat"));

   if ((infile=fopen(filename, "rb")) == NULL)
   {
        printf("Can't open %s\n", filename);
        return NO;
   }
   return YES;
}

/*=============================================================
============================
 *-----------------------------------------------------------*/
void sim_stop()
{
   fclose(infile);
}

/*=============================================================
============================
```

```
*----------  ----------*/
int sim_next_sample(Sim_sample *sample)
{
   /*----------
    */
   if (sim_delay_ms)
   {
        if (sim_delay_flag == 0)  return -1;
        sim_delay_flag = 0;
        rtc_timer_set(sim_delay_ms, &sim_delay_flag);
   }

   /*----------
    */
   if (fread(sample, sizeof(Sim_sample),1, infile) != 1)  return NO;
   return YES;
}

/*==========
==========
 *----------*/
int sim_key_handler(int key) { return YES; }

/*==========
==========
 *--123456789 123456789 123456789 123456789 123456789 123456789
 * 1                SIM Settings
 * 2
 * 3   Realtime delay (ms): X
 *----------*/
static Edit_label labels1[] =
{
   { 1,20, "SIM Settings"},
   { 3, 4, "Realtime delay (ms):"}
};
#define LABEL1_COUNT (sizeof(labels1)/sizeof(Edit_label))

static Edit_field fields1[] =
{
   { 3,25, edit_field_get_int,   edit_field_put_int,   edit_field_key_int,   &sim_delay_ms }
};
#define FIELD1_COUNT (sizeof(fields1)/sizeof(Edit_field))

Edit_screen screen1 =
{
   LABEL1_COUNT, labels1,
   FIELD1_COUNT, fields1,
   NULL, NO, NULL, NULL
};
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
static Edit_screen *screens[] =
{
   &screen1
};
#define MAX_SCREENS (sizeof(screens)/sizeof(Edit_screen *))

static int current_screen;

/*=============================================================
============================
 *-------------------------------------------------------------*/
int sim_edit_display_init()
{
   edit_screen_init(screens[current_screen]);
   return YES;
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
int sim_edit_key_handler(int key)
{
   int rc = edit_screen_key_handler(screens[current_screen], key);

   if (rc > 0)  return rc;

   /*-----------------------------------------------------------
    */
   switch (key)
   {
        case KEY_pgup:
           if (--current_screen < 0)  current_screen = MAX_SCREENS - 1;
           break;

        case KEY_pgdn:
           if (++current_screen >= MAX_SCREENS)  current_screen = 0;
           break;

        default:  return NO;
   }

   edit_screen_init(screens[current_screen]);
   return YES;
}
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#include <stdio.h>

#include <edit.h>
#include "sim.h"

/*=============================================================
============================
 *-------------------------------------------------------------*/
static FILE *infile1, *infile2;

/*=============================================================
============================
 *-------------------------------------------------------------*/
int sim_init()
{
   char *filename1, *filename2;

   filename1 = edit_text("Enter input filename:", "sim.dat");
   if ((infile1=fopen(filename1, "rb")) == NULL)
   {
        printf("Can't open %s\n", filename1);
        return NO;
   }

   filename2 = edit_text("Enter input filename:", "tans_sa.dat");
   if ((infile2=fopen(filename2, "r")) == NULL)
   {
        printf("Can't open %s\n", filename2);
        return NO;
   }

   return YES;
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
void sim_stop()
{
   fclose(infile1);
   fclose(infile2);
}

/*=============================================================
============================
 * WARNING:
 *  I assume the ORB_GPS and ORB_INS vectors are propagated more often than the
```

```
 * SPAS vector.  I use the very first SPAS time as the base/starting time.
 *-------------------------------------------------------------------*/
int sim_next_sample(Sim_sample *sample)
{
   static double initial_time;
   static first_time = YES;

   int cnt;

   /*-----------------------------------------------------------------
    */
   if (fread(sample, sizeof(Sim_sample),1, infile1) != 1)  return NO;

   cnt = fscanf(infile2,"%lf %lf %lf %lf %lf %lf %lf", &sample->orb_gps_m50.time,
     &sample->orb_gps_m50.pos.x, &sample->orb_gps_m50.pos.y, &sample->orb_gps_m50.pos.z,
     &sample->orb_gps_m50.vel.x, &sample->orb_gps_m50.vel.y, &sample->orb_gps_m50.vel.z);

   if (cnt < 7)  return NO;

   /*-----------------------------------------------------------------
    */
   if (first_time)
   {
        initial_time = sample->spas.time;
        first_time = NO;
   }

   sample->orb_gps.time += initial_time;

   return YES;
}

/*==================================================================
=======================
 *-------------------------------------------------------------------*/
int sim_key_handler(int key) { return YES; }

int sim_edit_display_init() { return NO; }
int sim_edit_key_handler(int key) { return YES; }
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#include <stdio.h>

#include <edit.h>
#include "sim.h"

/*=============================================================
============================
 *-------------------------------------------------------------*/
static FILE *infile1, *infile2;

/*=============================================================
============================
 *-------------------------------------------------------------*/
int sim_init()
{
   char *filename1, *filename2;

   filename1 = edit_text("Enter input filename:", "sim.dat");
   if ((infile1=fopen(filename1, "rb")) == NULL)
   {
        printf("Can't open %s\n", filename1);
        return NO;
   }

   filename2 = edit_text("Enter input filename:", "tans_sa.dat");
   if ((infile2=fopen(filename2, "r")) == NULL)
   {
        printf("Can't open %s\n", filename2);
        return NO;
   }

   return YES;
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
void sim_stop()
{
   fclose(infile1);
   fclose(infile2);
}

/*=============================================================
============================
 * WARNING:
 *  I assume the ORB_GPS and ORB_INS vectors are propagated more often than the
```

```
 * SPAS vector.  I use the very first SPAS time as the base/starting time.
 *-------------------------------------------------------------------------*/
int sim_next_sample(Sim_sample *sample)
{
   static double initial_time;
   static first_time = YES;

   int cnt;

   /*-----------------------------------------------------------------------
    */
   if (fread(sample, sizeof(Sim_sample),1, infile1) != 1)  return NO;

   cnt = fscanf(infile2,"%lf %lf %lf %lf %lf %lf %lf", &sample->orb_gps_m50.time,
     &sample->orb_gps_m50.pos.x, &sample->orb_gps_m50.pos.y, &sample->orb_gps_m50.pos.z,
     &sample->orb_gps_m50.vel.x, &sample->orb_gps_m50.vel.y, &sample->orb_gps_m50.vel.z);

   if (cnt < 7)  return NO;

   /*-----------------------------------------------------------------------
    */
   if (first_time)
   {
        initial_time = sample->spas.time;
        first_time = NO;
   }

   sample->orb_gps.time += initial_time;

   return YES;
}

/*=========================================================================
=========================
 *-------------------------------------------------------------------------*/
int sim_key_handler(int key) { return YES; }

int sim_edit_display_init() { return NO; }
int sim_edit_key_handler(int key) { return YES; }
```

```
/*=============================================================
==========================
 *-------------------------------------------------------------*/
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <bios.h>
#include <time.h>

#include "sim.h"
#include <keys.h>
#include <nps_if.h>

extern unsigned _stklen = 15000U;                //larger stack

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static int
   main_key_handler(int key);

static void
   key_process(void),
   main_display_init(void),
   main_display_status(void);

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static int (*my_key_handler)(int) = main_key_handler;

static int terminate = NO, suspended = YES;

/*=============================================================
==========================
 *-------------------------------------------------------------*/
void main(int argc, char **argv)
{
   long cnt;
   int got_stop_time;
   long start_time, stop_time;
   double run_time;

   /*-----------------------------------------------------------
    */
   clrscr();
   if (sim_init(argc, argv) == NO)
   {
        puts("sim_init() failed during init");
        return;
```

```
}

if (nps_system_init() == NO)
{
      puts("nps_system_init() failed during init");
      return;
}

if (gen_init() == NO)
{
      puts("gen_init() failed during init");
      return;
}

start_time = biostime(0, 0);
got_stop_time = NO;

main_display_init();

/*-----------------------------------------------------------------------
 */
for (cnt = 0; ; )
{
      static Sim_sample sample;
      int rc;

      /*-----------------------------------------------------------------
       *  Check for key hit, process the key, see if somebody wants to stop
       * the program.
       */
      if (key_hit())
      {
         key_process();
         if (terminate)  break;
      }

      if (suspended)  continue;

      /*-----------------------------------------------------------------
       *  Get next sample, call all data update routines.  The _next_sample
       * routine must return one of these values:
       *        -1  Still trying to get a sample, call me again
       *         0  Done, no more samples available (usually from disk_sim)
       *         1  Ok, got the sample
       */
      if ((rc = sim_next_sample(&sample)) < 1)
      {
         if (rc < 0)  continue;
         if (got_stop_time == NO)
         {
```

```
                stop_time = biostime(0, 0);
                got_stop_time = YES;
            }
            continue;
        }

        cnt++;
        nps_data_update(&sample.orb_ins, &sample.orb_quat, &sample.orb_targ,
          &sample.orb_gps_m50, &sample.orb_gps_ecef,
          &sample.spas_sv, &sample.spas_gps);
        gen_save_sample(&sample);

        /*-----------------------------------------------------------------
         *  This chunk is, in effect, the main_data_update() routine.
         */
        if (my_key_handler == main_key_handler)
        {
            gotoxy(11, 20); cprintf("%7ld", cnt);
        }
    }

    if (got_stop_time == NO)  stop_time = biostime(0, 0);
    run_time = (stop_time - start_time) / CLK_TCK;

    gen_stop();
    nps_system_stop();
    sim_stop();

    printf("\n\nDid %ld vectors in %lf secs,\n", cnt, run_time);
    printf("  That's %lf vectors per sec\n", (double)cnt / run_time);
}

/*=============================================================================
============================
 *-----------------------------------------------------------------------*/
static void key_process()
{
    int rc, key = key_get();

    rc = (*my_key_handler)(key);

    if (rc == 2)
    {
        my_key_handler = main_key_handler;
        main_display_init();
    }
    if (rc == 0)
    {
        if (key == KEY_alt_s)
        {
```

```
            suspended = (suspended ? NO : YES);
            main_display_status();
        }
        else sim_key_handler(key);
    }
}

/*=============================================================
============================
 * Returns:
 * 0    Didn't use the key
 * 1    Used the key
 * 2    Used the key, My subsystem is done
 *-------------------------------------------------------------*/
static int main_key_handler(int key)
{
    switch (key)
    {
        case KEY_esc:
            terminate = YES;
            return YES;

        case KEY_f1:
            if (sim_edit_display_init() == YES)
                my_key_handler = sim_edit_key_handler;
            break;

        case KEY_f2:
            sim_stop();
            nps_data_flush();
            clrscr();
            if (sim_init(0, NULL) == NO)  terminate = YES;
            main_display_init();
            break;

        case KEY_f3:
            nps_data_flush();
            break;

        case KEY_f6:
            if (nps_display_init() == YES)
                my_key_handler = nps_key_handler;
            return YES;
    }
    return NO;
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
```

```
typedef struct { int col, row; char *text; } Label;

static void main_display_init()
{
   static Label labels[] =
   {
        {16,6, "Simulator Main Menu"},
        { 6, 8, "ALT-S  Start/Stop the sim (toggle)"},
        { 6,10, "F1   Sim settings"},
        { 6,11, "F2   Restart Sim"},
        { 6,12, "F3   Clear prev Sim"},
        { 6,13, "F4"},
        { 6,14, "F5"},
        { 6,15, "F6   NPS Plots"},
        { 6,17, "ESC  Exit"},
        { 2,20, "Cycles: "},
        { 0,0, NULL }
   };

   Label *label;

   /*-------------------------------------------------------------------------
    */
   clrscr();

   for (label = labels;  label->text != NULL;  label++)
   {
        gotoxy(label->col, label->row);
        cputs(label->text);
   }
   main_display_status();
}

static void main_display_status()
{
   if (my_key_handler != main_key_handler)  return;
   gotoxy(2, 19);
   cputs((suspended ? "(halted) " : "(running)"));
}
```

```
/*=============================================================
============================*
 *-------------------------------------------------------------*/
#ifndef _SIM_
#define _SIM_

#include <orbmech1.h>
#include <vector.h>

/*=============================================================
============================*
 *  This structure is passed from the simulator code to the main routine.  This
 * is also the structure used in disk read/writes.
 *-------------------------------------------------------------*/
typedef struct
{
   State_vector orb_gps_m50, orb_gps_ecef, spas_gps, spas_sv, orb_ins, orb_targ;
   Quaternion orb_quat;
} Sim_sample;

/*-------------------------------------------------------------*/
extern int
   sim_init(int argc, char **argv),
   sim_edit_display_init(void),
   sim_edit_key_handler(int),
   sim_key_handler(int),
   sim_next_sample(Sim_sample *sample);

extern void
   sim_stop(void);

/*-------------------------------------------------------------*/
extern int
   gen_init(void);

extern void
   gen_save_sample(Sim_sample *sample),
   gen_stop(void);

#endif _SIM_
```

```
/*=============================================================
==========================
 *-------------------------------------------------------------*/
#include <stdio.h>
#include <alloc.h>

#include <edit.h>
#include "sim.h"

/*=============================================================
==========================
 *-------------------------------------------------------------*/
#define MAX_SAMPLES (0xffff / sizeof(Sim_sample) - 1)

typedef struct _mem_buffer
{
   struct _mem_buffer *next;
   int count;
   Sim_sample samples[MAX_SAMPLES];
} Mem_buffer;

static Mem_buffer *buffer_first, *buffer_current;
static Sim_sample *sample_next;
static int sample_cnt;

/*=============================================================
==========================
 *-------------------------------------------------------------*/
int sim_init()
{
   char *filename;
   FILE *infile;

   int samples;

   Mem_buffer *buffer, *buffer_prev;

   /*-------------------------------------------------------------
    */
   filename = edit_text("Enter input filename:", "sim.dat");

   if ((infile=fopen(filename, "rb")) == NULL)
   {
        printf("Can't open %s\n", filename);
        return NO;
   }

   /*-------------------------------------------------------------
    */
   fseek(infile, 0L, SEEK_END);
```

```
    samples = (int)(ftell(infile) / sizeof(Sim_sample));
    samples = edit_integer("How many samples should I load:", samples);

    if (samples <= 0)
    {
        fclose(infile);
        return NO;
    }
    fseek(infile, 0L, SEEK_SET);

    /*-----------------------------------------------------------------
     */
    for (buffer_prev = NULL;
         samples > 0;
         samples -= MAX_SAMPLES, buffer_prev = buffer)
    {
        if ((buffer = malloc(sizeof(Mem_buffer))) == NULL)
        {
            printf("Ran out of memory, %d samples remaining\n", samples);
            fclose(infile);
            return NO;
        }

        if (buffer_first == NULL)  buffer_first = buffer;
        else                       buffer_prev->next = buffer;

        buffer->next = NULL;
        buffer->count = fread(buffer->samples, sizeof(Sim_sample), MAX_SAMPLES, infile);

        if (buffer->count <= 0)  samples = 0;
    }

    /*-----------------------------------------------------------------
     */
    buffer_current = buffer_first;
    sample_cnt = buffer_current->count;
    sample_next = buffer_current->samples;

    fclose(infile);
    return YES;
}

/*=================================================================
=======================
 *-----------------------------------------------------------------*/
void sim_stop() { }

/*=================================================================
=======================
 *-----------------------------------------------------------------*/
```

```
int sim_next_sample(Sim_sample *sample)
{
   /*-----------------------------------------------------------------
    */
   if (buffer_current == NULL) return NO;

   *sample = *sample_next++;

   if (--sample_cnt <= 0)
   {
        buffer_current = buffer_current->next;
        if (buffer_current != NULL)
        {
           sample_cnt = buffer_current->count;
           sample_next = buffer_current->samples;
        }
   }

   return YES;
}

/*=================================================================
=======================
 *-----------------------------------------------------------------*/
int sim_key_handler(int key) { return YES; }

int sim_edit_display_init() { return NO; }
int sim_edit_key_handler(int key) { return YES; }
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


#include <edit_rt.h>
#include <keys.h>
#include <times.h>
#include "sim.h"
#include "../nps/nps_m50.h"

/*=============================================================
============================
 *-------------------------------------------------------------*/
static State_vector tans, spas, orb_ins, noisy_spas, noisy_tans, hold_tans;
static Quaternion orb_quat;

static double tans_delta_t, tans_step;
static double spas_delta_t, spas_step;
static double orb_ins_delta_t, orb_ins_step;
static Vector r_variance, v_variance;

static Perturbations far tans_P, far orb_P, far spas_P;

static double impulse = 2.0*0.00003048, time_to_rndv, dv2_time;
static Vector delta_v2;
static int rndv_in_progress, spas_noise, tans_noise, tans_running = YES;

static int sim_delay_flag = 1, sim_delay_ms = 0;

/*=============================================================
============================
 *-------------------------------------------------------------*/
static void
   thrust(State_vector *, State_vector *, Vector *, Vector *, Perturbations *),
   rdvz_delta_v(State_vector *, State_vector *, double, Vector *, Vector *, Perturbations *),
   getquat(void);

static int
   testquat(Vector *, Vector *, Vector *);

/*=============================================================
============================
#define julian_date_2000 2451545.0
 *-------------------------------------------------------------*/
int sim_init(int argc, char **argv)
{
```

```
/*-----------------------------------------------------------------
 */
tans.time = ((((210 * 24) + 2) * 60.0) + 15) * 60.0 + 0.0;

tans.pos.x = 6600.0;  tans.pos.y = 0.0;  tans.pos.z = 0.0;

tans.vel.x = 0.0;
tans.vel.y = sqrt(3.9860064e+5/vector_magnitude(&tans.pos))+0.0;
tans.vel.z = 0.0;

tans_delta_t = 10.0;  tans_step = 10;              /* step size (sec) */

time_to_rndv = tans_delta_t*(double)((int)(4020.0/tans_delta_t));

/*-----------------------------------------------------------------
 */
spas.time = tans.time;

spas.pos.x = 6600.0;  spas.pos.y = 0.0;  spas.pos.z = 0.0;

spas.vel.x = 0.0;
spas.vel.y = sqrt(3.9860064e+5/vector_magnitude(&spas.pos));
spas.vel.z = 0.0;

spas_delta_t = 45.0;  spas_step = 10;

/*-----------------------------------------------------------------
 */
orb_ins.time = tans.time;
orb_ins.pos.x = 6600.0;  orb_ins.pos.y = 0.0;  orb_ins.pos.z = 0.0;

orb_ins.vel.x = 0.0;
orb_ins.vel.y = sqrt(3.9860064e+5/vector_magnitude(&tans.pos))+0.0;
orb_ins.vel.z = 0.0;

orb_ins_delta_t = 10.0;  orb_ins_step = 10;        /* step size (sec) */

/*-----------------------------------------------------------------
 */
tans_P.Drag.mode = ON;
orb_P.Drag.mode  = OFF;
spas_P.Drag.mode = ON;

tans_P.Drag.solar_flux =
orb_P.Drag.solar_flux  =
spas_P.Drag.solar_flux = 185.9593403793603;

tans_P.Drag.mean_solar_flux =
orb_P.Drag.mean_solar_flux  =
spas_P.Drag.mean_solar_flux = 178.6447698684861;
```

```
    tans_P.Drag.geomagnetic_index_ap =
    orb_P.Drag.geomagnetic_index_ap  =
    spas_P.Drag.geomagnetic_index_ap = 19.87811961456716;

    tans_P.Drag.ballistic_number = 64.0;
    orb_P.Drag.ballistic_number  = 64.0;
    spas_P.Drag.ballistic_number = 64.0;

    tans_P.Gravity.number_zonal = 6;
    orb_P.Gravity.number_zonal  = 4;
    spas_P.Gravity.number_zonal = 6;

    tans_P.Gravity.number_tesseral = 6;
    orb_P.Gravity.number_tesseral  = 4;
    spas_P.Gravity.number_tesseral = 6;

    instantiate_gotpot(1, &tans_P);
    instantiate_gotpot(1, &orb_P);
    instantiate_gotpot(1, &spas_P);

    if (bgprop_init() == NO)  return NO;

    /*---------------------------------------------------------------
     */
    getquat();

    return YES;
}

/*==========================================================
==========================
 *-------------------------------------------------------------*/
void sim_stop()
{
}

/*==========================================================
==========================
 *-------------------------------------------------------------*/
int sim_next_sample(Sim_sample *sample)
{
    State_vector temp_wgs84, temp_j2000;

    /*---------------------------------------------------------------
     */
    if (sim_delay_ms)
    {
        if (sim_delay_flag == 0)  return -1;
        sim_delay_flag = 0;
        rtc_timer_set(sim_delay_ms, &sim_delay_flag);
```

```
}

/*-------------------------------------------------------------------
 */
if (rndv_in_progress  &&  tans.time >= dv2_time)
{
     tans.vel.x -= delta_v2.x;
     tans.vel.y -= delta_v2.y;
     tans.vel.z -= delta_v2.z;
     orb_ins.vel.x -= delta_v2.x;
     orb_ins.vel.y -= delta_v2.y;
     orb_ins.vel.z -= delta_v2.z;
     rndv_in_progress = NO;
}

/*-------------------------------------------------------------------
 *   Note that the tans.pos and tans.vel vectors are used for input AND for
 * output, this is ok with the current version of bgprop().
 */
bgprop(tans.time, &tans.pos, &tans.vel, &tans_P,tans_delta_t,tans_step, &tans.pos,&tans.vel);
tans.time += tans_delta_t;

/*-------------------------------------------------------------------
 */
if (spas.time + spas_delta_t <= tans.time + tans_delta_t)
{
     bgprop(spas.time, &spas.pos,&spas.vel, &spas_P,spas_delta_t,spas_step, &spas.pos,&spas.vel);
     spas.time += spas_delta_t;
}

/*-------------------------------------------------------------------
 */
bgprop(orb_ins.time, &orb_ins.pos, &orb_ins.vel,
  &orb_P,orb_ins_delta_t,orb_ins_step, &orb_ins.pos,&orb_ins.vel);
orb_ins.time += orb_ins_delta_t;

/*-------------------------------------------------------------------
 */
if (tans_noise)
{
     r_variance.x = pow((double)(random(1000)-500)/500.0, 3.0) * .2;
     r_variance.y = pow((double)(random(1000)-500)/500.0, 3.0) * .2;
     r_variance.z = pow((double)(random(1000)-500)/500.0, 3.0) * .2;
     v_variance.x = pow((double)(random(1000)-500)/500.0, 3.0) * .001;
     v_variance.y = pow((double)(random(1000)-500)/500.0, 3.0) * .001;
     v_variance.z = pow((double)(random(1000)-500)/500.0, 3.0) * .001;

     noisy_tans.pos.x = r_variance.x + tans.pos.x;
     noisy_tans.pos.y = r_variance.y + tans.pos.y;
     noisy_tans.pos.z = r_variance.z + tans.pos.z;
```

```
        noisy_tans.vel.x = v_variance.x + tans.vel.x;
        noisy_tans.vel.y = v_variance.y + tans.vel.y;
        noisy_tans.vel.z = v_variance.z + tans.vel.z;
    }
    else
    {
        noisy_tans.pos = tans.pos;
        noisy_tans.vel = tans.vel;
    }

    noisy_tans.time = tans.time;

    if (tans_running == YES)  hold_tans = noisy_tans;

    /*-----------------------------------------------------------------------
     */
    if (spas_noise)
    {
        r_variance.x = pow((double)(random(1000)-500)/500.0, 3.0) * .2;
        r_variance.y = pow((double)(random(1000)-500)/500.0, 3.0) * .2;
        r_variance.z = pow((double)(random(1000)-500)/500.0, 3.0) * .2;
        v_variance.x = pow((double)(random(1000)-500)/500.0, 3.0) * .001;
        v_variance.y = pow((double)(random(1000)-500)/500.0, 3.0) * .001;
        v_variance.z = pow((double)(random(1000)-500)/500.0, 3.0) * .001;

        noisy_spas.pos.x = r_variance.x + spas.pos.x;
        noisy_spas.pos.y = r_variance.y + spas.pos.y;
        noisy_spas.pos.z = r_variance.z + spas.pos.z;
        noisy_spas.vel.x = v_variance.x + spas.vel.x;
        noisy_spas.vel.y = v_variance.y + spas.vel.y;
        noisy_spas.vel.z = v_variance.z + spas.vel.z;
    }
    else
    {
        noisy_spas.pos = spas.pos;
        noisy_spas.vel = spas.vel;
    }

    noisy_spas.time = spas.time;

    /*-----------------------------------------------------------------------
     */
    getquat();

    sample->orb_targ = spas;
    sample->orb_ins  = orb_ins;
    sample->orb_quat = orb_quat;

#   ifdef NPS_GPS
    temp_wgs84 = noisy_spas;
```

```
   nps_m50_to_wgs84(&temp_wgs84, NPS_M50_computed, NO);
   sample->spas_gps = temp_wgs84;

   temp_j2000 = noisy_spas;
   nps_m50_to_j2000(&temp_j2000);
   sample->spas_sv = temp_j2000;

   sample->orb_gps_m50 = hold_tans;
   temp_wgs84 = hold_tans;
   nps_m50_to_wgs84(&temp_wgs84, NPS_M50_computed, YES);
   sample->orb_gps_ecef = temp_wgs84;
#  endif NPS_GPS

   return YES;
}

/*==========================================================================
==========================
 *--------------------------------------------------------------------------*/
int sim_key_handler(int key)
{
   // static char work[32];

   Vector deltav_lvlh, deltav_M50, delta_v1;
   int new_rndv_flag = NO;

   /*-----------------------------------------------------------------------
    */
   switch (key)
   {
        case KEY_alt_f1:
           orb_ins = tans;                    //orb_ins update
           break;

        case KEY_alt_f2:                      //TANS running toggle
           tans_running = (tans_running ? NO : YES);
           break;

        case KEY_alt_f3:                      //TANS noise toggle
           tans_noise = (tans_noise ? NO : YES);
           break;

        case KEY_alt_f4:                      //SPAS noise toggle
           spas_noise = (spas_noise ? NO : YES);
           break;

        case KEY_alt_f5:                      //positive 'v'
           deltav_lvlh.x = impulse; deltav_lvlh.y = 0.0; deltav_lvlh.z = 0.0;
           thrust(&spas, &tans, &deltav_lvlh, &deltav_M50, &spas_P);
           tans.vel.x += deltav_M50.x;
```

```
    tans.vel.y += deltav_M50.y;
    tans.vel.z += deltav_M50.z;
    orb_ins.vel.x += deltav_M50.x;
    orb_ins.vel.y += deltav_M50.y;
    orb_ins.vel.z += deltav_M50.z;
    break;

case KEY_alt_f6:                          //negative 'v'
    deltav_lvlh.x = -impulse; deltav_lvlh.y = 0.0; deltav_lvlh.z = 0.0;
    thrust(&spas, &tans, &deltav_lvlh, &deltav_M50, &spas_P);
    tans.vel.x += deltav_M50.x;
    tans.vel.y += deltav_M50.y;
    tans.vel.z += deltav_M50.z;
    orb_ins.vel.x += deltav_M50.x;
    orb_ins.vel.y += deltav_M50.y;
    orb_ins.vel.z += deltav_M50.z;
    break;

case KEY_alt_f7:                          //positive 'r'
    deltav_lvlh.x = 0.0; deltav_lvlh.y = 0.0; deltav_lvlh.z = impulse;
    thrust(&spas, &tans, &deltav_lvlh, &deltav_M50, &spas_P);
    tans.vel.x += deltav_M50.x;
    tans.vel.y += deltav_M50.y;
    tans.vel.z += deltav_M50.z;
    orb_ins.vel.x += deltav_M50.x;
    orb_ins.vel.y += deltav_M50.y;
    orb_ins.vel.z += deltav_M50.z;
    break;

case KEY_alt_f8:                          //negative 'r'
    deltav_lvlh.x = 0.0; deltav_lvlh.y = 0.0; deltav_lvlh.z = -impulse;
    thrust(&spas, &tans, &deltav_lvlh, &deltav_M50, &spas_P);
    tans.vel.x += deltav_M50.x;
    tans.vel.y += deltav_M50.y;
    tans.vel.z += deltav_M50.z;
    orb_ins.vel.x += deltav_M50.x;
    orb_ins.vel.y += deltav_M50.y;
    orb_ins.vel.z += deltav_M50.z;
    break;

case KEY_alt_f9:                          //increase 'delta v'
    impulse += 0.00003048;
    // setcolor(0);
    // outtextxy(80,80, work);
    // sprintf(work, "%6.2f", impulse/0.0003048);
    // setcolor(7);
    // outtextxy(80,80, work);
    break;

case KEY_alt_f10:                         //decrease 'delta v'
```

```
            impulse -= 0.00003048;
            // setcolor(0);
            // outtextxy(80,80, work);
            // sprintf(work,"%6.2f", impulse/0.0003048);
            // setcolor(7);
            // outtextxy(80,80, work);
            break;

         case KEY_alt_r:                          //secret stuff
            rdvz_delta_v(&spas, &tans, time_to_rndv, &delta_v1, &delta_v2, &spas_P);
            tans.vel.x += delta_v1.x;
            tans.vel.y += delta_v1.y;
            tans.vel.z += delta_v1.z;
            orb_ins.vel.x += delta_v1.x;
            orb_ins.vel.y += delta_v1.y;
            orb_ins.vel.z += delta_v1.z;
            dv2_time = tans.time + time_to_rndv;
            new_rndv_flag = YES;
            break;

         default:
            new_rndv_flag = rndv_in_progress;
            break;
      }

      rndv_in_progress = new_rndv_flag;
      return YES;
   }

   /*=============================================================
   ========================
    *  These are helper routines for the edit screens below.  These routines help
    * by copying one just-set variable to others that must be the same value.
    *-------------------------------------------------------------------*/
   static int drag_fields_put(char *work, void *data)
   {
      edit_field_put_dbl(work, data);

      if (data == &orb_P.Drag.solar_flux)
           tans_P.Drag.solar_flux =
           spas_P.Drag.solar_flux =  orb_P.Drag.solar_flux;
      else if (data == &orb_P.Drag.mean_solar_flux)
           tans_P.Drag.mean_solar_flux =
           spas_P.Drag.mean_solar_flux =  orb_P.Drag.mean_solar_flux;
      else
           tans_P.Drag.geomagnetic_index_ap =
           spas_P.Drag.geomagnetic_index_ap = orb_P.Drag.geomagnetic_index_ap;

      return YES;
   }
```

```
static int init_conds_put(char *work, void *data)
{
   edit_field_put_dbl(work, data);
   if (data == &tans.pos.x)
        spas.pos.x = orb_ins.pos.x = tans.pos.x;
   else if (data == &tans.pos.y)
        spas.pos.y = orb_ins.pos.y = tans.pos.y;
   else if (data == &tans.pos.z)
        spas.pos.z = orb_ins.pos.z = tans.pos.z;
   else if (data == &tans.vel.x)
        spas.vel.x = orb_ins.vel.x = tans.vel.x;
   else if (data == &tans.vel.y)
        spas.vel.y = orb_ins.vel.y = tans.vel.y;
   else
        spas.vel.z = orb_ins.vel.z = tans.vel.z;
   return YES;
}

static int init_time_put(char *work, void *data)
{
   edit_field_put_dhms1(work, data);
   spas.time = orb_ins.time = tans.time;
   return YES;
}

static int orb_ballistic_put(char *work, void *data)
{
   edit_field_put_dbl(work, data);
   tans_P.Drag.ballistic_number = orb_P.Drag.ballistic_number;
   return YES;
}

/*==========================================================
==========================
 *--123456789 123456789 123456789 123456789 123456789 123456789
 * 1              SIM Propagation Perturbations
 * 2
 * 3  Orb   TANS  SPAS                    Sim Keys:
 * 4   Y     Y     Y   Drag mode on              Alt-F1   Orb-INS = TANS
 * 5     XX        XX   Ballistic number   Alt-F2   TANS on/off
 * 6   X     X     X   Zonal Harmonic      Alt-F3   TANS noise on/off
 * 7   X     X     X   Tesseral Harmonic           Alt-F4   SPAS noise on/off
 * 8         Y     Y   Noise mode on       Alt-F5   Thrust +V
 * 9                                              Alt-F6   Thrust -V
 *10   Drag Info                                   Alt-F7   Thrust +R
 *11    Solar Flux: X                             Alt-F8   Thrust -R
 *12    Mean Solar Flux: X                        Alt-F9   Inc delta-v
 *13    Geomagnetic Activity Index: X             Alt-F10  Dec delta-v
 *14                                              Alt-R    Rendezvous
 *15
```

```
*16                SIM Initial Conditons
*17   Position (km)              Velocity (km/s)
*18     x: X                     vx: X
*19     y: X                     vy: X
*20     z: X                     vz: X
*21
*22   Time (ddd/hh:mm:ss): DDD/HH:MM:SS
*23
*24   Realtime delay (ms): X
*-------------------------------------------------------------------------*/
static Edit_label labels1[] =
{
   { 1,20, "SIM Propagation Perturbations"},
   { 3, 3, "Orb    TANS   SPAS"},
   { 4,24, "Drag mode on"},
   { 5,24, "Ballistic number"},
   { 6,24, "Zonal harmonic"},
   { 7,24, "Tesseral harmonic"},
   { 8,24, "Noise mode on"},
   {10, 4, "Drag Information"},
   {11, 6, "Solar Flux:"},
   {12, 6, "Mean Solar Flux:"},
   {13, 6, "Geomagnetic Activity Index:"},
   { 3,47, "Sim Keys:"},
   { 4,49, "Alt-F1    Orb-INS = TANS"},
   { 5,49, "Alt-F2    TANS on/off"},
   { 6,49, "Alt-F3    TANS noise on/off"},
   { 7,49, "Alt-F4    SPAS noise on/off"},
   { 8,49, "Alt-F5    Thrust +V"},
   { 9,49, "Alt-F6    Thrust -V"},
   {10,49, "Alt-F7    Thrust +R"},
   {11,49, "Alt-F8    Thrust -R"},
   {12,49, "Alt-F9    Inc delta-v"},
   {13,49, "Alt-F10   Dec delta-v"},
   {14,49, "Alt-R     Rendezvous"},
   {16,20, "SIM Initial Conditions"},
   {17, 4, "Position (km)                Velocity (km/s)"},
   {18, 6, "x:"},
   {18,37, "vx:"},
   {19, 6, "y:"},
   {19,37, "vy:"},
   {20, 6, "z:"},
   {20,37, "vz:"},
   {22, 4, "Time (ddd/hh:mm:ss):"},
   {24, 4, "Realtime delay (ms):"}
};
#define LABEL1_COUNT (sizeof(labels1)/sizeof(Edit_label))

static Edit_field fields1[] =
{
```

```
   { 4, 5, edit_field_get_yesno, edit_field_put_yesno, edit_field_key_yesno, &orb_P.Drag.mode },
   { 4,12, edit_field_get_yesno, edit_field_put_yesno, edit_field_key_yesno, &tans_P.Drag.mode },
   { 4,19, edit_field_get_yesno, edit_field_put_yesno, edit_field_key_yesno, &spas_P.Drag.mode },
   { 5, 8, edit_field_get_dbl,   orb_ballistic_put,    edit_field_key_dbl,   &orb_P.Drag.ballistic_number },
   { 5,18, edit_field_get_dbl,   edit_field_put_dbl,   edit_field_key_dbl,   &spas_P.Drag.ballistic_number },
   { 6, 5, edit_field_get_int,   edit_field_put_int,   edit_field_key_int,   &orb_P.Gravity.number_zonal },
   { 6,12, edit_field_get_int,   edit_field_put_int,   edit_field_key_int,   &tans_P.Gravity.number_zonal },
   { 6,19, edit_field_get_int,   edit_field_put_int,   edit_field_key_int,   &spas_P.Gravity.number_zonal },
   { 7, 5, edit_field_get_int,   edit_field_put_int,   edit_field_key_int,   &orb_P.Gravity.number_tesseral },
   { 7,12, edit_field_get_int,   edit_field_put_int,   edit_field_key_int,   &tans_P.Gravity.number_tesseral },
   { 7,19, edit_field_get_int,   edit_field_put_int,   edit_field_key_int,   &spas_P.Gravity.number_tesseral },
   { 8,12, edit_field_get_yesno, edit_field_put_yesno, edit_field_key_yesno, &tans_noise },
   { 8,19, edit_field_get_yesno, edit_field_put_yesno, edit_field_key_yesno, &spas_noise },
   {11,18, edit_field_get_dbl,   drag_fields_put,      edit_field_key_dbl,   &orb_P.Drag.solar_flux },
   {12,23, edit_field_get_dbl,   drag_fields_put,      edit_field_key_dbl,   &orb_P.Drag.mean_solar_flux },
   {13,34, edit_field_get_dbl,   drag_fields_put,      edit_field_key_dbl,   &orb_P.Drag.geomagnetic_index_ap
},
   {18, 9, edit_field_get_dbl,   init_conds_put,       edit_field_key_dbl,   &tans.pos.x },
   {18,41, edit_field_get_dbl,   init_conds_put,       edit_field_key_dbl,   &tans.vel.x },
   {19, 9, edit_field_get_dbl,   init_conds_put,       edit_field_key_dbl,   &tans.pos.y },
   {19,41, edit_field_get_dbl,   init_conds_put,       edit_field_key_dbl,   &tans.vel.y },
   {20, 9, edit_field_get_dbl,   init_conds_put,       edit_field_key_dbl,   &tans.pos.z },
   {20,41, edit_field_get_dbl,   init_conds_put,       edit_field_key_dbl,   &tans.vel.z },
   {22,25, edit_field_get_dhms1, init_time_put,        edit_field_key_dhms,  &tans.time },
   {24,25, edit_field_get_int,   edit_field_put_int,   edit_field_key_int,   &sim_delay_ms }
};
#define FIELD1_COUNT (sizeof(fields1)/sizeof(Edit_field))

Edit_screen screen1 =
{
   LABEL1_COUNT, labels1,
   FIELD1_COUNT, fields1,
   NULL, NO, NULL, NULL
};

/*=============================================================
=========================
 *-------------------------------------------------------------*/
static Edit_screen *screens[] =
{
   &screen1
};
#define MAX_SCREENS (sizeof(screens)/sizeof(Edit_screen *))

static int current_screen;

/*=============================================================
=========================
 *-------------------------------------------------------------*/
int sim_edit_display_init()
```

```
{
   edit_screen_init(screens[current_screen]);
   return YES;
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
int sim_edit_key_handler(int key)
{
   int rc = edit_screen_key_handler(screens[current_screen], key);

   if (rc > 0)  return rc;

   /*-----------------------------------------------------------
    */
   switch (key)
   {
        case KEY_pgup:
           if (--current_screen < 0)  current_screen = MAX_SCREENS - 1;
           break;

        case KEY_pgdn:
           if (++current_screen >= MAX_SCREENS)  current_screen = 0;
           break;

        default:  return NO;
   }

   edit_screen_init(screens[current_screen]);
   return YES;
}

/*=============================================================
=========================
 * This function is designed to turn thrust commands in an LVC (local LVLH)
 * coordinate frame into M50 commanded delta v's to be applied before the
 * next step in the driver.
 */
void thrust(State_vector *target, State_vector *chaser,
            Vector *deltav_lvlh, Vector *deltav_M50, Perturbations *P)
{
   double magpos, magxxv, magvbar, dt, gcl[3][3];
   double step = 5.0;
   Vector mhhat, vb, pos_o, vel_o, xtxvt;

   /*-----------------------------------------------------------
    */
   if (chaser->time >= target->time)
   {
```

```
        pos_o = chaser->pos;
        vel_o = chaser->vel;
    }
    else
    {
        dt = target->time - chaser->time;
        bgprop(chaser->time, &chaser->pos, &chaser->vel, P, dt, step, &pos_o, &vel_o);
    }

    /*-------------------------------------------------------------
     */
    magpos = vector_magnitude(&pos_o);

    gcl[0][2] = -pos_o.x / magpos;
    gcl[1][2] = -pos_o.y / magpos;
    gcl[2][2] = -pos_o.z / magpos;

    /*-------------------------------------------------------------
     */
    vector_cross(&pos_o, &vel_o, &xtxvt);
    magxxv = vector_magnitude(&xtxvt);

    gcl[0][1] =  mhhat.x =  -(xtxvt.x / magxxv);
    gcl[1][1] =  mhhat.y =  -(xtxvt.y / magxxv);
    gcl[2][1] =  mhhat.z =  -(xtxvt.z / magxxv);

    vector_cross(&pos_o, &mhhat, &vb);

    magvbar = vector_magnitude(&vb);

    gcl[0][0] = vb.x / magvbar;
    gcl[1][0] = vb.y / magvbar;
    gcl[2][0] = vb.z / magvbar;

    vector_transform(deltav_lvlh,deltav_M50,gcl);
}

/*==========================================================
==========================
* This is the "bad" function.  The inputs are the target state vector, the
* chaser state vector, and the time alloted for the rendezvous (ttr
* this must be in even increments of the propogation to allow for input of
* the second delta v).  The output is the delta v's (in M50) to be input at
* the next step, and at the end of ttr for the chaser to effect a rendezvous
* with the target.
*/

void rdvz_delta_v(State_vector *target, State_vector *chaser,
                  double ttr, Vector *dv1_M50, Vector *dv2_M50,
                  Perturbations *P)
```

```
{
   static double mu = 398600.64;

   double step = 15.0;

   double magpos_t, magvel_t, magxxv, magvbar, lcg[3][3], gcl[3][3];
   double dt, delta;
   Vector xtxvt, hhat, vb, rdot, ro, dv1_M50_1;
   Vector pos_dif, dv1_lvlh, dv2_lvlh, omega, wxr_term;
   Vector pos_t, vel_t, pos_o, vel_o, pos_tnew, vel_tnew;
   double T,w;
   int i,j;

/*
* first match the time tags by propogating target to chaser regardless of
* which is more recent.  This is artificial due to the restriction of
* applying delta v's only at integrator (sim) time steps for chaser.
*/
        dt = chaser->time - target->time;
        if(fabs(dt) > 1.0e-8)
         {
           bgprop(target->time, &target->pos, &target->vel, P, dt,
                  step, &pos_t, &vel_t);
         }
        else
         {
           pos_t = target->pos;
           vel_t = target->vel;
         }

        pos_o = chaser->pos;
        vel_o = chaser->vel;


   /*-----------------------------------------------------------------
    */

   magpos_t = vector_magnitude(&pos_t);
   magvel_t = vector_magnitude(&vel_t);
   T = 2*PI*sqrt(mu*mu/(pow(2*mu/magpos_t - magvel_t*magvel_t,3)));
   w = 2*PI/T;

   lcg[1][0] = pos_t.x / magpos_t;
   lcg[1][1] = pos_t.y / magpos_t;
   lcg[1][2] = pos_t.z / magpos_t;

   /*-----------------------------------------------------------------
    */
   vector_cross(&pos_t,&vel_t, &xtxvt);
   magxxv = vector_magnitude(&xtxvt);
```

```
lcg[2][0] =  hhat.x = xtxvt.x / magxxv;
lcg[2][1] =  hhat.y = xtxvt.y / magxxv;
lcg[2][2] =  hhat.z = xtxvt.z / magxxv;

vector_cross(&pos_t, &hhat, &vb);

magvbar = vector_magnitude(&vb);

lcg[0][0] = vb.x / magvbar;  pos_dif.x = pos_o.x - pos_t.x;
lcg[0][1] = vb.y / magvbar;  pos_dif.y = pos_o.y - pos_t.y;
lcg[0][2] = vb.z / magvbar;  pos_dif.z = pos_o.z - pos_t.z;


vector_transform(&pos_dif, &ro, lcg);

/*-----------------------------------------------------------------
 */


rdot.x = ro.x*sin(w*ttr) + ro.y*(6*w*ttr*sin(w*ttr)-14*(1 - cos(w*ttr)));
delta = 3*w*ttr*sin(w*ttr) - 8*(1 - cos(w*ttr));
rdot.x = rdot.x/delta;
rdot.y = ro.x+(4*rdot.x-6*ro.y)*sin(w*ttr)+(6*ro.y-3*rdot.x)*w*ttr;
rdot.y = rdot.y/(-2)/(1-cos(w*ttr));
rdot.x = rdot.x*w;
rdot.y = rdot.y*w;

rdot.z = -ro.z*w/tan(w*ttr);

omega.x = 0.0;
omega.y = 0.0;
omega.z = w;

vector_cross(&omega, &ro, &wxr_term);


dv1_lvlh.x = rdot.x + wxr_term.x;
dv1_lvlh.y = rdot.y + wxr_term.y;
dv1_lvlh.z = rdot.z + wxr_term.z;

for(j=0; j<3; j++)
{
     for(i=0; i<3; i++)
      {
        gcl[i][j] = lcg[j][i];
      }
}

vector_transform(&dv1_lvlh, &dv1_M50_1, gcl);
```

```
    dv1_M50->x = vel_t.x + dv1_M50_1.x - vel_o.x;
    dv1_M50->y = vel_t.y + dv1_M50_1.y - vel_o.y;
    dv1_M50->z = vel_t.z + dv1_M50_1.z - vel_o.z;

    /*-------------------------------------------------------------
     */

    dv2_lvlh.x = (2*rdot.y*sin(w*ttr)+(4*rdot.x-6*w*ro.y)*cos(w*ttr)+
                    6*w*ro.y-3*rdot.x);
    dv2_lvlh.y = ((3*w*ro.y - 2*rdot.x)*sin(w*ttr) + rdot.y*cos(w*ttr));
    dv2_lvlh.z = rdot.z*cos(w*ttr) - ro.z*w*sin(w*ttr);

/*
* next we have to propogate the target again to a time ttr plus where we
* we are right now (a second propagation)
*/

    /*-------------------------------------------------------------------
     */
    bgprop(chaser->time, &pos_t, &vel_t, P, ttr, step, &pos_tnew, &vel_tnew);

        magpos_t = vector_magnitude(&pos_tnew);

        gcl[0][1] = pos_tnew.x / magpos_t;
        gcl[1][1] = pos_tnew.y / magpos_t;
        gcl[2][1] = pos_tnew.z / magpos_t;

        /*-----------------------------------------------------------------
         */
        vector_cross(&pos_tnew, &vel_tnew, &xtxvt);
        magxxv = vector_magnitude(&xtxvt);

        gcl[0][2] =  hhat.x = xtxvt.x / magxxv;
        gcl[1][2] =  hhat.y = xtxvt.y / magxxv;
        gcl[2][2] =  hhat.z = xtxvt.z / magxxv;

        vector_cross(&pos_tnew, &hhat, &vb);

        magvbar = vector_magnitude(&vb);

        gcl[0][0] = vb.x / magvbar;
        gcl[1][0] = vb.y / magvbar;
        gcl[2][0] = vb.z / magvbar;

        vector_transform(&dv2_lvlh,dv2_M50,gcl);

}

/*=============================================================
============================
```

```
 * getquat(*quat, *orbiter)...produces a quaternian for the attitude of the
 * orbiter "Qbi" based on its position and velocity.  This is by no means
 * an implication that this propogates the orbiter attitude.  This is however
 * a way of creating a time varying quaternian based on a desired attitude.
 * this function is currently designed to produce a gravity gradient position
 * with the payload bay facing aft, as if preparing to rendezvous.
 *-----------------------------------------------------------------------*/
void getquat(void)
{
   Vector pos_o, vel_o, b1, b2, b3, rxv, b1xb2;
   double A, B, C, magpos, magrxv, magb1xb2;
   int good_quat, i;

   pos_o = tans.pos;
   vel_o = tans.vel;

   /*-----------------------------------------------------------------------
    */
   magpos = vector_magnitude(&pos_o);
   b1.x = pos_o.x/magpos;
   b1.y = pos_o.y/magpos;
   b1.z = pos_o.z/magpos;

   vector_cross(&pos_o,&vel_o,&rxv);
   magrxv = vector_magnitude(&rxv);
   b2.x = -rxv.x/magrxv;
   b2.y = -rxv.y/magrxv;
   b2.z = -rxv.z/magrxv;

   vector_cross(&b1,&b2,&b1xb2);
   magb1xb2 = vector_magnitude(&b1xb2);
   b3.x = b1xb2.x/magb1xb2;
   b3.y = b1xb2.y/magb1xb2;
   b3.z = b1xb2.z/magb1xb2;

   /*-----------------------------------------------------------------------
    */
   A = b1.y + b2.x;
   B = b2.z + b3.y;
   C = b1.z + b3.x;

   if( fabs(B) >= 1.0e-8 )
   {
        orb_quat.x = sqrt(A*C/4.0/B);
        if( fabs(orb_quat.x) >= 1.0e-8 )
        {
           orb_quat.y = A/4.0/orb_quat.x;
           orb_quat.z = C/4.0/orb_quat.x;
           if( fabs(orb_quat.y) >= 1.0e-8 )
           {
```

```
                orb_quat.s = (b1.z - 2*orb_quat.x*orb_quat.z)/2/orb_quat.y;
            }
            else
            {
                orb_quat.y = orb_quat.z = 0.0;
                orb_quat.s = orb_quat.x = 0.7071;
            }
        }
        else
        {
            orb_quat.y = orb_quat.z = 0.0;
            orb_quat.s = orb_quat.x = 0.7071;
        }
    }
    else
    {
        orb_quat.y = orb_quat.z = 0.0;
        orb_quat.s = orb_quat.x =  0.7071;
    }

    /*-----------------------------------------------------------------
     */
    good_quat = NO;
    orb_quat.x = -orb_quat.x;
    i = 0;
    while (good_quat == NO)
    {
        orb_quat.x = -orb_quat.x;
        good_quat = testquat(&b1,&b2,&b3);
        i++;
        if (i > 2)
        {
            orb_quat.y = orb_quat.z =  0.0;
            orb_quat.s = orb_quat.x =  0.7071;
            good_quat = YES;
        }
    }
    orb_quat.time = tans.time;
}

/*==================================================================
====================
 * This function evaluates the valitity of the relationship between a
 * quaternion and a DCM composed of three orthonormal vectors as the rows
 *-----------------------------------------------------------------*/
int testquat(Vector *b1, Vector *b2, Vector *b3)
{
    double dif;
    int good_quat;
```

```
    dif = fabs(b1->y - 2*(orb_quat.x*orb_quat.y-orb_quat.s*orb_quat.z));
    dif = dif + fabs(b1->z - 2*(orb_quat.x*orb_quat.z+orb_quat.s*orb_quat.y));
    dif = dif + fabs(b2->x - 2*(orb_quat.x*orb_quat.y+orb_quat.s*orb_quat.z));
    dif = dif + fabs(b3->x - 2*(orb_quat.x*orb_quat.z-orb_quat.s*orb_quat.y));
    dif = dif + fabs(b3->y - 2*(orb_quat.y*orb_quat.z+orb_quat.s*orb_quat.x));
    dif = dif + fabs(b2->z - 2*(orb_quat.y*orb_quat.z-orb_quat.s*orb_quat.x));

    if( dif < 1.e-6 )
    {
        good_quat = YES;
    }
    else
    {
        good_quat = NO;
    }
    return(good_quat);
}
```

```
/*=============================================================
=========================
 *-------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include <edit.h>
#include <edit_rt.h>
#include <keys.h>
#include "sim.h"

/*=============================================================
=========================
 *-------------------------------------------------------------*/
static State_vector tans, spas, orb_ins, noisy_tans, hold_tans;
static Quaternion orb_quat;

static double tans_delta_t, tans_step;
static double spas_delta_t, spas_step;
static double orb_ins_delta_t, orb_ins_step;
static double rx_variance, vx_variance;
static double ry_variance, vy_variance;
static double rz_variance, vz_variance;

static Perturbations far tans_P, far orb_P, far spas_P;

static double impulse = 2.0*0.00003048, time_to_rndv, dv2_time;
static Vector delta_v2;
static int rndv_in_progress, noise, tans_running = YES;

/*=============================================================
=========================
 *-------------------------------------------------------------*/
static void
   thrust(State_vector *, State_vector *, Vector *, Vector *, Perturbations *),
   rdvz_delta_v(State_vector *, State_vector *, double, Vector *, Vector *, Perturbations *),
   getquat(void);

static int
   testquat(Vector *, Vector *, Vector *);

static FILE *infile2;

/*=============================================================
=========================
#define julian_date_2000 2451545.0
 *-------------------------------------------------------------*/
int sim_init()
{
```

```
char  *filename2;


filename2 = edit_text("Enter input filename:", "tans_sa.dat");
if ((infile2=fopen(filename2, "r")) == NULL)
{
     printf("Can't open %s\n", filename2);
     return NO;
}

/*-----------------------------------------------------------------
 */
tans.time = ((((20 * 24) + 2) * 60.0) + 15) * 60.0 + 0.0;

tans.pos.x = 6600.0;  tans.pos.y = 0.0;  tans.pos.z = 0.0;

tans.vel.x = 0.0;
tans.vel.y = sqrt(3.9860064e+5/vector_magnitude(&tans.pos))+0.0;
tans.vel.z = 0.0;

tans_delta_t = 10.0;  tans_step = 10;            /* step size (sec) */

time_to_rndv = tans_delta_t*(double)((int)(4020.0/tans_delta_t));

/*-----------------------------------------------------------------
 */
spas.time = ((((20 * 24) + 2) * 60.0) + 15) * 60.0 + 0.0;

spas.pos.x = 6600.0;  spas.pos.y = 0.0;  spas.pos.z = 0.0;

spas.vel.x = 0.0;
spas.vel.y = sqrt(3.9860064e+5/vector_magnitude(&spas.pos));
spas.vel.z = 0.0;

spas_delta_t = 45.0;  spas_step = 10;

/*-----------------------------------------------------------------
 */
orb_ins.time = ((((20 * 24) + 2) * 60.0) + 15) * 60.0 + 0.0;
orb_ins.pos.x = 6600.0;  orb_ins.pos.y = 0.0;  orb_ins.pos.z = 0.0;

orb_ins.vel.x = 0.0;
orb_ins.vel.y = sqrt(3.9860064e+5/vector_magnitude(&tans.pos))+0.0;
orb_ins.vel.z = 0.0;

orb_ins_delta_t = 10.0;  orb_ins_step = 10;            /* step size (sec) */

/*-----------------------------------------------------------------
 */
tans_P.Drag.mode =
```

```
    orb_P.Drag.mode  =
    spas_P.Drag.mode = ON;

    tans_P.Drag.solar_flux =
    orb_P.Drag.solar_flux  =
    spas_P.Drag.solar_flux = 185.9593403793603;

    tans_P.Drag.mean_solar_flux =
    orb_P.Drag.mean_solar_flux  =
    spas_P.Drag.mean_solar_flux = 178.6447698684861;

    tans_P.Drag.geomagnetic_index_ap =
    orb_P.Drag.geomagnetic_index_ap  =
    spas_P.Drag.geomagnetic_index_ap = 19.87811961456716;

    tans_P.Drag.ballistic_number = 64.0;
    orb_P.Drag.ballistic_number  = 64.0;
    spas_P.Drag.ballistic_number = 64.0;

    tans_P.Gravity.number_zonal =
    orb_P.Gravity.number_zonal  =
    spas_P.Gravity.number_zonal = 6;

    tans_P.Gravity.number_tesseral =
    orb_P.Gravity.number_tesseral  =
    spas_P.Gravity.number_tesseral = 6;

    instantiate_gotpot(1, &tans_P);
    instantiate_gotpot(1, &orb_P);
    instantiate_gotpot(1, &spas_P);

    if (bgprop_init() == NO)  return NO;

    /*-----------------------------------------------------------------
     */
    getquat();

    return YES;
}

/*=================================================================
=========================
 *-----------------------------------------------------------------*/
void sim_stop()
{
}

/*=================================================================
=========================
 *-----------------------------------------------------------------*/
```

```
int sim_next_sample(Sim_sample *sample)
{

   static double initial_time;
   static first_time = YES;

   int cnt;

   /*-----------------------------------------------------------------
    */
   if (rndv_in_progress && tans.time >= dv2_time)
   {
        tans.vel.x -= delta_v2.x;
        tans.vel.y -= delta_v2.y;
        tans.vel.z -= delta_v2.z;
        orb_ins.vel.x -= delta_v2.x;
        orb_ins.vel.y -= delta_v2.y;
        orb_ins.vel.z -= delta_v2.z;
        rndv_in_progress = NO;
   }

   /*-----------------------------------------------------------------
    */
   cnt = fscanf(infile2,"%lf %lf %lf %lf %lf %lf %lf", &sample->orb_gps.time,
     &sample->orb_gps.pos.x, &sample->orb_gps.pos.y, &sample->orb_gps.pos.z,
     &sample->orb_gps.vel.x, &sample->orb_gps.vel.y, &sample->orb_gps.vel.z);

   if (cnt < 7) return NO;

   if (first_time)
   {
        initial_time = sample->spas.time;
        first_time = NO;
   }

   sample->orb_gps.time += initial_time;

   /*-----------------------------------------------------------------
    */
   if (spas.time + spas_delta_t <= orb_ins.time + orb_ins_delta_t)
   {
        bgprop(spas.time, &spas.pos,&spas.vel, &spas_P,spas_delta_t,spas_step, &spas.pos,&spas.vel);
        spas.time += spas_delta_t;
   }

   /*-----------------------------------------------------------------
    */
   bgprop(orb_ins.time, &orb_ins.pos, &orb_ins.vel,
     &orb_P,orb_ins_delta_t,orb_ins_step, &orb_ins.pos,&orb_ins.vel);
   orb_ins.time += orb_ins_delta_t;
```

```
    /*-----------------------------------------------------------
     */
    if (noise)
    {
        rx_variance = pow((double)(random(1000)-500)/500.0, 3.0) * .2;
        ry_variance = pow((double)(random(1000)-500)/500.0, 3.0) * .2;
        rz_variance = pow((double)(random(1000)-500)/500.0, 3.0) * .2;
        vx_variance = pow((double)(random(1000)-500)/500.0, 3.0) * .001;
        vy_variance = pow((double)(random(1000)-500)/500.0, 3.0) * .001;
        vz_variance = pow((double)(random(1000)-500)/500.0, 3.0) * .001;
    }
    else
    {
        rx_variance = 0.0;
        ry_variance = 0.0;
        rz_variance = 0.0;
        vx_variance = 0.0;
        vy_variance = 0.0;
        vz_variance = 0.0;
    }

    noisy_tans.pos.x = rx_variance + tans.pos.x;
    noisy_tans.pos.y = ry_variance + tans.pos.y;
    noisy_tans.pos.z = rz_variance + tans.pos.z;

    noisy_tans.vel.x = vx_variance + tans.vel.x;
    noisy_tans.vel.y = vy_variance + tans.vel.y;
    noisy_tans.vel.z = vz_variance + tans.vel.z;

    noisy_tans.time = tans.time;
    getquat();

    if (tans_running == YES)  hold_tans = noisy_tans;

    /*-----------------------------------------------------------------
     */
    sample->spas_sv = spas;
    sample->spas_gps = spas;
    sample->orb_targ = spas;
    sample->orb_ins  = orb_ins;
    sample->orb_quat = orb_quat;
    sample->orb_gps = hold_tans;
    return YES;
}

/*=================================================================
==========================
 *-----------------------------------------------------------------*/
int sim_key_handler(int key)
{
```

```
// static char work[32];

Vector deltav_lvlh, deltav_M50, delta_v1;
int new_rndv_flag = NO;

/*-----------------------------------------------------------------
 */
switch (key)
{
     case KEY_alt_f1:
        orb_ins = tans;                                  //orb_ins update
        break;

     case KEY_alt_f2:                                    //noise toggle
        noise = (noise ? NO : YES);
        break;

     case KEY_alt_f3:                                    //TANS running toggle
        tans_running = (tans_running ? NO : YES);
        break;

     case KEY_alt_f5:                                    //positive 'v'
        deltav_lvlh.x = impulse; deltav_lvlh.y = 0.0; deltav_lvlh.z = 0.0;
        thrust(&spas, &tans, &deltav_lvlh, &deltav_M50, &tans_P);
        tans.vel.x += deltav_M50.x;
        tans.vel.y += deltav_M50.y;
        tans.vel.z += deltav_M50.z;
        orb_ins.vel.x += deltav_M50.x;
        orb_ins.vel.y += deltav_M50.y;
        orb_ins.vel.z += deltav_M50.z;
        break;

     case KEY_alt_f6:                                    //negative 'v'
        deltav_lvlh.x = -impulse; deltav_lvlh.y = 0.0; deltav_lvlh.z = 0.0;
        thrust(&spas, &tans, &deltav_lvlh, &deltav_M50, &tans_P);
        tans.vel.x += deltav_M50.x;
        tans.vel.y += deltav_M50.y;
        tans.vel.z += deltav_M50.z;
        orb_ins.vel.x += deltav_M50.x;
        orb_ins.vel.y += deltav_M50.y;
        orb_ins.vel.z += deltav_M50.z;
        break;

     case KEY_alt_f7:                                    //positive 'r'
        deltav_lvlh.x = 0.0; deltav_lvlh.y = 0.0; deltav_lvlh.z = impulse;
        thrust(&spas, &tans, &deltav_lvlh, &deltav_M50, &tans_P);
        tans.vel.x += deltav_M50.x;
        tans.vel.y += deltav_M50.y;
        tans.vel.z += deltav_M50.z;
        orb_ins.vel.x += deltav_M50.x;
```

```
        orb_ins.vel.y += deltav_M50.y;
        orb_ins.vel.z += deltav_M50.z;
        break;

    case KEY_alt_f8:                                    //negative 'r'
        deltav_lvlh.x = 0.0; deltav_lvlh.y = 0.0; deltav_lvlh.z = -impulse;
        thrust(&spas, &tans, &deltav_lvlh, &deltav_M50, &tans_P);
        tans.vel.x += deltav_M50.x;
        tans.vel.y += deltav_M50.y;
        tans.vel.z += deltav_M50.z;
        orb_ins.vel.x += deltav_M50.x;
        orb_ins.vel.y += deltav_M50.y;
        orb_ins.vel.z += deltav_M50.z;
        break;

    case KEY_alt_f9:                                    //increase 'delta v'
        impulse += 0.00003048;
        // setcolor(0);
        // outtextxy(80,80, work);
        // sprintf(work,"%6.2f", impulse/0.0003048);
        // setcolor(7);
        // outtextxy(80,80, work);
        break;

    case KEY_alt_f10:                                   //decrease 'delta v'
        impulse -= 0.00003048;
        // setcolor(0);
        // outtextxy(80,80, work);
        // sprintf(work,"%6.2f", impulse/0.0003048);
        // setcolor(7);
        // outtextxy(80,80, work);
        break;

    case KEY_alt_r:                                     //secret stuff
        rdvz_delta_v(&spas, &tans, time_to_rndv, &delta_v1, &delta_v2, &tans_P);
        tans.vel.x += delta_v1.x;
        tans.vel.y += delta_v1.y;
        tans.vel.z += delta_v1.z;
        orb_ins.vel.x += delta_v1.x;
        orb_ins.vel.y += delta_v1.y;
        orb_ins.vel.z += delta_v1.z;
        dv2_time = tans.time + time_to_rndv;
        new_rndv_flag = YES;
        break;

    default:
        new_rndv_flag = rndv_in_progress;
        break;
}
```

```
   rndv_in_progress = new_rndv_flag;
   return YES;
}

/*=============================================================
============================
 *  These are helper routines for the edit screens below.  These routines help
 * by copying one just-set variable to others that must be the same value.
 *-------------------------------------------------------------*/
static int drag_fields_put(char *work, void *data)
{
   edit_field_put_dbl(work, data);

   if (data == &orb_P.Drag.solar_flux)
        tans_P.Drag.solar_flux =
        spas_P.Drag.solar_flux = orb_P.Drag.solar_flux;
   else if (data == &orb_P.Drag.mean_solar_flux)
        tans_P.Drag.mean_solar_flux =
        spas_P.Drag.mean_solar_flux = orb_P.Drag.mean_solar_flux;
   else
        tans_P.Drag.geomagnetic_index_ap =
        spas_P.Drag.geomagnetic_index_ap = orb_P.Drag.geomagnetic_index_ap;

   return YES;
}

static int orb_ballistic_put(char *work, void *data)
{
   edit_field_put_dbl(work, data);
   tans_P.Drag.ballistic_number = orb_P.Drag.ballistic_number;
   return YES;
}

/*=============================================================
============================
 *--123456789 123456789 123456789 123456789 123456789 123456789
 * 1              SIM Propagation Perturbations
 * 2
 * 3  Orb   TANS  SPAS
 * 4   Y     Y     Y   Drag mode on
 * 5     XX       XX   Ballistic number
 * 6   X     X     X   Zonal Harmonic
 * 7   X     X     X   Tesseral Harmonic
 * 8         Y         Noise mode on
 * 9
 *10  Drag Info
 *11    Solar Flux: X
 *12    Mean Solar Flux: X
 *13    Geomagnetic Activity Index: X
 *-------------------------------------------------------------*/
```

```
static Edit_label labels1[] =
{
   { 1,20, "SIM Propagation Perturbations"},
   { 3, 3, "Orb   TANS  SPAS"},
   { 4,24, "Drag mode on"},
   { 5,24, "Ballistic number"},
   { 6,24, "Zonal harmonic"},
   { 7,24, "Tesseral harmonic"},
   { 8,24, "Noise mode on"},
   {10, 4, "Drag Information"},
   {11, 6, "Solar Flux:"},
   {12, 6, "Mean Solar Flux:"},
   {13, 6, "Geomagnetic Activity Index:"}

};
#define LABEL1_COUNT (sizeof(labels1)/sizeof(Edit_label))

static Edit_field fields1[] =
{
   { 4, 5, edit_field_get_yesno, edit_field_put_yesno, edit_field_key_yesno, &orb_P.Drag.mode },
   { 4,12, edit_field_get_yesno, edit_field_put_yesno, edit_field_key_yesno, &tans_P.Drag.mode },
   { 4,19, edit_field_get_yesno, edit_field_put_yesno, edit_field_key_yesno, &spas_P.Drag.mode },
   { 5, 8, edit_field_get_dbl,   orb_ballistic_put,    edit_field_key_dbl,   &orb_P.Drag.ballistic_number },
   { 5,18, edit_field_get_dbl,   edit_field_put_dbl,   edit_field_key_dbl,   &spas_P.Drag.ballistic_number },
   { 6, 5, edit_field_get_int,   edit_field_put_int,   edit_field_key_int,   &orb_P.Gravity.number_zonal },
   { 6,12, edit_field_get_int,   edit_field_put_int,   edit_field_key_int,   &tans_P.Gravity.number_zonal },
   { 6,19, edit_field_get_int,   edit_field_put_int,   edit_field_key_int,   &spas_P.Gravity.number_zonal },
   { 7, 5, edit_field_get_int,   edit_field_put_int,   edit_field_key_int,   &orb_P.Gravity.number_tesseral },
   { 7,12, edit_field_get_int,   edit_field_put_int,   edit_field_key_int,   &tans_P.Gravity.number_tesseral },
   { 7,19, edit_field_get_int,   edit_field_put_int,   edit_field_key_int,   &spas_P.Gravity.number_tesseral },
   { 8,12, edit_field_get_yesno, edit_field_put_yesno, edit_field_key_yesno, &noise },
   {11,18, edit_field_get_dbl,   drag_fields_put,      edit_field_key_dbl,   &orb_P.Drag.solar_flux },
   {12,23, edit_field_get_dbl,   drag_fields_put,      edit_field_key_dbl,   &orb_P.Drag.mean_solar_flux },
   {13,34, edit_field_get_dbl,   drag_fields_put,      edit_field_key_dbl,   &orb_P.Drag.geomagnetic_index_ap
}
};
#define FIELD1_COUNT (sizeof(fields1)/sizeof(Edit_field))

Edit_screen screen1 =
{
   LABEL1_COUNT, labels1,
   FIELD1_COUNT, fields1
};

/*=============================================================
============================
 *-------------------------------------------------------------*/
static Edit_screen *screens[] =
{
   &screen1
```

```
};
#define MAX_SCREENS (sizeof(screens)/sizeof(Edit_screen *))

static int current_screen;

/*=============================================================
============================
 *-------------------------------------------------------------*/
int sim_edit_display_init()
{
   edit_screen_init(screens[current_screen]);
   return YES;
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
int sim_edit_key_handler(int key)
{
   int rc = edit_screen_key_handler(screens[current_screen], key);

   if (rc > 0) return rc;

   /*-----------------------------------------------------------
    */
   switch (key)
   {
        case KEY_pgup:
           if (--current_screen < 0) current_screen = MAX_SCREENS - 1;
           break;

        case KEY_pgdn:
           if (++current_screen >= MAX_SCREENS) current_screen = 0;
           break;

        default: return NO;
   }

   edit_screen_init(screens[current_screen]);
   return YES;
}

/*=============================================================
============================
 * This function is designed to turn thrust commands in an LVC (local LVLH)
 * coordinate frame into M50 commanded delta v's to be applied before the
 * next step in the driver.
 */
void thrust(State_vector *target, State_vector *chaser,
            Vector *deltav_lvlh, Vector *deltav_M50, Perturbations *P)
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#include <stdio.h>
#include <alloc.h>

#include <edit.h>
#include "sim.h"

/*=============================================================
============================
 *-----------------------------------------------------------*/
#define MAX_SAMPLES (0xffff / sizeof(Sim_sample) - 1)

typedef struct _mem_buffer
{
   struct _mem_buffer *next;
   int count;
   Sim_sample samples[MAX_SAMPLES];
} Mem_buffer;

static Mem_buffer *buffer_first, *buffer_current;
static Sim_sample *sample_next;
static int sample_cnt;

/*=============================================================
============================
 *-----------------------------------------------------------*/
int sim_init()
{
   char *filename;
   FILE *infile;

   int samples;

   Mem_buffer *buffer, *buffer_prev;

   /*-------------------------------------------------------------
    */
   filename = edit_text("Enter input filename:", "sim.dat");

   if ((infile=fopen(filename, "rb")) == NULL)
   {
        printf("Can't open %s\n", filename);
        return NO;
   }

   /*-------------------------------------------------------------
    */
   fseek(infile, 0L, SEEK_END);
```

```
    samples = (int)(ftell(infile) / sizeof(Sim_sample));
    samples = edit_integer("How many samples should I load:", samples);

    if (samples <= 0)
    {
        fclose(infile);
        return NO;
    }
    fseek(infile, 0L, SEEK_SET);

    /*-------------------------------------------------------------
     */
    for (buffer_prev = NULL;
         samples > 0;
         samples -= MAX_SAMPLES, buffer_prev = buffer)
    {
        if ((buffer = malloc(sizeof(Mem_buffer))) == NULL)
        {
            printf("Ran out of memory, %d samples remaining\n", samples);
            fclose(infile);
            return NO;
        }

        if (buffer_first == NULL)  buffer_first = buffer;
        else                       buffer_prev->next = buffer;

        buffer->next = NULL;
        buffer->count = fread(buffer->samples, sizeof(Sim_sample), MAX_SAMPLES, infile);

        if (buffer->count <= 0)  samples = 0;
    }

    /*-------------------------------------------------------------
     */
    buffer_current = buffer_first;
    sample_cnt = buffer_current->count;
    sample_next = buffer_current->samples;

    fclose(infile);
    return YES;
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
void sim_stop() { }

/*=============================================================
==========================
 *-------------------------------------------------------------*/
```

```
{
   double magpos, magxxv, magvbar, dt, gcl[3][3];
   double step = 5.0;
   Vector mhhat, vb, pos_o, vel_o, xtxvt;

   /*-----------------------------------------------------------------
    */
   if( chaser->time >= target->time )
   {
        pos_o = chaser->pos;
        vel_o = chaser->vel;
   }
   else
   {
        dt = target->time - chaser->time;
        bgprop(chaser->time, &chaser->pos, &chaser->vel, P, dt,
        step, &pos_o, &vel_o);
   }

   /*-----------------------------------------------------------------
    */
   magpos = vector_magnitude(&pos_o);

   gcl[0][2] = -pos_o.x / magpos;
   gcl[1][2] = -pos_o.y / magpos;
   gcl[2][2] = -pos_o.z / magpos;

   /*-----------------------------------------------------------------
    */
   vector_cross(&pos_o, &vel_o, &xtxvt);
   magxxv = vector_magnitude(&xtxvt);

   gcl[0][1] =  mhhat.x =  -( xtxvt.x / magxxv);
   gcl[1][1] =  mhhat.y =  -( xtxvt.y / magxxv);
   gcl[2][1] =  mhhat.z =  -( xtxvt.z / magxxv);

   vector_cross(&pos_o, &mhhat, &vb);

   magvbar = vector_magnitude(&vb);

   gcl[0][0] = vb.x / magvbar;
   gcl[1][0] = vb.y / magvbar;
   gcl[2][0] = vb.z / magvbar;

   vector_transform(deltav_lvlh,deltav_M50,gcl);
}

/*==========================================================
==========================
* This is the "bad" function.  The inputs are the target state vector, the
```

```
* chaser state vector, and the time alloted for the rendezvous (ttr
* this must be in even increments of the propogation to allow for input of
* the second delta v). The output is the delta v's (in M50) to be input at
* the next step, and at the end of ttr for the chaser to effect a rendezvous
* with the target.
*/

void rdvz_delta_v(State_vector *target, State_vector *chaser,
                  double ttr, Vector *dv1_M50, Vector *dv2_M50,
                  Perturbations *P)
{
   static double mu = 398600.64;

   double step = 15.0;

   double magpos_t, magvel_t, magxxv, magvbar, lcg[3][3], gcl[3][3];
   double dt, delta;
   Vector xtxvt, hhat, vb, rdot, ro, dv1_M50_1;
   Vector pos_dif, dv1_lvlh, dv2_lvlh, omega, wxr_term;
   Vector pos_t, vel_t, pos_o, vel_o, pos_tnew, vel_tnew;
   double T,w;
   int i,j;

/*
* first match the time tags by propogating target to chaser regardless of
* which is more recent. This is artificial due to the restriction of
* applying delta v's only at integrator (sim) time steps for chaser.
*/
        dt = chaser->time - target->time;
        if(fabs(dt) > 1.0e-8)
         {
           bgprop(target->time, &target->pos, &target->vel, P, dt,
                  step, &pos_t, &vel_t);
         }
        else
         {
           pos_t = target->pos;
           vel_t = target->vel;
         }

        pos_o = chaser->pos;
        vel_o = chaser->vel;


   /*-----------------------------------------------------------------
    */

   magpos_t = vector_magnitude(&pos_t);
   magvel_t = vector_magnitude(&vel_t);
   T = 2*PI*sqrt(mu*mu/(pow(2*mu/magpos_t - magvel_t*magvel_t,3)));
```

```
w = 2*PI/T;

lcg[1][0] = pos_t.x / magpos_t;
lcg[1][1] = pos_t.y / magpos_t;
lcg[1][2] = pos_t.z / magpos_t;

/*-----------------------------------------------------------------
 */
vector_cross(&pos_t,&vel_t, &xtxvt);
magxxv = vector_magnitude(&xtxvt);

lcg[2][0] =  hhat.x = xtxvt.x / magxxv;
lcg[2][1] =  hhat.y = xtxvt.y / magxxv;
lcg[2][2] =  hhat.z = xtxvt.z / magxxv;

vector_cross(&pos_t, &hhat, &vb);

magvbar = vector_magnitude(&vb);

lcg[0][0] = vb.x / magvbar;  pos_dif.x = pos_o.x - pos_t.x;
lcg[0][1] = vb.y / magvbar;  pos_dif.y = pos_o.y - pos_t.y;
lcg[0][2] = vb.z / magvbar;  pos_dif.z = pos_o.z - pos_t.z;


vector_transform(&pos_dif, &ro, lcg);

/*-----------------------------------------------------------------
 */


rdot.x = ro.x*sin(w*ttr) + ro.y*(6*w*ttr*sin(w*ttr)-14*(1 - cos(w*ttr)));
delta = 3*w*ttr*sin(w*ttr) - 8*(1 - cos(w*ttr));
rdot.x = rdot.x/delta;
rdot.y = ro.x+(4*rdot.x-6*ro.y)*sin(w*ttr)+(6*ro.y-3*rdot.x)*w*ttr;
rdot.y = rdot.y/(-2)/(1-cos(w*ttr));
rdot.x = rdot.x*w;
rdot.y = rdot.y*w;

rdot.z = -ro.z*w/tan(w*ttr);

omega.x = 0.0;
omega.y = 0.0;
omega.z = w;

vector_cross(&omega, &ro, &wxr_term);


dv1_lvlh.x = rdot.x + wxr_term.x;
dv1_lvlh.y = rdot.y + wxr_term.y;
dv1_lvlh.z = rdot.z + wxr_term.z;
```

```
    for(j=0; j<3; j++)
    {
        for(i=0; i<3; i++)
        {
          gcl[i][j] = lcg[j][i];
        }
    }

    vector_transform(&dv1_lvlh, &dv1_M50_1, gcl);

    dv1_M50->x = vel_t.x + dv1_M50_1.x - vel_o.x;
    dv1_M50->y = vel_t.y + dv1_M50_1.y - vel_o.y;
    dv1_M50->z = vel_t.z + dv1_M50_1.z - vel_o.z;

    /*-----------------------------------------------------------------
     */

    dv2_lvlh.x = (2*rdot.y*sin(w*ttr)+(4*rdot.x-6*w*ro.y)*cos(w*ttr)+
                  6*w*ro.y-3*rdot.x);
    dv2_lvlh.y = ((3*w*ro.y - 2*rdot.x)*sin(w*ttr) + rdot.y*cos(w*ttr));
    dv2_lvlh.z = rdot.z*cos(w*ttr) - ro.z*w*sin(w*ttr);

/*
* next we have to propogate the target again to a time ttr plus where we
* we are right now (a second propagation)
*/

    /*-----------------------------------------------------------------
     */
    bgprop(chaser->time, &pos_t, &vel_t, P, ttr, step, &pos_tnew, &vel_tnew);

        magpos_t = vector_magnitude(&pos_tnew);

        gcl[0][1] = pos_tnew.x / magpos_t;
        gcl[1][1] = pos_tnew.y / magpos_t;
        gcl[2][1] = pos_tnew.z / magpos_t;

        /*-----------------------------------------------------------------
         */
        vector_cross(&pos_tnew, &vel_tnew, &xtxvt);
        magxxv = vector_magnitude(&xtxvt);

        gcl[0][2] =  hhat.x = xtxvt.x / magxxv;
        gcl[1][2] =  hhat.y = xtxvt.y / magxxv;
        gcl[2][2] =  hhat.z = xtxvt.z / magxxv;

        vector_cross(&pos_tnew, &hhat, &vb);

        magvbar = vector_magnitude(&vb);
```

```
        gcl[0][0] = vb.x / magvbar;
        gcl[1][0] = vb.y / magvbar;
        gcl[2][0] = vb.z / magvbar;

        vector_transform(&dv2_lvlh,dv2_M50,gcl);

}

/*=============================================================================
============================
 * getquat(*quat, *orbiter)...produces a quaternian for the attitude of the
 * orbiter "Qbi" based on its position and velocity.  This is by no means
 * an implication that this propogates the orbiter attitude.  This is however
 * a way of creating a time varying quaternian based on a desired attitude.
 * this function is currently designed to produce a gravity gradient position
 * with the payload bay facing aft, as if preparing to rendezvous.
 *-----------------------------------------------------------------------------*/
void getquat(void)
{
   Vector pos_o, vel_o, b1, b2, b3, rxv, b1xb2;
   double A, B, C, magpos, magrxv, magb1xb2;
   int good_quat, i;

   pos_o = tans.pos;
   vel_o = tans.vel;

   /*-------------------------------------------------------------------------
    */
   magpos = vector_magnitude(&pos_o);
   b1.x = pos_o.x/magpos;
   b1.y = pos_o.y/magpos;
   b1.z = pos_o.z/magpos;

   vector_cross(&pos_o,&vel_o,&rxv);
   magrxv = vector_magnitude(&rxv);
   b2.x = -rxv.x/magrxv;
   b2.y = -rxv.y/magrxv;
   b2.z = -rxv.z/magrxv;

   vector_cross(&b1,&b2,&b1xb2);
   magb1xb2 = vector_magnitude(&b1xb2);
   b3.x = b1xb2.x/magb1xb2;
   b3.y = b1xb2.y/magb1xb2;
   b3.z = b1xb2.z/magb1xb2;

   /*-------------------------------------------------------------------------
    */
   A = b1.y + b2.x;
   B = b2.z + b3.y;
   C = b1.z + b3.x;
```

```
    if( fabs(B) >= 1.0e-8 )
    {
        orb_quat.x = sqrt(A*C/4.0/B);
        if( fabs(orb_quat.x) >= 1.0e-8 )
        {
            orb_quat.y = A/4.0/orb_quat.x;
            orb_quat.z = C/4.0/orb_quat.x;
            if( fabs(orb_quat.y) >= 1.0e-8 )
            {
                orb_quat.s = (b1.z - 2*orb_quat.x*orb_quat.z)/2/orb_quat.y;
            }
            else
            {
                orb_quat.y = orb_quat.z = 0.0;
                orb_quat.s = orb_quat.x = 0.7071;
            }
        }
        else
        {
            orb_quat.y = orb_quat.z = 0.0;
            orb_quat.s = orb_quat.x = 0.7071;
        }
    }
    else
    {
        orb_quat.y = orb_quat.z = 0.0;
        orb_quat.s = orb_quat.x =  0.7071;
    }

    /*-------------------------------------------------------------
     */
    good_quat = NO;
    orb_quat.x = -orb_quat.x;
    i = 0;
    while (good_quat == NO)
    {
        orb_quat.x = -orb_quat.x;
        good_quat = testquat(&b1,&b2,&b3);
        i++;
        if (i > 2)
        {
            orb_quat.y = orb_quat.z =  0.0;
            orb_quat.s = orb_quat.x =  0.7071;
            good_quat = YES;
        }
    }
    orb_quat.time = tans.time;
}

/*==========================================================
```

```
=========================
 * This function evaluates the valitity of the relationship between a
 * quaternion and a DCM composed of three orthonormal vectors as the rows
 *-------------------------------------------------------------------*/
int testquat(Vector *b1, Vector *b2, Vector *b3)
{
   double dif;
   int good_quat;

   dif = fabs(b1->y - 2*(orb_quat.x*orb_quat.y-orb_quat.s*orb_quat.z));
   dif = dif + fabs(b1->z - 2*(orb_quat.x*orb_quat.z+orb_quat.s*orb_quat.y));
   dif = dif + fabs(b2->x - 2*(orb_quat.x*orb_quat.y+orb_quat.s*orb_quat.z));
   dif = dif + fabs(b3->x - 2*(orb_quat.x*orb_quat.z-orb_quat.s*orb_quat.y));
   dif = dif + fabs(b3->y - 2*(orb_quat.y*orb_quat.z+orb_quat.s*orb_quat.x));
   dif = dif + fabs(b2->z - 2*(orb_quat.y*orb_quat.z-orb_quat.s*orb_quat.x));

   if( dif < 1.e-6 )
   {
        good_quat = YES;
   }
   else
   {
        good_quat = NO;
   }
   return(good_quat);
}
```

```
;=============================================================================
;*  Public Domain
;*              Thomas J. Silva and Bruce G. Jackson & Associates, Inc.
;*
ideal
model large,C

include "com_port.inc"

;=============================================================================
codeseg
locals __
PORT  equ (Com_port_info es:bx)
BFR   equ (Com_port_buffer es:bx)

;=============================================================================
; int _com_port_reset(Com_port_info *port)
;
;  I check for the existance of the com-chip, then I reset the chip a known
; initial state.  I return an existance flag:
;               ax = 0 NO, chip doesn't respond
;               ax > 0 OK, chip initialized to: 9600-n-8-1, no interrupts
;
; Trashed:      ax bx cx dx  es
;-----------------------------------------------------------------------
REG_2_INIT    = ACC_w2_rx_fifo_14 + ACC_w2_fifo_on
REG_4_INIT    = ACC_rw4_RTS + ACC_rw4_DTR + ACC_rw4_output2

proc _com_port_reset
        arg arg_PORT:ptr Com_port_info

        ;-----------------------------------------------------------------------
        les             bx,[arg_PORT]
        mov             dx,[PORT.port]

        ;-----------------------------------------------------------------------
        ;  Test for the chip's presence by loading the divisor latch and reading it
        ; back.
        ;
        ACC_WR          3,mov,ACC_rw3_dlab    ;Access the divisor latches
        ACC_WR          0,mov,0fah                    ;load up registers 0 & 1 with a pattern
        ACC_WR          1,mov,0deh
        ACC_RD          0
        mov             ch,al
        ACC_RD          1
        mov             cl,al
        ACC_WR          3,mov,0                               ;Turn off divisor access
```

```
        cmp             cx,0fadeh
        jz              __reset                         ;if (register != pattern)  return NO
                xor             ax,ax
                jmp             short __done

        ;-----------------------------------------------------------------
        ;  Put the chip into a known state: interrupts off, fifos enabled, 9600
        ; baud, RTS DTR and OUT2 enabled
        ;
        __reset:                                ;Initialize the chip:
        ACC_WR      1,mov,0                                 ; Turn off interrupts
        ACC_WR      2,mov,REG_2_INIT    ; Fifos enabled, 14-char rx fifo trigger
        ACC_WR      3,mov,ACC_rw3_dlab  ; Divisor latch access
        ACC_WR      0,mov,0ch                       ;  BRG count = 12, 9600 baud (with x16 clock)
        ACC_WR      1,mov,00h
        ACC_WR      3,mov,ACC_rw3_8bits ; DLAB off, rx/tx 8 bits, no parity, 1 stop
        ACC_WR      4,mov,REG_4_INIT    ; RTS, DTR, OUT2 (connects interrupt line)

        ACC_RD      5
        ACC_RD      6                                       ;read line/modem status to clear them

        mov             al,1                            ;return OK

        ;-----------------------------------------------------------------
        __done:
        ret
endp

;=================================================================
============================
; void _com_port_set(Com_port_info *port, ushort divisor, int parity, int stops)
;
;  I set the baud-rate divisor latches, the caller must compute the divisor.
;
; Trashed:      ax bx dx  es
;-----------------------------------------------------------------------
proc _com_port_set
        arg arg_PORT:ptr Com_port_info, arg_DIVISOR:word, arg_PARITY:word, arg_STOPS:word

        ;-----------------------------------------------------------------
        les             bx,[arg_PORT]
        mov             dx,[PORT.port]
        mov             bx,[arg_DIVISOR]

        ACC_WR      3,mov,ACC_rw3_dlab + ACC_rw3_8bits
        ACC_WR      0,mov,bl
        ACC_WR      1,mov,bh
        mov             ax,[arg_PARITY]
        or              ax,[arg_STOPS]
        and             al,1ch
```

```
        ACC_WR      3,or,ACC_rw3_8bits

        ret
endp

;=============================================================================
; void _com_port_intr_catch(Com_port_info *port)
;
;  I perform these tasks:
;               Save the old interrupt handler
;               Create the new interrupt handler entry point
;               Install the new handler
;               Enable the interrupt control line
;               Enable the chip's interrupts
;
; Trashed:      ax bx cx dx  es
;-----------------------------------------------------------------------------
proc _com_port_intr_catch uses ds
        arg arg_PORT:ptr Com_port_info

        ;---------------------------------------------------------------------
        ;  Get and save the current interrupt handler.
        ;
        les             bx,[arg_PORT]
        mov             al,[PORT.irq]               ;al = base interrupt id + irq
        add             al,PC_int_irq0
        mov             cl,al
        mov             ah,35h
        int             21h                                 ;get and save the current handler
        mov             ax,bx
        mov             dx,es

        les             bx,[arg_PORT]
        mov             [PORT.old_handler.off],ax
        mov             [PORT.old_handler.seg],dx

        ;---------------------------------------------------------------------
        ;  Create a small interrupt handler at the beginning of the port structure.
        ; This code will call the base-handler below, which will then be able to
        ; find the port structure.
        ;
        mov             [PORT.entry.op_es],OP_PUSH_ES
        mov             [PORT.entry.op_bx],OP_PUSH_BX
        mov             [PORT.entry.op_nop],OP_NOP
        mov             [PORT.entry.op_call],OP_CALLF
        mov             [PORT.entry.handler.off],offset _handler
        mov             [PORT.entry.handler.seg],cs

        ;---------------------------------------------------------------------
```

```
        ;  Install the new interrupt handler for this IRQ.
        ;
        mov             dx,es
        mov             ds,dx
        mov             dx,bx                                   ;ds:dx -> port structure
        mov             ah,25h
        mov             al,cl
        int             21h                                             ;install the proper interrupt
handler

        ;---------------------------------------------------------------
        ;  Tell the interrupt controller to enable my interrupt line.
        ;
        mov             cl,[PORT.irq]
        mov             ah,1
        shl             ah,cl
        mov             [PORT.irq_mask],ah              ;my bit mask = 1 << irq

        cli                                                             ;disable interrupts while I change
the mask
        in              al,PC_IC0_r1                    ;al = current interrupts-enabled mask
        mov             cl,al
        not             ah
        and             al,ah                                   ;al = current_mask & ~my_mask
        out             PC_IC0_r1,al                    ;set new interrupt mask with my interrupt on
        sti

        not             ah
        and             cl,ah                                   ;I must remember is the interrupt was off
        jz              __do_chip
                or              [PORT.status],COM_PORT_irq_was_off

        ;---------------------------------------------------------------
        ;  Tell the chip to enable interrupts for rx and/or tx.
        ;
        __do_chip:
        mov             ax,[PORT.status]
        xor             cl,cl                                   ;initial mask = 0

        test    ax,COM_PORT_rx_enabled
        jz              __chk_tx                                ;if (rx_enabled)
                or              cl,ACC_rw1_int_rx               ;mask |= rx_int_mask

        __chk_tx:
        test    ax,COM_PORT_tx_enabled
        jz              __set_ier                               ;if (tx_enabled)
                or              cl,ACC_rw1_int_tx               ;mask |= tx_int_mask

        __set_ier:
        mov             dx,[PORT.port]
```

```
        ACC_WR      1,mov,cl                        ;Enable interrupts

        ;-------------------------------------------------------------
        ret
endp

;=============================================================================
; void _com_port_intr_uncatch(Com_port_info *port)
;
;  I perform these tasks:
;               Disable the chip's interrupts
;               Disable the interrupt controller (for my interrupt)
;               Restore the old handler
;
; Trashed:      ax bx cx dx  es
;-----------------------------------------------------------------------
proc _com_port_intr_uncatch uses ds
        arg arg_PORT:ptr Com_port_info

        ;-------------------------------------------------------------
        ;  Tell the chip to disable all interrupts.
        ;
        les             bx,[arg_PORT]
        mov             dx,[PORT.port]
        ACC_WR          1,mov,0                                 ;Disable interrupts

        ;-------------------------------------------------------------
        ;  Tell the interrupt controller to disable my interrupt line, but only if
        ; it was disabled when I began.
        ;
        test    [PORT.status],COM_PORT_irq_was_off
        jz              __do_handler                    ;if (it was disabled)
                cli
                in              al,PC_IC0_r1                    ;al = current interrupts-enabled mask
                or              al,[PORT.irq_mask]              ;al |= my_mask
                out             PC_IC0_r1,al                    ;turn off my interrupt line
                sti

        ;-------------------------------------------------------------
        __do_handler:
        les             bx,[arg_PORT]
        mov             al,[PORT.irq]                   ;al = base interrupt id + irq
        add             al,PC_int_irq0
        mov             ah,25h
        lds             dx,[PORT.old_handler.ptr]
        int             21h                                     ;restore the old handler

        ;-------------------------------------------------------------
        ret
```

```
endp

;=============================================================================
; int _com_port_get(Com_port_info *port)
;
;  I get the next byte from the receive queue.  I return the value as follows:
;                       ax = 0xffff  Queue empty (-1)
;                       ax = 0x00xx  Low byte has value, high byte is zero
;
; Trashed:      ax bx  es
;-----------------------------------------------------------------------
proc _com_port_get uses ds si
        arg arg_PORT:ptr Com_port_info

        ;-------------------------------------------------------------
        les             bx,[arg_PORT]
        lea             bx,[PORT.rx]            ;es:bx -> rx buffer
        cld
        cli
        call    near _buffer_get ;get one byte from buffer
        sti

        ;-------------------------------------------------------------
        jnz             __got_byte                      ;if (queue empty)
                xor             ax,ax
                dec             ax                                      ;value = -1
                jmp             short __done
        __got_byte:                                     ;else
                xor             ah,ah                           ;high_byte = 0

        ;-------------------------------------------------------------
        __done:
        ret
endp

;=============================================================================
; int _com_port_put(Com_port_info *port, ushort value)
;
;  I put a byte into the transmit queue.  I return a flag as follows:
;                       ax = 0  Queue full
;                       ax = 1  Ok
;
; Trashed:      ax bx dx  es
;-----------------------------------------------------------------------
proc _com_port_put uses ds di
        arg arg_PORT:ptr Com_port_info, arg_VALUE:word

        ;-------------------------------------------------------------
```

```
        mov             ax,[arg_VALUE]          ;al = byte to send
        les             bx,[arg_PORT]
        cld

        cli
        test    [PORT.status],COM_PORT_tx_stopped

        ;-----------------------------------------------------------------
        jz              __tx_put                ;if (tx_stopped)
                mov             dx,[PORT.port]
                out             dx,al                   ;send the byte
                and             [PORT.status],not COM_PORT_tx_stopped
                mov             ax,1
                jmp             short __done

        ;-----------------------------------------------------------------
        __tx_put:                               ;else
                lea             bx,[PORT.tx]            ;es:bx -> tx buffer
                call    near _buffer_put ;put one byte into buffer

        ;-----------------------------------------------------------------
        __done:
        sti
        ret
endp

;=================================================================
=======================
;  I get one byte from a buffer.
;
; Inputs:       es:bx -> buffer structure
;                       direction up
; Returns:      ah = 0 if no data, >0 if ok
;                       al = value
;                       ZR/NZ follows ah
; Trashed:      ax  si  ds
;-----------------------------------------------------------------
proc _buffer_get near

        mov             ax,[BFR.bused]
        or              ax,ax
        jz              __done                          ;if (buffer.used == 0)  return NO

        lds             si,[BFR.bstart.ptr]
        add             si,[BFR.btail]          ;ds:si -> tail byte

        lodsb                                           ;al = *tail++

        sub             si,[BFR.bstart.off]
        cmp             si,[BFR.bsize]
```

```
        jb              __set_tail                              ;if (tail > size)
                xor             si,si                                   ;tail = 0
        __set_tail:
        mov             [BFR.btail],si
        dec             [BFR.bused]
        or              ah,1                                    ;return YES

        __done:
        ret
endp

;=============================================================================
;  I put one byte into a buffer.
;
; Inputs:       es:bx -> buffer structure
;                       al = value
; Returns:      ax = 0 if full, 1 if ok
;                       ZR/NZ follows ax
; Trashed:      ax  di  ds
;-----------------------------------------------------------------------------
proc _buffer_put near

        mov             di,[BFR.bused]
        cmp             di,[BFR.bsize]
        jb              __put_byte                              ;if (buffer.used >= buffer.size)  return
NO
                xor             ax,ax
                ret

        __put_byte:
        lds             di,[BFR.bstart.ptr]
        add             di,[BFR.bhead]                  ;ds:di -> head byte

        mov             [di],al                                 ;al = *head++
        inc             di

        sub             di,[BFR.bstart.off]
        cmp             di,[BFR.bsize]
        jb              __set_head                              ;if (head > size)
                xor             di,di                                   ;head = 0
        __set_head:
        mov             [BFR.bhead],di
        inc             [BFR.bused]
        xor             ax,ax
        inc             ax                                              ;return YES
        ret
endp

;=============================================================================
```

```
;============================
;  I am the interrupt function for the tx_empty condition.
;
; Inputs:       es:bx -> port info structure
;                       dx = port
;
; Trashed:      ax cx  si  ds
;-------------------------------------------------------------------
proc _tx_empty near

        ;-----------------------------------------------------------
        mov             cx,bx
        lea             bx,[PORT.tx]            ;es:bx -> tx buffer

        ;-----------------------------------------------------------
        ;  Loop while the chip can take more bytes.
        ;
        __loop:                                 ;for (;;)
                ACC_RD      5                                   ;get line status
                test    al,ACC_r5_tx_empty
                jz              __done                  ;if (tx full)  break
                call    _buffer_get             ;get one byte from buffer
                jz              __tx_stopped    ;if (got one ok)
                        out             dx,al                   ;send the byte
                        jmp             __loop                  ;continue
                __tx_stopped:                   ;else
                        mov             bx,cx
                        or              [PORT.status],COM_PORT_tx_stopped

        ;-----------------------------------------------------------
        __done:
        mov             bx,cx
        ret
endp

;============================================================
============================
;  I am the interrupt function for the rx_ready condition.
;
; Inputs:       es:bx -> port info structure
;                       dx = port
;
; Trashed:      ax cx  di  ds
;-------------------------------------------------------------------
proc _rx_ready near

        ;-----------------------------------------------------------
        mov             cx,bx
        lea             bx,[PORT.rx]            ;es:bx -> rx buffer
```

```
        ;-----------------------------------------------------------
        ;  Loop while the chip has more bytes.
        ;
        __loop:                                         ;for (;;)
                in              al,dx                           ;get byte
                call    _buffer_put                     ;put one byte into buffer
                ACC_RD          5                                       ;get line status
                test    al,ACC_r5_rx_avail
                jnz             __loop                          ;if (rx not ready)  break

        ;-----------------------------------------------------------
        __done:
        mov             bx,cx
        ret
endp

;=================================================================
=========================
;  I am the interrupt handler, I expect to be called by the snippets of code
; created above by the _catch routine.
;-----------------------------------------------------------------
proc _handler

        ;-----------------------------------------------------------
        ;  Retrieve the port_info pointer off the stack, es and bx have been saved
        ; on the stack, I must save others.
        ;
        sti
        cld
        pop             bx
        pop             es
        sub             bx,8                            ;es:bx -> port pointer
        push    ds
        pusha

        ;-----------------------------------------------------------
        ;  Loop while I've got something to do.  The es, bx, and dx registers must
        ; be maintained across subroutine calls.
        ;
        mov             dx,[PORT.port]

        __loop:                                         ;while (interrupts pending)
                cli
                ACC_RD          2                                       ;get interrupt id flags
                test    al,ACC_r2_int_pending
                jnz             __int_clr

                xor             ah,ah
                and             al,ACC_r2_int_type
                mov             si,ax
```

```
                jmp             [cs:switch+si]          ;switch (interrupt_type)

                switch  dw      __case_modem_stat, __case_tx_empty
                        dw      __case_rx_avail, __case_line_stat

                ;-------------------------------------------------------------
                __case_modem_stat:                      ;case modem_status:
                        ACC_RD      6                                   ;read modem
status to clear
                        jmp         __loop

                ;-------------------------------------------------------------
                __case_line_stat:                       ;case line_status:
                        ACC_RD      5                                   ;read line status
to clear
                        jmp         __loop

                ;-------------------------------------------------------------
                __case_rx_avail:                        ;case rx_available:
                        call    _rx_ready                       ;empty the chip's rx queue
;                       jmp         __loop                      ;fall through to tx chk

                ;-------------------------------------------------------------
                __case_tx_empty:                        ;case tx_empty:
                        call    _tx_empty                       ;fill the chip's tx queue
                        jmp         __loop

        ;---------------------------------------------------------------
        ;  Clear the interrupt controller, release this interrupt line and all the
        ; lower priority lines.
        ;
        __int_clr:
        cli
        mov             al,IC_eoi
        out             PC_IC0_r0,al            ;End-of-interrupt command

        __done:
        popa
        pop             ds bx es
        iret
endp

;===============================================================================
;-------------------------------------------------------------------
public _com_port_reset, _com_port_set
public _com_port_intr_catch, _com_port_intr_uncatch
public _com_port_get, _com_port_put

end
```

```
/*=============================================================
============================
 *  Public Domain
 *      Thomas J. Silva and Bruce G. Jackson & Associates, Inc.
 *
 *-------------------------------------------------------------*/
#include <alloc.h>

#include <com_port.h>

static Com_port_info com_ports[COM_PORT_max] =
{
   /* entry      status                    port     irq */
   { {0},       COM_PORT_tx_stopped,      0x03f8,  4 },
   { {0},       COM_PORT_tx_stopped,      0x02f8,  3 },
   { {0},       COM_PORT_tx_stopped,      0x03e8,        4 },
   { {0},       COM_PORT_tx_stopped,      0x02e8,        3 }
};

/*=============================================================
============================
 *-------------------------------------------------------------*/
static Com_po[illegible] *com_port_verify(int port_id)
{
   if (port_id > 0  &&  port_id <= COM_PORT_max)
   {
        Com_port_info *port = com_ports + port_id - 1;
        if (port->status & COM_PORT_in_use)  return port;
   }
   return NULL;
}

/*=============================================================
============================
 * Returns:
 *      0  No: bad id, port in use, chip doesn't respond, or out of memory
 *      1  Ok:
 *-------------------------------------------------------------*/
int com_port_start(int port_id, ushort rx_bsize, ushort tx_bsize)
{
   Com_port_info *port = com_ports + port_id - 1;

   /*-----------------------------------------------------------
    *  Check the port number for validity, make sure it's not already in use.
    * Then make sure the chip for that port responds (i.e. it exists).
    */
   if (port_id <= 0  ||  port_id > COM_PORT_max
    || port->status & COM_PORT_in_use)                    return NO;

   if (_com_port_reset(port) == NO)  return NO;
```

```
    /*-----------------------------------------------------------------
     *  Allocate the rx and tx buffers.
     */
    if (rx_bsize)
    {
        if ((port->rx.start = malloc(port->rx.size = rx_bsize)) == NULL)
            return NO;
        port->rx.used = port->rx.head = port->rx.tail = 0;
    }

    if (tx_bsize)
    {
        if ((port->tx.start = malloc(port->tx.size = tx_bsize)) == NULL)
        {
            if (rx_bsize) free(port->rx.start);
            return NO;
        }
        port->tx.used = port->tx.head = port->tx.tail = 0;
    }

    /*-----------------------------------------------------------------
     *  This port is now in use, it is not currently transmitting.
     */
    port->status = COM_PORT_in_use | COM_PORT_tx_stopped;

    if (rx_bsize) port->status |= COM_PORT_rx_enabled;
    if (tx_bsize) port->status |= COM_PORT_tx_enabled;

    /*-----------------------------------------------------------------
     *  Install the proper interrupt handler, and enable the chip's interrupts.
     */
    _com_port_intr_catch(port);

    return YES;
}

/*=================================================================
=========================
 * Returns:
 *      0  No: bad id, or port not in use
 *      1  Ok:
 *-----------------------------------------------------------------*/
int com_port_stop(int port_id)
{
    Com_port_info *port = com_port_verify(port_id);

    /*-----------------------------------------------------------------
     *  Shut off the chip's interrupts, and remove the interrupt handler.
     */
    if (port == NULL) return NO;
```

```
    _com_port_intr_uncatch(port);

    port->status = COM_PORT_tx_stopped;

    /*-------------------------------------------------------------
     *  Free the rx and tx buffers.
     */
    if (port->rx.size) { free(port->rx.start);  port->rx.size = 0; }

    if (port->tx.size) { free(port->tx.start);  port->tx.size = 0; }

    return YES;
}

/*=============================================================
============================
 * Returns:
 *       0  No: bad id, or port not in use
 *       1  Ok:
 *-------------------------------------------------------------*/
int com_port_set(int port_id, long baud, int parity_flag, int stop_bits)
{
    Com_port_info *port = com_port_verify(port_id);
    ushort divisor;

    /*-------------------------------------------------------------
     */
    if (port == NULL)  return NO;

    divisor = (ushort)(COM_PORT_clock / baud) / 16;

    _com_port_set(port, divisor, parity_flag, stop_bits);

    return YES;
}

/*=============================================================
============================
 * Returns:
 *       -1    No: bad id, port not in use, or queue empty
 *       00xx  Ok: high byte zero and value in low byte
 *-------------------------------------------------------------*/
int com_port_get(int port_id)
{
    Com_port_info *port = com_port_verify(port_id);

    /*-------------------------------------------------------------
     */
    if (port == NULL)  return -1;
```

```
    return _com_port_get(port);
}

/*=============================================================
============================
 * Returns:
 *       -1  No: bad id, or port not in use
 *        0  No: queue empty
 *       >0  Ok: number of bytes
 *-------------------------------------------------------------*/
#ifdef NOT_YET
int com_port_read(int port_id, ubyte *where, int size)
{
    Com_port_info *port = com_port_verify(port_id);

    /*-----------------------------------------------------------
     */
    if (port == NULL)  return -1;

    return _com_port_read(port, where, size);
}
#endif NOT_YET

/*=============================================================
============================
 * Returns:
 *       0  No: bad id, port not in use, or queue full
 *       1  Ok:
 *-------------------------------------------------------------*/
int com_port_put(int port_id, ubyte value)
{
    Com_port_info *port = com_port_verify(port_id);

    /*-----------------------------------------------------------
     */
    if (port == NULL)  return NO;

    return _com_port_put(port, value);
}

/*=============================================================
============================
 * Returns:
 *       -1  No: bad id, or port not in use
 *        0  No: queue full
 *       >0  Ok: number of bytes
 *-------------------------------------------------------------*/
#ifdef NOT_YET
int com_port_write(int port_id, ubyte *where, int size)
{
```

```
    Com_port_info *port = com_port_verify(port_id);

    /*-----------------------------------------------------------------
     */
    if (port == NULL)  return -1;

    return _com_port_write(port, where, size);
}
#endif NOT_YET
```

```
/*=============================================================
==========================
 *-------------------------------------------------------------*/
#include <stdio.h>
#include <io.h>
#include <fcntl.h>

#include <edit.h>
#include <pkt_if.h>

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static int file_status, file_handle_rx = -1;
// static int file_handle_tx = -1;
static FILE *file_rx = NULL;

#define FILE_IS_DEVICE      0x0080
#define FILE_NOT_EOF        0x0040
#define FILE_IS_RAW   0x0020
#define FILE_SET_BITS 0x00ff

/*=============================================================
==========================
 *-------------------------------------------------------------*/
int orb_port_init()
{
   char *filename;

   /*-----------------------------------------------------------
    */
   filename = edit_text("Enter Orbiter file/device:", "com2");

   return orb_port_init_byname(filename);
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
int orb_port_init_byname(char *filename)
{
   /*-----------------------------------------------------------
    */
   if ((file_handle_rx = open(filename, O_RDWR|O_BINARY)) < 0)
   {
        printf("Can't open file/device %s\n", filename);
        return NO;
   }

   /*-----------------------------------------------------------
```

```
    *  If input file is a device, place that device in "raw" mode.  Otherwise
    * reopen using buffered I/O.
    */
   file_status = ioctl(file_handle_rx, 0, 0,0);

   if (file_status & FILE_IS_DEVICE)
   {
        file_status = (file_status & FILE_SET_BITS) | FILE_IS_RAW | FILE_NOT_EOF;
        ioctl(file_handle_rx, 1, file_status,0);
        // file_handle_tx = file_handle_rx;
   }
   else
   {
        if ((file_rx = fdopen(file_handle_rx, "rb")) == NULL)
        {
           close(file_handle_rx);
           printf("Can't open %s, out of memory\n", filename);
           return NO;
        }

        // filename = edit_text("Enter Orbiter output file:", filename);
        // if ((file_handle_tx = open(filename, O_RDWR|O_BINARY|O_CREAT, 0666)) < 0)
        // {
           // printf("Can't create file %s\n", filename);
           // close(file_handle_rx);
           // return NO;
        // }
   }

   return YES;
}

/*=============================================================
=======================
 *-------------------------------------------------------------*/
int orb_port_stop()
{
   /*-------------------------------------------------------------
    *  If input file is a device, place that device back in "cooked" mode.
    */
   if (file_status & FILE_IS_DEVICE)
   {
        file_status &= ~FILE_IS_RAW;
        ioctl(file_handle_rx, 1, file_status,0);
        close(file_handle_rx);
   }
   else
   {
        fclose(file_rx);
        // close(file_handle_tx);
```

```
    }

    return YES;
}

/*=============================================================
============================
 *  DOS sucks.  DOS has a bug regarding devices in raw mode, this piece of code
 * fixes the error caused by the bug.  The bug occurs under these conditions:
 *      - You are reading from a device in raw mode.
 *      - The memory you are reading into just happens to fall on a paragraph
 *      boundary (i.e. the address is divisible by 16).
 *      - The device driver doesn't have any bytes ready for you.
 *
 *  Under these conditions, DOS will erroneously mark your file with an EOF
 * flag, after which read() will always fail, even if the device has some bytes
 * ready.  This kludge marks the file as not-EOF.
 *-------------------------------------------------------------*/
static void fix_dos_eof_bug(void *location)
{
    if (((ushort)location & 15) == 0  &&  file_status & FILE_IS_DEVICE)
    {
        file_status |= FILE_NOT_EOF;
        ioctl(file_handle_rx, 1, file_status,0);
    }
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
int orb_port_getb()
{
    /*-----------------------------------------------------------
     */
    if (file_status & FILE_IS_DEVICE)
    {
        ubyte work;
        if (read(file_handle_rx, &work, 1) > 0)  return work;
        fix_dos_eof_bug(&work);
        return -1;
    }
    else  return getc(file_rx);
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
#ifdef NOT_YET
int orb_port_read(ubyte *where, int bytes)
{
```

```
    if ((bytes = read(file_handle_rx, where, bytes)) <= 0)  fix_dos_eof_bug(where);
    return bytes;
}
#endif NOT_YET
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#include <pkt_if.h>
#include <packet.h>
#include <const.h>

/*=============================================================
============================
 *  The orbiter-gps packet is used for testing only.
 *
 *  The vectors have these units:
 *       orbiter vectors are M50 in ft and ft/s
 *       spas vectors are WGS84 in ft and ft/s
 *       orb-gps vectors are WGS84 in m and m/s
 *
 * The orbiter time is absolute, 001/00:00:01.000 == 1.0 sec + 86400 secs (one
 * day). The spas time is also absolute. The orb-gps time is also absolute.
 *-----------------------------------------------------------*/
Orb_packet orbiter_packet;
Spas_packet spas_packet;
Orb_gps_packet orbiter_gps_packet;

int orb_port_pkt_error;

/*=============================================================
============================
 *-----------------------------------------------------------*/
enum states
{
   STATE_find_eb,
   STATE_find_90,
   STATE_get_type,
   STATE_fill_pkt,
   STATE_use_pkt
};

static union
{
   Orb_packet orb;
   Spas_packet spas;
   Orb_gps_packet orb_gps;
   ubyte buffer[1];
} packet;

/*=============================================================
============================
 *-----------------------------------------------------------*/
static int check_packets = YES;
```

```
void orb_port_packet_check(int on_off) { check_packets = on_off; }

static int check_packet(void);
// static void dump_vector(State_vector *sv, char *msg);

/*==========================================================
==========================
 * Returns:
 *  0  Nothing available
 *  >0  Packet type
 *----------------------------------------------------------*/
int orb_port_process()
{
   static int state = STATE_find_eb;
   static int buffer_index = 0;
   static int bytes_needed;

   int packet_type;
   ushort value;

   /*-------------------------------------------------------
    *  Fill the packet buffer until: the packet is filled, or no more bytes
    * are available from the port.
    */
   for (packet_type = NO;
        packet_type == NO  &&  (value = orb_port_getb()) != (ushort)-1;
        packet.buffer[buffer_index] = value)
   {
        switch (state)
        {
           /*-------------------------------------------------
            */
           case STATE_find_eb:
                buffer_index = 0;
                if (value != 0x00eb)  break;
                state = STATE_find_90;
                break;

           /*-------------------------------------------------
            */
           case STATE_find_90:
                if (value == 0x0090)
                {
                   buffer_index++;
                   state = STATE_get_type;
                }
                else state = STATE_find_eb;
                break;

           /*-------------------------------------------------
```

```
         */
        case STATE_get_type:
             if (value == PKT_orbiter)   bytes_needed = sizeof(Orb_packet) - 3;
             else if (value == PKT_spas) bytes_needed = sizeof(Spas_packet) - 3;
             else if (value == PKT_gps)  bytes_needed = sizeof(Orb_gps_packet) - 3;
             else
             {
                state = STATE_find_eb;
                break;
             }
             state = STATE_fill_pkt;
             buffer_index++;
             break;

        /*-------------------------------------------------------------
         */
        case STATE_fill_pkt:
             buffer_index++;
             if (--bytes_needed > 0) break;

             if (value != 0x00ef  ||  packet.buffer[buffer_index-1] != 0x00be)
             {
                state = STATE_find_eb;
                break;
             }
             else state = STATE_use_pkt;

        /*-------------------------------------------------------------
         */
        case STATE_use_pkt:
             packet_type = check_packet();
             state = STATE_find_eb;
             break;
      }
   }

   /*-------------------------------------------------------------
    */
   switch (packet_type)
   {
        case PKT_orbiter: orbiter_packet = packet.orb;          break;
        case PKT_spas:    spas_packet = packet.spas;            break;
        case PKT_gps:     orbiter_gps_packet = packet.orb_gps;  break;
        default: break;
   }

   return packet_type;
}

/*=============================================================
```

```
============================
 *-------------------------------------------------------------*/
#define POS_LOW_FT 21000000.0
#define POS_HI_FT  23000000.0
#define VEL_LOW_FT 10000.0
#define VEL_HI_FT  40000.0

#define POS_LOW_M  (POS_LOW_FT * FT_TO_M)
#define POS_HI_M   (POS_HI_FT * FT_TO_M)
#define VEL_LOW_M  (VEL_LOW_FT * FT_TO_M)
#define VEL_HI_M   (VEL_HI_FT * FT_TO_M)

static State_vector_limits
   limit_ft = { { POS_LOW_FT, POS_HI_FT }, { VEL_LOW_FT, VEL_HI_FT } },
   limit_m  = { { POS_LOW_M,  POS_HI_M  }, { VEL_LOW_M,  VEL_HI_M  } };

/*============================================================
============================
 *-------------------------------------------------------------*/
static int check_packet()
{
   int packet_type = packet.orb.header.type;

   switch (packet_type)
   {
        case PKT_orbiter:
           if (check_packets)
           {
                if (state_vector__check(&packet.orb.orbiter_vector,  &limit_ft) == NO)
                {
                   // dump_vector(&packet.orb.orbiter_vector, "Orb");
                   packet_type = 0;
                   break;
                }
                if (state_vector__check(&packet.orb.target_vector,  &limit_ft) == NO)
                {
                   // dump_vector(&packet.orb.target_vector, "Tgt");
                   packet_type = 0;
                   break;
                }
           }
           break;

        case PKT_spas:
           if (check_packets)
           {
                if (state_vector__check(&packet.spas.gps_vector,  &limit_ft) == NO)
                {
                   // dump_vector(&packet.spas.gps_vector, "Spas");
                   packet_type = 0;
```

```
                break;
            }
        }
        break;

    case PKT_gps:
        if (check_packets)
        {
            if (state_vector__check(&packet.orb_gps.gps_vector, &limit_m) == NO)
            {
                // dump_vector(&packet.spas.gps_vector, "OrbGps");
                packet_type = 0;
                break;
            }
        }
        break;

    default: packet_type = 0; break;
  }

  return packet_type;
}

/*=============================================================================
============================
 *-----------------------------------------------------------------------------*/
#ifdef GIGO
#include <stdio.h>
#include <times.h>

static void dump_vector(State_vector *sv, char *msg)
{
  char work[32];

  time_dbl_to_string(&sv->time, work);
  fprintf(stderr, "%-6s %s\n", msg, work);
  fprintf(stderr,"\tpos %9.0f %9.0f %9.0f\n", sv->pos.x, sv->pos.y, sv->pos.z);
  fprintf(stderr,"\tvel %9.0f %9.0f %9.0f\n", sv->vel.x, sv->vel.y, sv->vel.z);
}
#endif GIGO
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>

#include <edit.h>
#include <com_port.h>
#include <pkt_if.h>

/*=============================================================
============================
 *-----------------------------------------------------------*/
static int port_id;

#define RX_SIZE 2048

static int _orb_port_init(int id, ushort rx_size);

/*=============================================================
============================
 *  I expect about 220 bytes per second, so a receive buffer of 2048 gives me
 * over 9 seconds of buffering.
 *-----------------------------------------------------------*/
int orb_port_init()
{
   static char msg[] = "Enter Orbiter port id (RS232 com1 or com2) > ";

   int id;
   // ushort rx_size = 2048;

   /*-------------------------------------------------------------
    */
   /* rx_size = edit_integer("        Enter rx size ", 512); */
   id = edit_integer(msg, 2);

   return _orb_port_init(id, RX_SIZE);
}

int orb_port_init_byname(char *port_name)
{
   return _orb_port_init(atoi(port_name), RX_SIZE);
}

/*=============================================================
============================
 *-----------------------------------------------------------*/
static int _orb_port_init(int id, ushort rx_size)
{
   /*-------------------------------------------------------------
```

```
    */
   if ((port_id = id) == 0)  return YES;

   if (com_port_start(port_id, rx_size, 0) == NO)
   {
        printf("Can't start port %d\n", port_id);
        port_id = 0;
        return NO;
   }

   com_port_set(port_id, 9600L, COM_PORT_par_none, COM_PORT_stop_1);

   return YES;
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
int orb_port_stop()
{
   if (port_id)  com_port_stop(port_id);
   return YES;
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
int orb_port_getb()
{
   return (port_id ? com_port_get(port_id) : -1);
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
#ifdef NOT_YET
int orb_port_read(ubyte *where, int bytes)
{
   return com_port_read(port_id, where, bytes);
}
#endif NOT_YET
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#include <tans_pkt.h>
#include <pkt_if.h>

/*=============================================================
============================
 *-----------------------------------------------------------*/
#define DLE 0x10
#define ETX 0x03

/*=============================================================
============================
 * Returns:
 *  0 NO = Still collecting bytes, nothing available yet
 *  1 YES = Got something, could be good or bad
 *-----------------------------------------------------------*/
int tans_packet_recv(Tans_packet *packet)
{
   int value;

   /*-------------------------------------------------------------
    */
   for (;;) switch (packet->state)
   {
        /*-------------------------------------------------------
         *  Here I get bytes and fill in the raw buffer until I find the DLE
         * that starts the packet.
         *
         *  If I have received more than just the DLE, then I return OK to give
         * my caller a chance to do something with the garbage data. I remove
         * the DLE from the buffer first: it is not part of the bad data.
         */
        case TANS_state__find_start:
           if ((value = tans_port_getb()) == -1) return NO;

           packet->raw[packet->raw_size] = (ubyte)value;

           if (++packet->raw_size >= TANS_PKT_SIZE_RAW)
           {
                packet->state = TANS_state__oversize;
                return YES;
           }

           if (value != DLE) continue;

           if (packet->raw_size != 1)
           {
                packet->raw_size--;
```

```
        packet->state = TANS_state__had_gigo;
        return YES;
    }
    packet->state = TANS_state__packet_start;
    continue;

/*-------------------------------------------------------------------
 *  Here I recover from the condition where I found bad bytes before
 * the start-of-packet.  I fall through to the next state.
 */
case TANS_state__had_gigo:
    packet->raw[0] = DLE;
    packet->raw_size = 1;
    /* packet->state = TANS_state__packet_start; --- Redundant */

/*-------------------------------------------------------------------
 *  Here I initialize the cooked buffer with the start-of-packet DLE.
 * I fall through to the next state.
 */
case TANS_state__packet_start:
    packet->cooked[0] = DLE;
    packet->cooked_size = 1;
    packet->state = TANS_state__find_end;

/*-------------------------------------------------------------------
 *  Here I get bytes and fill in both buffers until I find the DLE that
 * ends the packet.  I fall through to the next state when I find a DLE.
 */
case TANS_state__find_end:
    if ((value = tans_port_getb()) == -1)  return NO;

    packet->raw[packet->raw_size] =
    packet->cooked[packet->cooked_size] = (ubyte)value;

    if (++packet->raw_size >= TANS_PKT_SIZE_RAW
     || ++packet->cooked_size >= TANS_PKT_SIZE_COOKED)
    {
        packet->state = TANS_state__oversize;
        return YES;
    }

    if (value != DLE)  continue;

    packet->state = TANS_state__found_dle;

/*-------------------------------------------------------------------
 *  Here I get one byte and check for ETX or DLE.  I always save the
 * byte in the raw buffer, but not always in the cooked buffer.
 *  A DLE will send me back to the _find_end state, an ETX will send me
 * to the _packet_ok state, any other value sends me to _packet_bad.
```

```
 *  If the packet is done, then I return OK to give my caller a chance
 * to do something with the packet (good or bad).  For bad packets, I
 * remove the DLE and non-ETX bytes from the buffer first: they are not
 * part of the bad data.
 */
case TANS_state__found_dle:
   if ((value = tans_port_getb()) == -1)  return NO;

   packet->raw[packet->raw_size++] = (ubyte)value;

   if (value == DLE)
   {
        packet->state = TANS_state__find_end;
        continue;
   }

   if (value == ETX)
   {
        packet->cooked[packet->cooked_size++] = (ubyte)value;
        packet->state = TANS_state__packet_ok;
   }
   else
   {
        packet->raw_size -= 2;
        packet->state = TANS_state__packet_bad;
   }

   return YES;

/*-----------------------------------------------------------------
 *  Here I recover from the condition where I found a bad packet.  I've
 * got two bytes that might be ok: DLE and whatever I found after DLE.
 */
case TANS_state__packet_bad:
   packet->raw[0] = packet->cooked[0] = DLE;
   packet->raw[1] = packet->cooked[1] = packet->raw[packet->raw_size + 1];

   packet->raw_size = packet->cooked_size = 2;

   packet->state = TANS_state__find_end;
   continue;

/*-----------------------------------------------------------------
 *  Here I reset all the counters and start looking for DLE again.
 */
case TANS_state__packet_ok:
case TANS_state__oversize:
   packet->raw_size = packet->cooked_size = 0;
   packet->state = TANS_state__find_start;
   continue;
```

```
    }
}
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#include <stdio.h>
#include <io.h>
#include <fcntl.h>

#include <edit.h>
#include <pkt_if.h>

/*=============================================================
============================
 *-----------------------------------------------------------*/
static int file_status, file_handle_rx = -1;
// static int file_handle_tx = -1;
static FILE *file_rx = NULL;

#define FILE_IS_DEVICE      0x0080
#define FILE_NOT_EOF        0x0040
#define FILE_IS_RAW   0x0020
#define FILE_SET_BITS 0x00ff

/*=============================================================
============================
 *-----------------------------------------------------------*/
int tans_port_init()
{
   char *filename;

   /*-------------------------------------------------------
    */
   filename = edit_text("Enter TANS file/device:", "com1");

   return tans_port_init_byname(filename);
}

/*=============================================================
============================
 *-----------------------------------------------------------*/
int tans_port_init_byname(char *filename)
{
   /*-------------------------------------------------------
    */
   if ((file_handle_rx = open(filename, O_RDWR|O_BINARY)) < 0)
   {
        printf("Can't open file/device %s\n", filename);
        return NO;
   }

   /*-------------------------------------------------------
```

```
    *  If input file is a device, place that device in "raw" mode.  Otherwise
    * reopen using buffered I/O.
    */
   file_status = ioctl(file_handle_rx, 0, 0,0);

   if (file_status & FILE_IS_DEVICE)
   {
        file_status = (file_status & FILE_SET_BITS) | FILE_IS_RAW | FILE_NOT_EOF;
        ioctl(file_handle_rx, 1, file_status,0);
        // file_handle_tx = file_handle_rx;
   }
   else
   {
        if ((file_rx = fdopen(file_handle_rx, "rb")) == NULL)
        {
           close(file_handle_rx);
           printf("Can't open %s, out of memory\n", filename);
           return NO;
        }

        // filename = edit_text("Enter TANS output file:", filename);
        // if ((file_handle_tx = open(filename, O_RDWR|O_BINARY|O_CREAT, 0666)) < 0)
        // {
           // printf("Can't create file %s\n", filename);
           // close(file_handle_rx);
           // return NO;
        // }
   }

   return YES;
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
int tans_port_stop()
{
   /*-------------------------------------------------------------
    *  If input file is a device, place that device back in "cooked" mode.
    */
   if (file_status & FILE_IS_DEVICE)
   {
        file_status &= ~FILE_IS_RAW;
        ioctl(file_handle_rx, 1, file_status,0);
        close(file_handle_rx);
   }
   else
   {
        fclose(file_rx);
        // close(file_handle_tx);
```

```
    }

    return YES;
}

/*=============================================================
============================
 *   DOS has a bug regarding devices in raw mode, this piece of code
 * fixes the error caused by the bug.  The bug occurs under these conditions:
 *       - You are reading from a device in raw mode.
 *       - The memory you are reading into just happens to fall on a paragraph
 *       boundary (i.e. the address is divisible by 16).
 *       - The device driver doesn't have any bytes ready for you.
 *
 *  Under these conditions, DOS will erroneously mark your file with an EOF
 * flag, after which read() will always fail, even if the device has some bytes
 * ready.  This kludge marks the file as not-EOF.
 *-------------------------------------------------------------*/
static void fix_dos_eof_bug(void *location)
{
    if (((ushort)location & 15) == 0  &&  file_status & FILE_IS_DEVICE)
    {
        file_status |= FILE_NOT_EOF;
        ioctl(file_handle_rx, 1, file_status,0);
    }
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
int tans_port_getb()
{
    /*-----------------------------------------------------------
     */
    if (file_status & FILE_IS_DEVICE)
    {
        ubyte work;
        if (read(file_handle_rx, &work, 1) > 0)  return work;
        fix_dos_eof_bug(&work);
        return -1;
    }
    else  return getc(file_rx);
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
#ifdef NOT_YET
int tans_port_read(ubyte *where, int bytes)
{
```

```
    if ((bytes = read(file_handle_rx, where, bytes)) <= 0)  fix_dos_eof_bug(where);
    return bytes;
}
#endif NOT_YET

/*=============================================================================
=========================
 *   I will not write to
 *-----------------------------------------------------------------------*/
int tans_port_sendb(ubyte value)
{
    // return (write(file_handle_tx, &value, 1) <= 0 ? NO : YES);
}
```

```
/*=============================================================
==========================
 *-----------------------------------------------------------*/
#include <stdio.h>

#include <edit.h>
#include <com_port.h>
#include <pkt_if.h>

/*=============================================================
==========================
 *-----------------------------------------------------------*/
static int port_id;

/*=============================================================
==========================
 *-----------------------------------------------------------*/
int tans_port_init()
{
   static char msg[] = "Enter TANS port id (RS422 com3 or com4) > ";

   ushort rx_size = 2048;
   ushort tx_size = 512;

   /*-------------------------------------------------------------
    */
   port_id = edit_integer(msg, 3);
   if (port_id == 0)  return YES;
   /* rx_size = edit_integer("     Enter rx size ", 512); */
   /* rx_size = edit_integer("     Enter tx size ", 128); */

   if (com_port_start(port_id, rx_size, tx_size) == NO)
   {
        printf("Can't start port %d\n", port_id);
        port_id = 0;
        return NO;
   }

   com_port_set(port_id, 9600L, COM_PORT_par_odd, COM_PORT_stop_1);

   return YES;
}

/*=============================================================
==========================
 *-----------------------------------------------------------*/
int tans_port_stop()
{
   if (port_id)  com_port_stop(port_id);
   return YES;
```

```
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
int tans_port_getb()
{
    return (port_id ? com_port_get(port_id) : -1);
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
int tans_port_sendb(ubyte value)
{
    return (port_id ? com_port_put(port_id, value) : YES);
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
#ifdef NOT_YET
int tans_port_read(ubyte *where, int bytes)
{
    return com_port_read(port_id, where, bytes);
}
#endif NOT_YET
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#include <packet.h>
#include <pkt_if.h>
#include <tans_pkt.h>
#include <swap.h>
#include <times.h>

/*=============================================================
============================
 *  The orbiter-gps packet is actually owned by orb_port_process().
 *-----------------------------------------------------------*/
extern Orb_gps_packet orbiter_gps_packet;

/*=============================================================
============================
 *-----------------------------------------------------------*/
static struct
{
   Tans_packet packet;
   struct
   {
        int got;
        double prev;
        Tans_pkt_21_41 pkt;
        Time_gps gps;
        Time_utc utc;
   } time;
   struct { int got;  Tans_pkt_43 pkt; } vel;
   struct { int got;  Tans_pkt_83 pkt; } pos;
} my;

static int
   process_packet_41(void),
   process_packet_43(void),
   process_packet_83(void),
   check_packets(void);

/*=============================================================
============================
 * Returns:
 *  0  Nothing available
 *  >0  Packet type
 *-----------------------------------------------------------*/
int tans_port_process()
{
   int rc;

   /*-----------------------------------------------------------
```

```
    *  Get all the available packets until I find something I can use.
    */
   for (rc = NO;  rc == NO; )
   {
        if (tans_packet_recv(&my.packet) == NO
         || my.packet.state != TANS_state__packet_ok)  break;

        switch (my.packet.raw[1])
        {
           case 0x41: rc = process_packet_41(); break;
           case 0x43: rc = process_packet_43(); break;
           case 0x83: rc = process_packet_83(); break;

           default:  rc = NO;
        }

        if (rc)  rc = check_packets();
   }
   return rc;
}

/*=============================================================================
=========================
 *-----------------------------------------------------------------------*/
static int check_packets()
{
   /*-----------------------------------------------------------------------
    *  Week rollover is not handled perfectly, some data may be tossed out.
    */
   if (my.time.got  &&  my.pos.got  &&  my.vel.got)
   {
        if (my.vel.pkt.sol_time > my.pos.pkt.sol_time)              my.pos.got = NO;
        else if (my.vel.pkt.sol_time < my.pos.pkt.sol_time)         my.vel.got = NO;
        else if (my.vel.pkt.sol_time <= 0.0)       my.pos.got =  my.vel.got =  NO;
        else
        {
           if (my.vel.pkt.sol_time < my.time.prev)  my.time.pkt.week++;
           my.time.prev = my.vel.pkt.sol_time;

           my.time.gps.secs = my.vel.pkt.sol_time;
           my.time.gps.week = my.time.pkt.week;
           time_gps_to_utc(&my.time.gps, my.time.pkt.gps_utc_offset, &my.time.utc);

           orbiter_gps_packet.gps_vector.pos = my.pos.pkt.pos;
           orbiter_gps_packet.gps_vector.vel.x = my.vel.pkt.vel.x;
           orbiter_gps_packet.gps_vector.vel.y = my.vel.pkt.vel.y;
           orbiter_gps_packet.gps_vector.vel.z = my.vel.pkt.vel.z;
           orbiter_gps_packet.gps_vector.time = my.time.utc.secs;
           my.pos.got =  my.vel.got = NO;
```

```
            return orbiter_gps_packet.header.type = PKT_gps;
        }
    }

    return NO;
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static int process_packet_41()
{
    int size;
    Tans_pkt_21_41 *packet;

    /*-------------------------------------------------------------
     */
    size = my.packet.cooked_size - sizeof(Tans_pkt_head) - sizeof(Tans_pkt_tail);

    if (size != sizeof(Tans_pkt_21_41))  return NO;

    packet = (Tans_pkt_21_41 *)(my.packet.cooked + sizeof(Tans_pkt_head));

    /*-------------------------------------------------------------
     */
    swap_float(&packet->secs_of_week, &packet->secs_of_week);
    swap_float(&packet->gps_utc_offset, &packet->gps_utc_offset);
    swap_short(&packet->week, &packet->week);

    my.time.prev = 0.0;
    my.time.pkt = *packet;
    my.time.got = YES;
    return YES;
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static int process_packet_43()
{
    int size;
    Tans_pkt_43 *packet;

    /*-------------------------------------------------------------
     */
    size = my.packet.cooked_size - sizeof(Tans_pkt_head) - sizeof(Tans_pkt_tail);

    if (size != sizeof(Tans_pkt_43))  return NO;

    packet = (Tans_pkt_43 *)(my.packet.cooked + sizeof(Tans_pkt_head));
```

```
    /*-----------------------------------------------------------------
     */
    swap_float(&packet->sol_time, &packet->sol_time);
    swap_float(&packet->vel.x, &packet->vel.x);
    swap_float(&packet->vel.y, &packet->vel.y);
    swap_float(&packet->vel.z, &packet->vel.z);

    my.vel.pkt = *packet;
    my.vel.got = YES;
    return YES;
}

/*================================================================
==========================
 *-----------------------------------------------------------------*/
static int process_packet_83()
{
    int size;
    Tans_pkt_83 *packet;

    /*-----------------------------------------------------------------
     */
    size = my.packet.cooked_size - sizeof(Tans_pkt_head) - sizeof(Tans_pkt_tail);

    if (size != sizeof(Tans_pkt_83)) return NO;

    packet = (Tans_pkt_83 *)(my.packet.cooked + sizeof(Tans_pkt_head));

    /*-----------------------------------------------------------------
     */
    swap_float(&packet->sol_time, &packet->sol_time);
    swap_double(&packet->pos.x, &packet->pos.x);
    swap_double(&packet->pos.y, &packet->pos.y);
    swap_double(&packet->pos.z, &packet->pos.z);

    my.pos.pkt = *packet;
    my.pos.got = YES;
    return YES;
}
```

```
/*****************************************************************************
 Compute_a matrix calculates the 'A' matrix
 Input arguments are -
        time = time of day in seconds past midnight
 Return arguments are -
        a = 4x4 matrix with the zero elements unused;
        the 1,2,&3 row/column elements are the 'A' transformation
        matrix elements.
 Global variables are -
  epoch_.input

 'compute_a_matrix' is called by 'get_inertial_to_wgs84_matrix'
 'compute_a_matrix' calls 'get_rotation_angles'
*****************************************************************************/

#include <math.h>
#include "rnpm50.h"
#include "iers.h"

        void    compute_a_matrix(time,jd_midnight,IERS,a)
        double  time,jd_midnight,a[4][4] ;
        struct  IERS_DATA *IERS ;
  {
        double  jd, x, y, ut1_minus_utc, sin_x, cos_x, sin_y, cos_y ;

        jd = jd_midnight + time/86400.0 ;
        get_rotation_angles(jd, IERS, &x, &y, &ut1_minus_utc) ;
        sin_x = sin(x) ; cos_x = cos(x) ;
        sin_y = sin(y) ; cos_y = cos(y) ;

        a[1][1] =  cos_x ; a[1][2] = sin_x*sin_y ; a[1][3] = sin_x*cos_y ;
        a[2][1] =    0.0 ; a[2][2] =       cos_y ; a[2][3] =     - sin_y ;
        a[3][1] = -sin_x ; a[3][2] = cos_x*sin_y ; a[3][3] = cos_x*cos_y ;

  }
```

```
/*******  ************************************************************************
This is the function 'bgprop'.
Input arguments are -
        time_total  =  UTC time in secs from start of year
        pos         =       array of position vector in  M50
                    pos[0] = not used
                    pos[1] = x (km)
                    pos[2] = y (km)
                    pos[3] = z (km)
        vel         =       array of velocity vector
                    vel[0] = not used
                    vel[1] = vx (km/s)
                    vel[2] = vy (km/s)
                    vel[3] = vz (km/s)
        delta_t     =       desired time of propogation
        step        =       integrator step size

                    The following inputs are from the daily/weekly
                    US Naval Obsevatory Observations

        solar_flux = daily F10.7 solar flux
        mean_solar_flux = average F10.7 for previous 162 days
        geomagnetic_index_ap  = geomagnetic activity index 6.7 hours prior
                                to measurement
Return arguments are -
        posf        =       array of position vector in M50
                    posf[0] = not used
                    posf[1] = x (km)
                    posf[2] = y (km)
                    posf[3] = z (km)

        velf        =       array of velocity vector
                    velf[0] = not used
                    velf[1] = vx (km/s)
                    velf[2] = vy (km/s)
                    velf[3] = vz (km/s)
It does not return the new time. You must keep track of that from the
calling program.
*************************************************************************/

#include <stdio.h>

#include <orbmech1.h>

#include "rnpm50.h"
#include "iers.h"
#include "util.h"

static struct IERS_DATA far IERS;
```

```
static int first_time = 1;
static int my_year = 1993;

/*=============================================================
=========================
 *-------------------------------------------------------------*/
void bgprop_year(int year) { my_year = year; }

int bgprop_init()
{
   FILE *in_file;
   int   i;

   /*-----------------------------------------------------------
    *  See IERS data file in iers.dat for explaination.
    */
   if (first_time == 0)  return 1;

   if ((in_file = fopen("iers.dat","r")) == NULL)
   {
        puts("ERROR: Can't find IERS.DAT");
        return 0;
   }

   for (i=0; i<=89; i++)
   {
        fscanf(in_file,"%lf %lf %lf %lf\n",
          &IERS.mjd[i],&IERS.x[i],&IERS.y[i],&IERS.t[i]);
   }
   fclose(in_file);

   /*-----------------------------------------------------------
    */
   set_jacchia_data();

   first_time = 0;
   return 1;
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
static void make_rnp(double time_total, double *secs_of_day, EPOCH_REF *epoch, RNP *m50)
{
   double days;          // Total time expressed as days
   double time;          // Seconds in current day

   RNP_REFERENCE reference = AT_MIDNIGHT;

   /*-----------------------------------------------------------
```

```
    */
   time = time_total - (long)(days = time_total/86400.0) * 86400.0;
   *secs_of_day = time;

   epoch->time_scale          = GMT;
   epoch->julian_date         = julian_date(my_year, 1, 1, 0, 0, 0.0) + days;
   epoch->julian_date_midnight = epoch->julian_date - time/86400.0;

   epoch->delta_time_tdt_ut1 =
   epoch->delta_time_tdt_ut1_midnight = 60.9377447714656029;

   compute_rnp(&reference, epoch, m50);
}

/*=============================================================================
============================
 * Note:
 *  Pos_vec and posf_vec may point to the same Vector, i.e. you can use the
 * Vector for input and output.  This also holds true for vel_vec/velf_vec.
 *-----------------------------------------------------------------------*/
void bgprop(double time_total, Vector *pos_vec,
            Vector *vel_vec, Perturbations *P, double delta_t, double step,
            Vector *posf_vec, Vector *velf_vec)
{
   double pos[4], vel[4], posf[4], velf[4];
   double time, timef, rfin;

   EPOCH_REF    epoch;
   RNP          m50;

   /*-----------------------------------------------------------------------
    *  See IERS data file in iers.dat for explaination.
    */
   if (first_time)  bgprop_init();

   // Must now be called by user to initialize Pert. structure with Grav info.
   // instantiate_gotpot(1, P);

   /*-----------------------------------------------------------------------
    */
   pos[1] = pos_vec->x;  pos[2] = pos_vec->y;  pos[3] = pos_vec->z;
   vel[1] = vel_vec->x;  vel[2] = vel_vec->y;  vel[3] = vel_vec->z;

   /*-----------------------------------------------------------------------
    *  Time is: seconds in current day.
    */
   make_rnp(time_total, &time, &epoch, &m50);

   timef = time + delta_t;                    /* time after propagation */
```

```
   cowell(time, pos, vel, &epoch, &m50, P, &IERS, step, timef, posf, velf, &rfin);

   /*-----------------------------------------------------------------
    */
   posf_vec->x = posf[1];  posf_vec->y = posf[2];  posf_vec->z = posf[3];
   velf_vec->x = velf[1];  velf_vec->y = velf[2];  velf_vec->z = velf[3];
}

/*=================================================================
=======================
 *  This function transforms to and from M50 and WGS84 coords depending on the
 * value of 'direction'.  If direction = 0, svin is assumed to be in M50 and
 * svout is the same vector in WGS84.  If direction not 0, the transition is
 * from WGS84 to M50.  ALL TIMES ARE ASSUMED TO BE M50 TYPE.
 *-----------------------------------------------------------------------*/
void m50_wgs84_m50(State_vector *svin, State_vector *svout,
                   int direction, State_vector_transform *svt, int vel_is_ecef)
{
   double posin[4], velin[4], posout[4], velout[4], M50_to_WGS84[4][4];
   double WGS84_to_M50[4][4], time;
   double omega[4], omegaXr[4];
   int i,j;

   EPOCH_REF     epoch;
   RNP           m50;

   /*-----------------------------------------------------------------
    */
   if (first_time)  bgprop_init();

   make_rnp(svin->time, &time, &epoch, &m50);

   get_inertial_to_wgs84_matrix(time, epoch.julian_date_midnight, &m50,
                                EARTH_OMEGA, &IERS, M50_to_WGS84);

   if (svt != NULL)
   {
        svt->time = svin->time;
        *((Vector *)&svt->matrix[0][0]) = *((Vector *)&M50_to_WGS84[1][1]);
        *((Vector *)&svt->matrix[1][0]) = *((Vector *)&M50_to_WGS84[2][1]);
        *((Vector *)&svt->matrix[2][0]) = *((Vector *)&M50_to_WGS84[3][1]);
   }

   /*-----------------------------------------------------------------
    */
   posin[1] = svin->pos.x;
   posin[2] = svin->pos.y;
   posin[3] = svin->pos.z;
   velin[1] = svin->vel.x;
   velin[2] = svin->vel.y;
```

```
    velin[3] = svin->vel.z;

    if (vel_is_ecef)
    {
        omega[1] = EARTH_OMEGA * M50_to_WGS84[3][1];
        omega[2] = EARTH_OMEGA * M50_to_WGS84[3][2];
        omega[3] = EARTH_OMEGA * M50_to_WGS84[3][3];
    }

    /*---------------------------------------------------------------
     */
    if (direction == 0)             // M50 to WGS84
    {
        if (vel_is_ecef)
        {
            cross(omega, posin, omegaXr);
            velin[1] -= omegaXr[1];  velin[2] -= omegaXr[2];  velin[3] -= omegaXr[3];
        }
        matrix_x_vector(M50_to_WGS84, posin, posout);
        matrix_x_vector(M50_to_WGS84, velin, velout);
    }
    else                    // WGS84 to M50
    {
        matrixTranspose_x_vector(M50_to_WGS84, posin, posout);
        matrixTranspose_x_vector(M50_to_WGS84, velin, velout);
        if (vel_is_ecef)
        {
            cross(omega, posout, omegaXr);
            velout[1] += omegaXr[1];  velout[2] += omegaXr[2];  velout[3] += omegaXr[3];
        }
    }

    svout->pos.x = posout[1];
    svout->pos.y = posout[2];
    svout->pos.z = posout[3];
    svout->vel.x = velout[1];
    svout->vel.y = velout[2];
    svout->vel.z = velout[3];
    svout->time = svin->time;
}
```

```
/*****************************************************************************
 This is the function 'cowell'.
 Input arguments are -
        time = initial time in seconds
        pos     =       array of initial position vector
                pos[0] = not used
                pos[1] = x (km)
                pos[2] = y (km)
                pos[3] = z (km)
        vel     =       array of initial velocity vector
                vel[0] = not used
                vel[1] = vx (km/s)
                vel[2] = vy (km/s)
                vel[3] = vz (km/s)
        step = integrator step size in seconds
        timef= target time of propogation
        epoch   = Pointer to EPOCH_REF
        m50     = Pointer to M50 RNP Matrix
        P       = Pointer to Perturbations Structure
        IERS    = Pointer to IERS data file
 Return arguments are -
        posf    =       array of propogated position vector
                posf[0] = not used
                posf[1] = x (km)
                posf[2] = y (km)
                posf[3] = z (km)
        velf    =       array of propogated velocity vector
                velf[0] = not used
                velf[1] = vx (km/s)
                velf[2] = vy (km/s)
                velf[3] = vz (km/s)
        rfin = magnitude of position vector at destination

 'Cowell' is called by 'bgprop'.
 'Cowell' calls the Runge-Kutta integrator.
*****************************************************************************/
#include <stdio.h>

#include <orbmech1.h>
#include "iers.h"
#include "util.h"
#include "rnpm50.h"

void cowell(double time, double pos[4], double vel[4], EPOCH_REF *epoch,
            RNP *m50, Perturbations *P, struct IERS_DATA *IERS, double step,
            double timef, double posf[4], double velf[4], double rfin[1])
{
   double       Initial_State[8],Integrated_State[8],time_step,time_now  ;
```

```
/*****************************************************************************/
This is the function 'cowell'.
Input arguments are -
        time = initial time in seconds
        pos     =       array of initial position vector
                pos[0] = not used
                pos[1] = x (km)
                pos[2] = y (km)
                pos[3] = z (km)
        vel     =       array of initial velocity vector
                vel[0] = not used
                vel[1] = vx (km/s)
                vel[2] = vy (km/s)
                vel[3] = vz (km/s)
        step = integrator step size in seconds
        timef= target time of propogation
        epoch   = Pointer to EPOCH_REF
        m50     = Pointer to M50 RNP Matrix
        P       = Pointer to Perturbations Structure
        IERS    = Pointer to IERS data file
Return arguments are -
        posf    =       array of propogated position vector
                posf[0] = not used
                posf[1] = x (km)
                posf[2] = y (km)
                posf[3] = z (km)
        velf    =       array of propogated velocity vector
                velf[0] = not used
                velf[1] = vx (km/s)
                velf[2] = vy (km/s)
                velf[3] = vz (km/s)
        rfin = magnitude of position vector at destination

 'Cowell' is called by 'bgprop'.
 'Cowell' calls the Runge-Kutta integrator.
/*****************************************************************************/
#include <stdio.h>

#include <orbmech1.h>
#include "iers.h"
#include "util.h"
#include "rnpm50.h"

void cowell(double time, double pos[4], double vel[4], EPOCH_REF *epoch,
            RNP *m50, Perturbations *P, struct IERS_DATA *IERS, double step,
            double timef, double posf[4], double velf[4], double rfin[1])
{
   double       Initial_State[8],Integrated_State[8],time_step,time_now ;
```

```
    int   done = FALSE ;

    Initial_State[1] = pos[1] ;
    Initial_State[2] = pos[2] ;
    Initial_State[3] = pos[3] ;
    Initial_State[4] = vel[1] ;
    Initial_State[5] = vel[2] ;
    Initial_State[6] = vel[3] ;
    Initial_State[7] = magnitude(pos) ;

    time_now  = time ;
    time_step = step ;

    while(done == FALSE)
    {
        if((time_now+time_step) > timef)
        {
           time_step = timef - time_now ;
           done = TRUE ;
        }

        Runge_Kutta_4(time_now, Initial_State, epoch->julian_date_midnight,
          m50, P, IERS, time_step, Integrated_State);

        time_now = time_now + time_step;       /*update time */

        Initial_State[1] = Integrated_State[1];      /*update state */
        Initial_State[2] = Integrated_State[2];
        Initial_State[3] = Integrated_State[3];
        Initial_State[4] = Integrated_State[4];
        Initial_State[5] = Integrated_State[5];
        Initial_State[6] = Integrated_State[6];
        Initial_State[7] = Integrated_State[7];
    }

    posf[1] = Initial_State[1];                  /*final state */
    posf[2] = Initial_State[2];
    posf[3] = Initial_State[3];

    velf[1] = Initial_State[4];
    velf[2] = Initial_State[5];
    velf[3] = Initial_State[6];

    rfin[0] = Initial_State[7];
}
```

```
/*****************************************************************************
 This is the Derivatives function. It is called by the integrator and
 defines the first derivatives of the state vector.
 Input arguments are -
        time = state vector time in seconds
        xz[8] = state array consisting of the following elements -
                xz[0] = not used
                xz[1] = position in x (km)
                xz[2] = position in y (km)
                xz[3] = position in z (km)
                xz[4] = velocity in x (km/s)
                xz[5] = velocity in y (km/s)
                xz[6] = velocity in z (km/s)
                xz[7] = magnitude of position vector (km)
 Return arguments are-
        f[8] =  derivatives of input state vector elements

 'Derivatives' is called by Runge-Kutta
 'Derivatives' calls -
                get_inertial_to_wgs84_matrix
                matrix_x_vector
                gotpot
                matrixTranspose_x_vector
                jacchia_density
                magnitude
*****************************************************************************/
#include <stdio.h>

#include <orbmech1.h>
#include "iers.h"
#include "rnpm50.h"
#include "util.h"

        void    Derivatives(m50, P, IERS, jd_midnight, time, xz, f)
        double  time,xz[8],f[8],jd_midnight ;
        Perturbations *P ;
        struct  IERS_DATA *IERS ;
        RNP     *m50 ;
  {
        double  jacchia_density();
        double  density, velocity_mag, relative_velocity[4], temp ;
        double  position_inertial[4], velocity_inertial[4], position_wgs84[4] ;
        double  g[4], g_wgs84[4], pot, drag_acceleration[4] ;
        double  M50_to_WGS84[4][4];                     /* tranformation matrix */
        double  jd;                                     /* julian date */

        int     i;                                       /* loop counter */
        int     total_gravity = 1;
```

```
  for(i=1 ; i<=3 ; i++)
   {
     position_inertial[i]=xz[i] ;
     velocity_inertial[i]=xz[i+3] ;
     drag_acceleration[i]=0.0 ;
     g[i]=0.0 ;
   pot=0.0 ;
   }

  /* ------------------------//
  // Get Gravity Acceleration //
  // ------------------------*/

/*
 *  Rotate Position from M50 to WGS84
 *  Compute Gravity Perturbations
 *  Rotate gravity vector back to M50 for use in computing
 *  derivatives.
 */
  get_inertial_to_wgs84_matrix(time,jd_midnight,m50,
     P->Gravity.GMD.planet_omega,IERS,M50_to_WGS84);
  matrix_x_vector(M50_to_WGS84, position_inertial, position_wgs84) ;
  gotpot(P, total_gravity, position_wgs84, xz[7], g_wgs84, &pot) ;
  matrixTranspose_x_vector(M50_to_WGS84, g_wgs84, g) ;

  /*----------------------//
  // Get Drag Acceleration //
  //----------------------*/

if (P->Drag.mode == ON) {
    jd = jd_midnight+time/86400;

    density = jacchia_density(position_inertial,        /* (kg/m^3) */
                  position_wgs84, xz[7], jd,
                  P->Drag.solar_flux,P->Drag.mean_solar_flux,
                  P->Drag.geomagnetic_index_ap)  * 1000.0;

    relative_velocity[1] = velocity_inertial[1] +
                           position_inertial[2] *
                           P->Gravity.GMD.planet_omega ;
    relative_velocity[2] = velocity_inertial[2] -
                           position_inertial[1] *
                           P->Gravity.GMD.planet_omega ;
    relative_velocity[3] = velocity_inertial[3] ;

    velocity_mag = magnitude(relative_velocity) ;
  temp = -0.5*density*velocity_mag*1000.0 / P->Drag.ballistic_number ;
    for(i=1 ; i<=3 ; i++)
     {
       drag_acceleration[i] = temp * relative_velocity[i] ; /* (km/s^2) */
```

```
        }
    }

    /*--------------------//
    // Compute Derivatives //
    //--------------------*/

    for(i=1 ; i<=3 ; i++)
    {
       f[i] = xz[i+3] ;
       f[i+3] = g[i] + drag_acceleration[i];
    }
    f[7] = (xz[1]*xz[4] + xz[2]*xz[5] + xz[3]*xz[6])/xz[7] ;
}
```

```
/*=============================================================
============================
-- ++
-- COMPONENT NAME:
--   GEM_9_model_data
--
-- ABSTRACT:
--   This package provides the GEM 9 normalized coefficients which are
--   taken from the reference. This package also provides values of the
--   earth's gravitational constant, GM, earth's equatorial radius, ae,
--   and the earth's rotation rate, omega.
--
-- AUTHOR:
--  Robert G. Gottlieb
--
-- CREATION DATE:
--   25 January 1989
--   C version created 25 January 1990
--
-- PROJECT/PROGRAM:
--   McDonnell Douglas Space Systems Company
--   Engineering Services
--
-- REFERENCES:
--  These constants are given in Table 3  of "Gravity Model Improvements
--  Using Geos 3 (GEM 9 and GEM 10)" by F. Lerch, S. Klosko, R. Laubscher,
--  and C. Wagner, Journal of Geophysical Research, Vol. 84, N0. B8,
--  July 30, 1979.
--
--
-- KEYWORDS:
--   gravity model
--   gravitational potential
--   gravitational acceleration
--   central force
--   gravitational perturbations
--   nonspherical earth
--
-- IMPLEMENTATION DEPENDENCIES:
--   none
-- ++
 *-------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include "util.h"

/*=============================================================
============================
 *-------------------------------------------------------------------*/
static double
```

```
far c1[1]   = { 0.0 },

far c2[3]   = {-484.16555,-0.00021, 2.43400 },

far c3[4]   = { 0.95848, 2.02826, 0.89198, 0.70256 },

far c4[5]   = { 0.54154,-0.53287, 0.35259, 0.98849,-0.19661 },

far c5[6]   = { 0.06844,-0.04910, 0.64975,-0.47030,-0.29049, 0.16123},

far c6[7]   = {-0.15121,-0.07543, 0.04813, 0.05910,-0.10458,-0.25597,
                0.00188},

far c7[8]   = { 0.09331, 0.26724, 0.32786, 0.23987,-0.28553, 0.01667,
               -0.36330,-0.00924},

far c8[9]   = { 0.05095, 0.02483, 0.07315,-0.00992,-0.24624,-0.01251,
               -0.07471, 0.06778,-0.12123},

far c9[10]  = { 0.02733, 0.16065, 0.02445,-0.17360,-0.01134, 0.00490,
                0.03737,-0.10449, 0.19418,-0.05810},

far c10[11] = { 0.05284, 0.08238,-0.08429,-0.02071,-0.10099,-0.06216,
               -0.03888,-0.00022, 0.03989, 0.12430, 0.10563},

far c11[12] = {-0.04862, 0.00023, 0.03947,-0.02883,-0.04503, 0.04545,
                0.00256, 0.01076, 0.00878,-0.02893,-0.06709, 0.04047},

far c12[13] = { 0.03794,-0.06304, 0.00390, 0.06412,-0.07375, 0.04569,
                0.00273,-0.01282,-0.02430, 0.04154, 0.00028, 0.01603,
               -0.00764},

far c13[14] = { 0.04408,-0.03419, 0.02214,-0.04313,-0.02511, 0.06892,
               -0.03329,-0.00490,-0.02066, 0.02678, 0.03154,-0.04459,
               -0.03230,-0.06009},

far c14[15] = {-0.02111,-0.01764,-0.03632, 0.01271,-0.01831, 0.01940,
               -0.00684, 0.01804,-0.04572, 0.03880, 0.05791, 0.01920,
                0.00835, 0.02803,-0.05119},

far c15[16] = { 0.00136,-0.01848, 0.00864, 0.02086,-0.04391,-0.01243,
                0.02539, 0.06506,-0.01726, 0.01303, 0.02896, 0.00047,
               -0.03347,-0.02282, 0.00392,-0.02108},

far c16[17] = {-0.00846, 0.02543,-0.00524,-0.00379, 0.03351,-0.01371,
               -0.00697, 0.00412,-0.01005,-0.01885, 0.01564, 0.02186,
                0.01770, 0.01223,-0.01943,-0.01320,-0.02546},

far c17[18] = { 0.01619,-0.00353,-0.00985,-0.00821, 0.00095,-0.02428,
               -0.00165, 0.01583, 0.01245,-0.01075, 0.01069,-0.04212,
```

```
                0.03148, 0.01471,-0.01589, 0.00159,-0.02368,-0.01299},

far c18[19] = { 0.01117, 0.00227, 0.00649,-0.00006, 0.02206, 0.01637,
                0.01215,-0.00308, 0.00676,-0.02487,-0.00246,-0.00447,
               -0.03891,-0.01200,-0.00851,-0.05461, 0.00824, 0.01983,
                0.00198},

far c19[20] = { 0.00133,-0.03465, 0.00877, 0.00058,-0.00396,-0.00436,
                0.00006,-0.00866, 0.00813, 0.00493,-0.01507,-0.00497,
               -0.00752,-0.01235,-0.00552,-0.02174,-0.03536, 0.02971,
                0.03539, 0.00865},

far c20[21] = { 0.02562, 0.00129,-0.00395,-0.01561, 0.00072, 0.00153,
               -0.00620,-0.01072,-0.00306, 0.00534,-0.01150, 0.01075,
               -0.00911, 0.02319, 0.01232,-0.02311,-0.01434,-0.01435,
               -0.00275, 0.01586,-0.00510},

far c21[22] = {-0.00128,-0.01238, 0.01467, 0.00407, 0.00313, 0.0,
                0.0,     0.0,     0.0,     0.00467,-0.00048,-0.00506,
                0.00330,-0.01641, 0.01955, 0.01066, 0.00027,-0.00288,
                0.01417, 0.01202},

far c22[23] = {-0.00448, 0.00094,-0.00147,-0.00149,-0.00543, 0.0,
                0.0,     0.0,     0.0,     0.00507, 0.00216,-0.00501,
               -0.01577,-0.03006, 0.00954, 0.02425,-0.00372,-0.00225,
                0.01045,-0.01636},

far c23[24] = {-0.01837, 0.00248, 0.0,     0.0,     0.0,     0.0,
                0.0,     0.0,     0.0,     0.0,     0.0,     0.0,
                0.02040,-0.00217, 0.00881, 0.01267},

far c24[25] = {-0.00190,-0.00567, 0.0,     0.0,     0.0,     0.0,
                0.0,     0.0,     0.0,     0.0,     0.0,     0.0,
               -0.00257, 0.00610,-0.0165,  0.00487},

far c25[26] = {-0.00016, 0.01175, 0.0,     0.0,     0.0,     0.0,
                0.0,     0.0,     0.0,     0.0,     0.0,     0.0,
               -0.00189, 0.01522,-0.02361,-0.00289},

far c26[27] = { 0.00404, 0.0,     0.0,     0.0,     0.0,     0.0,
                0.0,     0.0,     0.0,     0.0,     0.0,     0.0,
                0.0,    -0.00323, 0.00688},

far c27[28] = { 0.01022, 0.0,     0.0,     0.0,     0.0,     0.0,
                0.0,     0.0,     0.0,     0.0,     0.0,     0.0,
                0.0,    -0.00770, 0.02350, 0.0,     0.0,     0.0,
                0.0,     0.0,     0.0,     0.0,     0.0,     0.0,
                0.0,     0.0,    -0.01133,-0.00100},

far c28[29] = {-0.02063, 0.0,     0.0,     0.0,     0.0,     0.0,
```

```
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.02064,-0.00531, 0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.00754,-0.00861, 0.00487},

far c29[30] = {-0.00915, 0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,   -0.01083,-0.00974, 0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,   -0.01043},

far c30[31] = { 0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,   -0.01820},

*cc[31] =
{
     c1,  c1,  c2,  c3,  c4,  c5,  c6,  c7,  c8,  c9, c10, c11, c12, c13,
    c14, c15, c16, c17, c18, c19, c20, c21, c22, c23, c24, c25, c26, c27,
    c28, c29, c30
};


/*=============================================================================
==========================
 *-----------------------------------------------------------------------*/
static double
   far s1[1]   = { 0.0 },

   far s2[3]   = { 0.0,   -0.00406,-1.39786 },

   far s3[4]   = { 0.0,    0.25244,-0.62241, 1.41140},

   far s4[5]   = { 0.0,   -0.46512, 0.66272,-0.20316, 0.29894},

   far s5[6]   = { 0.0,   -0.09225,-0.32488,-0.20788, 0.05037,-0.66181},

   far s6[7]   = { 0.0,    0.01587,-0.34854, 0.00138,-0.46086,-0.53881,
                  -0.24196},

   far s7[8]   = { 0.0,    0.09607, 0.10505,-0.21778,-0.12689, 0.04367,
                   0.13170, 0.02663},

   far s8[9]   = { 0.0,    0.06327, 0.05121,-0.08830, 0.07539, 0.08245,
                   0.32546, 0.07712, 0.12712},

   far s9[10]  = { 0.0,    0.01676,-0.02467,-0.09715, 0.01072,-0.06410,
                   0.21837,-0.07411,-0.01555, 0.09172},
```

```
far s10[11] = { 0.0,    -0.13553,-0.00375,-0.16867,-0.06890,-0.03550,
                -0.08597, 0.01616,-0.07159,-0.05053,-0.01962},

far s11[12] = { 0.0,    -0.01356,-0.09952,-0.13577,-0.06082, 0.07889,
                 0.03820,-0.08216, 0.02269, 0.03349,-0.00324,-0.07862},

far s12[13] = { 0.0,    -0.05415,-0.00931, 0.03136,-0.01663, 0.01287,
                 0.03705, 0.04908, 0.02380, 0.01083, 0.05133,-0.00483,
                -0.01178},

far s13[14] = { 0.0,     0.03644,-0.03804, 0.07611, 0.00559, 0.04607,
                 0.00278,-0.00121,-0.00321, 0.04194,-0.03227,-0.01686,
                 0.09061, 0.06977},

far s14[15] = { 0.0,     0.04976, 0.04590,-0.01504, 0.00804,-0.02184,
                -0.00702,-0.03207,-0.00017, 0.01116, 0.01384,-0.03660,
                -0.03024, 0.04223,-0.00591},

far s15[16] = { 0.0,    -0.00236,-0.02976, 0.00562,-0.00720,-0.00533,
                -0.04249, 0.01133, 0.02563, 0.04074, 0.01124,-0.00421,
                 0.01659,-0.00224,-0.02460,-0.00640},

far s16[17] = { 0.0,    -0.01232, 0.01566,-0.02645, 0.01158,-0.00269,
                -0.02266,-0.00553, 0.00567,-0.03781, 0.00788, 0.00936,
                 0.00977,-0.00750,-0.03787,-0.02607, 0.01402},

far s17[18] = { 0.0,    -0.02223, 0.01244, 0.00715,-0.00601, 0.00761,
                -0.00974, 0.01255, 0.00300,-0.01928, 0.01173,-0.00404,
                 0.02126, 0.01921, 0.01083, 0.00393, 0.01308,-0.00090},

far s18[19] = { 0.0,    -0.00242, 0.02368,-0.00023, 0.01095, 0.01262,
                -0.00964,-0.00071,-0.01226, 0.02529,-0.01030, 0.01734,
                -0.02305,-0.03737,-0.00934,-0.01615, 0.01840, 0.00931,
                -0.01267},

far s19[20] = { 0.0,    -0.00698,-0.00714, 0.00094,-0.00992,-0.00110,
                 0.00323, 0.00852,-0.01796, 0.01924,-0.00637, 0.02493,
                 0.00613,-0.03066,-0.01273,-0.01616,-0.01328,-0.00531,
                -0.02120,-0.00143},

far s20[21] = { 0.0,    -0.02215, 0.00673, 0.01130,-0.00797, 0.00204,
                -0.00024,-0.00353,-0.00465, 0.02218,-0.00746,-0.00997,
                 0.02576, 0.00394,-0.01019, 0.00947,-0.00130,-0.00674,
                 0.02895, 0.00864,-0.00508},

far s21[22] = { 0.0,     0.00749, 0.00653, 0.00381,-0.00155, 0.0,
                 0.0,     0.0,     0.0,    -0.00175, 0.00268,-0.01019,
                 0.02249, 0.01387, 0.01031, 0.01209,-0.00542,-0.00243,
                -0.01220,-0.00849},
```

```
far s22[23] = { 0.0,    0.00554,-0.00964, 0.01117, 0.00422, 0.0,
                0.0,    0.0,    0.0,    -0.00136, 0.00094,-0.02392,
               -0.01916, 0.00774, 0.00665, 0.00033,-0.00436,-0.00325,
               -0.01016,-0.02256},

far s23[24] = { 0.0,    0.00907, 0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.01049,-0.00165,-0.00568, 0.00258},

far s24[25] = { 0.0,   -0.00119, 0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
               -0.01462,-0.00728, 0.00317,-0.00157},

far s25[26] = { 0.0,   -0.00442, 0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.00732,-0.00880, 0.01669,-0.00148},

far s26[27] = { 0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,-0.00820, 0.00148},

far s27[28] = { 0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,   -0.01073, 0.00612, 0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,   -0.00526,-0.00332},

far s28[29] = { 0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.00875,-0.01147, 0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,   -0.00232, 0.00306, 0.00612},

far s29[30] = { 0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,   -0.00896, 0.00956, 0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,   -0.01602},

far s30[31] = { 0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,    0.0,    0.0,
                0.0,    0.0,    0.0,    0.0,   -0.03494},

*ss[31] =
{
     s1, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12,
    s13, s14, s15, s16, s17, s18, s19, s20, s21, s22, s23, s24, s25,
    s26, s27, s28, s29, s30
```

```
   };

/*=========================================================================
=========================
 *-------------------------------------------------------------*/
void gem_9_model_data(double *planet_mu, double *planet_radius,
                      double *planet_omega, double *planet_flattening,
                      double *planet_j2, double *c, double *s)
{
   /*-------------------------------------------------------------
    */
   int n,m,j_coef;
   double coef,facr;

   /*-------------------------------------------------------------
    */
   for(n = 2;n < 31; n++)
   {
        for(m = 0; m <= n; m++)
        {
           j_coef = (n*n + n -4)/2  + m;
           if(m == 0)
           {
                c[j_coef] = sqrt( (double)(2*n + 1) )* cc[n][0]*1.0e-6;
           }
           else
           {
                facr = factorial_ratio((n-m),(n+m));
                coef = sqrt( (double)(4*n + 2) *facr)*1.0e-6;
                c[j_coef] = coef * cc[n][m];
                s[j_coef] = coef * ss[n][m];
           }
        }
   }

   *planet_mu         = 398600.64;       /* kilometers**3/sec**2 */
   *planet_radius     = 6378.139;        /* kilometers */
   *planet_omega      = 7.29211514646e-5; /* radians/second */
   *planet_flattening = 1.0/298.257;     /* no units */
   *planet_j2         = -c[1];           /* no units */
}
```

```
/*****************************************************************************
This is 'get_rotation_angles'
 Input arguments are -
        jd =    julian date
 Return arguments are -
         x
        y
        utl_minus_utc

 'get_rotation_angles' is called by 'compute_a_matrix'
 Requires the IERS.dat file containing US Naval Observatory predictions
*****************************************************************************/
#include <const.h>
#include "iers.h"

void get_rotation_angles(double jd, struct IERS_DATA *IERS,
                         double *x, double *y, double *utl_minus_utc)
{
   int i, iml;
   double modified_julian_date, factor;

   modified_julian_date = jd - 2400000.5;

   if ((modified_julian_date < IERS->mjd[0]) ||
        (modified_julian_date > IERS->mjd[89]))
   {
        *x = 0.0;
        *y = 0.0;
        *utl_minus_utc = 0.0;
        printf("IERS error: I need %6.0f, file has %6.0f to %6.0f\n",
          modified_julian_date, IERS->mjd[0], IERS->mjd[89]);
   }
   else
   {
        for (i=0; i<=89; i++) { if (IERS->mjd[i] > modified_julian_date) break;}
        iml = i-1;

        factor = (modified_julian_date - IERS->mjd[iml]) / (IERS->mjd[i] - IERS->mjd[iml]);

        *x = IERS->x[iml] + (IERS->x[i] - IERS->x[iml]) * factor;
        *y = IERS->y[iml] + (IERS->y[i] - IERS->y[iml]) * factor;
        *utl_minus_utc = IERS->t[iml] + (IERS->t[i] - IERS->t[iml]) * factor;

        *x *= DEGREES / 3600.0;
        *y *= DEGREES / 3600.0;
   }
}
```

```
/*==========================================================================
==========================
 *-------------------------------------------------------------------*/
#include <orbmech1.h>

void gotpot(Perturbations *P, int total_gravity, double position[4],
            double position_mag, double g[4], double *pot)
{
   double ri, xovr, yovr, zovr, ep;
   double muor, muor2, reor, reorn, sumv;
   double pn[38], pnm1[38], pnm2[38], ctil[38], stil[38];
   double sumh, sumgam, sumj, sumk, sumh_n, np1, lambda;
   double sumgam_n, sumj_n, sumk_n, mxpnm, twonm1, npmp1;
   double sumv_n, bnmtil, pnmbnm, anmtm1, bnmtm1;

   int   mm1, mp1, nm1, nm;
   int   j, k, L, lim, m, n;

   /*-----------------------------------------------------------------
    */
   ri = 1.0/position_mag;
   xovr = position[1]*ri;
   yovr = position[2]*ri;
   zovr = position[3]*ri;
   ep = zovr;
   reor = P->Gravity.GMD.planet_radius * ri;
   reorn = reor;
   muor = P->Gravity.GMD.planet_mu*ri;
   muor2 = muor*ri;

   k = 0;
   L = P->Gravity.number_tesseral + 2;

   for (m = 1; m <= L; m++)
   {
        pnm2[m] = 0.0;
        pnm1[m] = 0.0;
   }

   pnm2[0] =  1.0;
   pnm1[0] =  ep;
   pnm1[1] =  1.0;
   ctil[0] =  1.0;
   stil[0] =  0.0;
   ctil[1] = xovr;
   stil[1] = yovr;
   sumv   = (double)total_gravity;
   sumh=0.0;
   sumj=0.0;
   sumk=0.0;
```

```
sumgam = (double)total_gravity;

/*-------------------------------------------------------------------
 */
for (n = 2; n <= P->Gravity.number_zonal; n++)
{                                                    /*Sum over zonals */
     reorn = reorn*reor;
     twonm1 = (double)(2*n-1);

     nm1 = n-1;
     np1 = (double)(n + 1);
     L   = 1;

     /*-----------------------------------------------------//
     // RECURSIVE COMPUTATION OF LEGENDRE POLYNOMIAL - pn[0] //
     // and first associated legendre function pn[1]         //
     //-----------------------------------------------------*/

     pn[0] = (twonm1*ep*pnm1[0] - (double)(nm1)*pnm2[0])/(double)(n);
     pn[1] = pnm2[1] + twonm1*pnm1[0];
     pn[2] = pnm2[2] + twonm1*pnm1[1];

     k = k + 1;
     sumv_n = pn[0]*P->Gravity.GMD.cc[k];
     sumh_n = pn[1]*P->Gravity.GMD.cc[k];
     sumgam_n = pn[0]*P->Gravity.GMD.cc[k]*np1;

     if(P->Gravity.number_tesseral > 0)
     {                          /* ignore tesserals if number_tesseral = 0 */
        sumj_n = 0.0;
        sumk_n = 0.0;

        ctil[n] = ctil[1]*ctil[nm1] - stil[1]*stil[nm1];
        stil[n] = stil[1]*ctil[nm1] + ctil[1]*stil[nm1];

        if (n < P->Gravity.number_tesseral)
             lim = n;
        else
             lim = P->Gravity.number_tesseral;

        for(m =1; m <= lim; m++)
        {                                            /*Sum over tesserals */
             mm1 = m-1;
             mp1 = m + 1;
             npmp1 = (double)(n + mp1);

             /*-------------------------------------------//
             // RECURSIVE COMPUTATION OF ASSOCIATED LEGENDRE //
             // FUNCTIONS - pn[m + 1]                       //
             //-------------------------------------------*/
```

```
            pn[mp1] = pnm2[mp1] + twonm1*pnm1[m];
            mxpnm =(double)(m)*pn[m];
            nm = k + m;
            bnmtil = P->Gravity.GMD.cc[nm]*ctil[m  ] + P->Gravity.GMD.ss[nm]*stil[m  ];
            pnmbnm = pn[m]*bnmtil;
            sumv_n = sumv_n + pnmbnm;
            bnmtm1 = P->Gravity.GMD.cc[nm]*ctil[mm1] + P->Gravity.GMD.ss[nm]*stil[mm1];
            anmtm1 = P->Gravity.GMD.cc[nm]*stil[mm1] - P->Gravity.GMD.ss[nm]*ctil[mm1];
            sumh_n = sumh_n + pn[mp1]*bnmtil;
            sumgam_n = sumgam_n + npmp1*pnmbnm;
            sumj_n = sumj_n + mxpnm*bnmtm1;
            sumk_n = sumk_n-mxpnm*anmtm1;
         }                              /* end of sum over tesserals */

         sumj = sumj + reorn*sumj_n;
         sumk = sumk + reorn*sumk_n;

      }

      /*---------------------------------------//
      // SUMS BELOW HERE HAVE VALUES WHEN M = 0 //
      //---------------------------------------*/

      sumv = sumv + reorn*sumv_n;
      sumh = sumh + reorn*sumh_n;
      sumgam = sumgam + reorn*sumgam_n;

      k = k + n;

      /*---------------------------------//
      // SHIFT LEGENDRE POLYNOMIALS DOWN //
      //---------------------------------*/

      if (n < P->Gravity.number_zonal)
      {
         L = n;
         for (j = 0; j <= L; j++)
         {
              pnm2[j] = pnm1[j];
              pnm1[j] = pn[j];
         }
      }
}                                     /*end of sum over zonals */

*pot   = muor*sumv;
lambda = sumgam  + ep*sumh;
g[1]  = -muor2*(lambda*xovr - sumj);
g[2]  = -muor2*(lambda*yovr - sumk);
g[3]  = -muor2*(lambda*zovr - sumh);
```

```
}

/*=========================================================================
=========================
 *-------------------------------------------------------------------*/
void get_gravity_data(Perturbations *P, double *mu, double *j2,
                      double *planet_radius, double *planet_rate,
                      double *flattening)
{
   *mu            = P->Gravity.GMD.planet_mu;
   *j2            = P->Gravity.GMD.planet_j2;
   *planet_radius = P->Gravity.GMD.planet_radius;
   *planet_rate   = P->Gravity.GMD.planet_omega;
   *flattening    = P->Gravity.GMD.planet_flattening;
}

/*=========================================================================
=========================
 *-------------------------------------------------------------------*/
void instantiate_gotpot(int model_name, Perturbations *P)
{
   switch (model_name)
   {
        case 1:
           gem_9_model_data(&P->Gravity.GMD.planet_mu,
                            &P->Gravity.GMD.planet_radius,
                            &P->Gravity.GMD.planet_omega,
                            &P->Gravity.GMD.planet_flattening,
                            &P->Gravity.GMD.planet_j2,
                            &P->Gravity.GMD.cc,
                            &P->Gravity.GMD.ss);
           break;
        case 2:
           break;
   }
}
```

```
#ifndef IERS_INCLUDED
#define IERS_INCLUDED

struct IERS_DATA {
  double mjd[90], x[90], y[90], t[90] ;
} ;

#endif
```

```
/*****************************************************************************
 This is the 'get_inertial_to_wgs84_matrix'
 Input arguments are -
        time = time of day in seconds past midnight
 Return arguments are -
        M50_to_WGS84 = 4x4 matrix with the zero elements unused;
                       the 1,2,&3 row/column elements are the M50_to_WGS84
                       transformation matrix elements.
 'get_inertial_to_wgs84_matrix' is called by 'Derivatives'
 'get_inertial_to_wgs84_matrix' calls -
                          'compute_a_matrix'
                          'rotation_generator'
                          'matrix_x_matrix'
*****************************************************************************/
#include <orbmech1.h>
#include "rnpm50.h"
#include "iers.h"

void get_inertial_to_wgs84_matrix(double time, double jd_midnight, RNP *m50,
                                  double planet_omega,
                                  struct IERS_DATA *IERS,
                                  double M50_to_WGS84[4][4])
{
   double time_since_epoch, G[4][4], M50_to_EF[4][4], A[4][4];

   time_since_epoch = time;  /* time in seconds since midnight UTC */
   compute_a_matrix(time,jd_midnight,IERS,A);
   rotation_generator(time_since_epoch * planet_omega, G);

   matrix_x_matrix(G,m50->rnp,M50_to_EF);
   matrix_x_matrix(A,M50_to_EF,M50_to_WGS84);
}
```

```
#include <math.h>
#include <const.h>
#include "util.h"

#define earth_flattening_factor          (1.0 / 298.257)
#define polar_equatorial_ratio_squared   pow(1.0 - earth_flattening_factor,2.0)
#define days_in_tropical_year            365.2422
#define planet_oblate_coefficient        ((earth_flattening_factor * (2.0 - earth_flattening_factor)) / pow(1.0
                                              - earth_flattening_factor,2.0))
#define beta      (-37.0 * DEGREES)      /* determines lag of the maximum exospheric temperature */
#define p         ( 6.0 * DEGREES)       /* introduces in the temperature curve an asymmetry */
#define gamma     ( 43.0 * DEGREES)      /* determines the location of the of the asymmetry */
#define r         0.31                   /* fixes the relative amplitude of the temperature variation */

        void    equatorial_solar_ephemeris ();
/*****************************************************************************
This is the call to 'set_data'. It creates 'density_table'.
Input arguments are -
        void
Return arguments are -
        density_table matrix defined in 'jacchia_data

This segment calls 'jacchia_data' in jacdat.c
*****************************************************************************/
        static double density_table[68][31];

        void set_jacchia_data ()
  {
        jacchia_data(density_table);
  }

/*****************************************************************************
This is the function 'jacchia_density'
Input arguments are -
        inertial_position       =       a vector
        earth_fixed_position    =       a vector
        radius                  =       magnitude of position vector
        at_time                 =       julian date
        solar_flux              =       F10.7 observed at time of measurement
        mean_solar_flux         =       mean F10.7 for previous 162 day period
        geomagnetic_index_ap    =       geomagnetic activity index 6.7 hours prior
                                        to measurement
Return arguments are -
        (double) value of atmospheric density in grams per cm^3

'jacchia_density' is called by 'Derivatives'
'jacchia_density' calls -       'equitorial_solar_ephemeris
                                'jdtocd' in util.c
                                'julian_date' in util.c
*****************************************************************************/
```

```
double jacchia_density(inertial_position,earth_fixed_position,
                radius, at_time, solar_flux,mean_solar_flux,
                geomagnetic_index_ap)
double inertial_position[], earth_fixed_position[], radius, at_time ;
double solar_flux,mean_solar_flux,geomagnetic_index_ap;
{

int year, month, day, hour, minute ;
int table_index_1 ;          /* for density_table */
int table_index_2 = 2;       /* for density_table */
int previous_table_index_1 ; /* for density_table */
int previous_table_index_2 ; /* for density_table */

double altitude_model(), altitude, log_density, dht ;
double sin_theta, day_ratio, seconds, tau, last_day ;
/*  double mean_solar_flux ; /* 10**-22 watts/m**2/hz */
/*  double solar_flux ;  /* 10**-22 watts/m**2/hz */

double exospheric_minimum_temperature ;        /* kelvin */
double geomagnetic_adjustment ;                /* kelvin */
double semi_annual_variation ;                 /* kelvin */

double sun_vector[4] ;
double sun_declination ;                       /* radians */
double sin_sun_right_ascension, cos_sun_right_ascension ;

double sun_hour_angle ;                        /* radians */

double geodetic_latitude ;              /* of inertial_position (precession & nutation assumed */
                                        /*                       to be neglible) */
double eta ;                                   /* radians */
double sin_m_theta ;                           /* sin ** m (theta), where m = 2.5 */

double diurnal_variation ;                     /* kelvin */
double exospheric_temperature ;                /* kelvin */

double delta_exospheric_temperature ;          /* kelvin       */
double delta_altitude ;                        /* km           */
double delta_log_density ;                     /* accounts for */
                                               /* seasonal-    */
                                               /* latitudinal  */
                                               /* variations   */


/*----------------------//
// compute altitude index //
//----------------------*/

sin_theta = earth_fixed_position[3]/radius ;
```

```
altitude = radius - 6368.14 / sqrt (1.0 +
        planet_oblate_coefficient * pow(sin_theta,2.0)) ;

if(altitude < 10.0)
 {
 /*-------------------------------------//
 // will extrapolate linearly below 10 km //
 //-------------------------------------*/
   dht = 5.0;
   table_index_1  = 3 ;
 }
else if(altitude < 90.0)
 {
   dht = 5.0;
   table_index_1 = (int)floor((altitude - 10.0)/5.0) + 3;
 }
else if(altitude < 110.0)
 {
   dht = 2.0;
   table_index_1 = (int)floor((altitude - 90.0)/2.0) + 19;
 }
else if(altitude < 160.0)
 {
   dht = 5.0;
   table_index_1 = (int)floor((altitude - 110.0)/5.0) + 29;
 }
else if(altitude < 400.0)
 {
   dht = 10.0;
   table_index_1 = (int)floor((altitude - 160.0)/10.0) + 39;
 }
else if (altitude < 1000.0)
 {
   dht = 20.0;
   table_index_1 = (int)floor((altitude - 400.0)/20.0) + 63;
 }
else if(altitude < 1500.0)
 {
   dht = 50.0;
   table_index_1 = (int)floor((altitude - 1000.0)/50.0) + 93;
 }
else
 {
   dht = 100.0;
   table_index_1 = (int)floor((altitude - 1500.0)/100.0) + 103;
   if(table_index_1 > 112)
    {
    /*-------------------------------------//
    // will extrapolate linearly above 2500 km //
    //-------------------------------------*/
```

```
    table_index_1 = 112;
   }
 }

previous_table_index_1 = table_index_1 - 1;

if(altitude > 90.0)
 {
/*-------------------------------------------------------//
// calculate diurnal_variation, equation 17, (jacchia, 1970) //
//-------------------------------------------------------*/
  equatorial_solar_ephemeris (at_time, sun_vector);

  sun_declination = asin (sun_vector [3]);

  sun_right_ascension = atan2 (sun_vector [2], sun_vector [1]);
  sin_sun_right_ascension = sin (sun_right_ascension);
  cos_sun_right_ascension = cos (sun_right_ascension);

  sun_hour_angle = atan2
                   ((inertial_position [2] * cos_sun_right_ascension -
                   inertial_position [1] * sin_sun_right_ascension),
                   (inertial_position [1] * cos_sun_right_ascension +
                   inertial_position [2] * sin_sun_right_ascension)) ;

  tau = sun_hour_angle + beta + p * sin (sun_hour_angle + gamma);

  if(tau < -PI)
   {
    tau = tau + TWO_PI;
   }
  else if(tau > PI)
   {
    tau = tau - TWO_PI;
   }

  geodetic_latitude = atan ((inertial_position [3] /
                      sqrt (pow(inertial_position [1],2.0) +
                      pow(inertial_position [2],2.0))) /
                 polar_equatorial_ratio_squared);

  eta = 0.5 * (geodetic_latitude - sun_declination);
  sin_m_theta = pow((sin (0.5 *fabs (geodetic_latitude +
                sun_declination))), 2.5);


/* equation 14, (jacchia, 1970) */
  exospheric_minimum_temperature =
       383.0 + 1.52 * mean_solar_flux + 1.8 * solar_flux;
```

```
  diurnal_variation = exospheric_minimum_temperature *
            (1.0 + r * sin_m_theta +
             r * ( pow(cos (eta),2.5) - sin_m_theta) *
                  pow(cos (tau / 2.0),3.0));
/*-------------------------------------------------------------//
// calculate geomagnetic_adjustment, equation 22, (jacchia, 1970) //
//-------------------------------------------------------------*/
geomagnetic_adjustment =
        geomagnetic_index_ap + 100.0 *
        (1.0 - exp (-0.08 * geomagnetic_index_ap));

/* calculate semi_annual_variation, equation 23, (jacchia, 1970) */

jdtocd (at_time, &month, &day, &year, &hour, &minute, &seconds);
last_day = julian_date(year-1, 12, 31, 0, 0, 0.0) ;
day_ratio = (at_time - last_day) / days_in_tropical_year;

tau = day_ratio + 0.1145 *
        (pow(((1.0 + sin (TWO_PI * day_ratio + 342.3 * DEGREES))
        / 2.0),2.16) - 0.5);

semi_annual_variation = 2.41 + mean_solar_flux *
        (0.349 + 0.206 * sin (TWO_PI * tau + 226.5 * DEGREES)) *
        sin (2.0 * TWO_PI * tau + 247.6 * DEGREES);

exospheric_temperature = diurnal_variation +
        geomagnetic_adjustment + semi_annual_variation;

if(exospheric_temperature > 600.0)
 {
  if(exospheric_temperature < 2000.0)
   {
    previous_table_index_2 = (int)floor(0.02 *
        (exospheric_temperature - 600.0)) + 2;
    delta_exospheric_temperature = 0.02 *
        (exospheric_temperature -
        density_table[1][previous_table_index_2]);
   }
  else
   {    /*use as 2000 (table_index_2 will be 30th point in table) */
    previous_table_index_2 = 29;
    delta_exospheric_temperature = 1.0;
   }

  table_index_2 = previous_table_index_2 + 1;

  log_density = density_table [previous_table_index_1][previous_table_index_2] +
            delta_exospheric_temperature *
            (density_table [previous_table_index_1][table_index_2] -
            density_table [previous_table_index_1][previous_table_index_2]);
```

```
  delta_log_density = density_table [table_index_1][previous_table_index_2] +
                      delta_exospheric_temperature*
                      (density_table [table_index_1][table_index_2] -
                      density_table [table_index_1][previous_table_index_2]);
 }
else
 {
 /*-------------------------------------//
 // assume exospheric_temperature is 600,//
 // interpolate on altitude only       //
 //-------------------------------------*/
log_density       = density_table [previous_table_index_1][2];
delta_log_density = density_table [table_index_1][2];

 }

delta_altitude = (altitude - density_table [previous_table_index_1][1]) /dht;

log_density = log_density + delta_altitude *
              (delta_log_density - log_density);

if(altitude < 300.0)
 {                                                    /* ( but >= 90 ) */
  /*-----------------------------//
  // equation 24, (jacchia, 1970) //
  //-----------------------------*/
  delta_log_density = 0.02 * (altitude - 90.0) *
                      exp (-0.045 * (altitude - 90.0)) *
                (pow(inertial_position [3],2.0)/
                      dot (inertial_position,inertial_position)) *
                sin (TWO_PI * (day + 99.0) /
                    days_in_tropical_year);
 }
else
 {
  delta_log_density = 0.0;
 }
 }
else
 { /* altitude < 90 here, and there is no temperature effect */

  delta_altitude = (altitude -
        density_table [previous_table_index_1][1]) / dht;

  log_density = density_table [previous_table_index_1][2] -
        delta_altitude * (density_table [table_index_1][2] -
        density_table [previous_table_index_1][2]);

  delta_log_density = 0.0;
```

```
        }                                                   /* altitude > 90.0 */

        return pow(10.0,(log_density + delta_log_density));
                                                            /* (grams/cm^3) */
 }

/*****************************************************************************
 This is the function equatorial_solar_ephemeris
 Input arguments are -
        at_time = julian date
 Return arguments are -
        sun_vector = normalized sun pointing vector

 'equitorial_solar_ephemeris' is called by 'jacchia_density'
 'equitorial_solar_ephemeris' calls -
                                        'jdtocd' in util.c
                                        'julian_date' in util.c
*****************************************************************************/

        void    equatorial_solar_ephemeris (at_time,sun_vector)
        double  at_time, sun_vector[] ;
 {

        double  celestial_longitude, mean_julian_date ;
        double  cos_declination, cos_right_ascension ;
        double  sin_declination, sin_right_ascension ;

        double  day_ratio, year, seconds ;

        int     whole_year, day, month, hour, minute ;

        jdtocd (at_time, &month, &day, &whole_year, &hour, &minute, &seconds);
        day = at_time - julian_date(whole_year, 1, 1, 0, 0, 0.0) ;

        day_ratio = day / days_in_tropical_year;
        year = whole_year + day_ratio;

        mean_julian_date = 2439856.0 + days_in_tropical_year *
                           (year - 1968.0) - 2435839.0;

        celestial_longitude = 0.017203 * mean_julian_date +
                        0.0335 * sin (0.017203 * mean_julian_date) - 1.41;
        celestial_longitude = celestial_longitude - TWO_PI *
                        ( (int) (celestial_longitude / (TWO_PI)));

        sin_declination = sin (celestial_longitude) * sin (OBLIQUITY);
        cos_declination = sqrt (1.0 - pow(sin_declination,2.0));

        sin_right_ascension = sin_declination /
                        (cos_declination * tan (OBLIQUITY));
```

```
        cos_right_ascension = cos (celestial_longitude) / cos_declination;

        sun_vector[1] = cos_declination * cos_right_ascension ;
        sun_vector[2] = cos_declination * sin_right_ascension ;
        sun_vector[3] = sin_declination ;

    }
/*****************************************************************************/
```

```
/*=============================================================
============================
 * built by Bob Gottlieb and Mike Fraietta
 *  Jan 1993
/*=============================================================
============================
 *-------------------------------------------------------------*/
static double
   far c1[31]  = { 0.0,    0.00,
                   600.00,   650.00,   700.00,   750.00,   800.00,
                   850.00,   900.00,   950.00,  1000.00,  1050.00,
                  1100.00,  1150.00,  1200.00,  1250.00,  1300.00,
                  1350.00,  1400.00,  1450.00,  1500.00,  1550.00,
                  1600.00,  1650.00,  1700.00,  1750.00,  1800.00,
                  1850.00,  1900.00,  1950.00,  2000.00 },

   far c2[31]  = { 0.0,    10.00000,
                    -3.38437, -3.38437, -3.38437, -3.38437, -3.38437,
                    -3.38437, -3.38437, -3.38437, -3.38437, -3.38437,
                    -3.38437, -3.38437, -3.38437, -3.38437, -3.38437,
                    -3.38437, -3.38437, -3.38437, -3.38437, -3.38437,
                    -3.38437, -3.38437, -3.38437, -3.38437, -3.38437,
                    -3.38437, -3.38437, -3.38437, -3.38437 },

   far c3[31]  = { 0.0,   15.00000,
                    -3.71287, -3.71287, -3.71287, -3.71287, -3.71287,
                    -3.71287, -3.71287, -3.71287, -3.71287, -3.71287,
                    -3.71287, -3.71287, -3.71287, -3.71287, -3.71287,
                    -3.71287, -3.71287, -3.71287, -3.71287, -3.71287,
                    -3.71287, -3.71287, -3.71287, -3.71287, -3.71287,
                    -3.71287, -3.71287, -3.71287, -3.71287 },

   far c4[31]  = { 0.0,   20.00000,
                    -4.05530, -4.05530, -4.05530, -4.05530, -4.05530,
                    -4.05530, -4.05530, -4.05530, -4.05530, -4.05530,
                    -4.05530, -4.05530, -4.05530, -4.05530, -4.05530,
                    -4.05530, -4.05530, -4.05530, -4.05530, -4.05530,
                    -4.05530, -4.05530, -4.05530, -4.05530, -4.05530,
                    -4.05530, -4.05530, -4.05530, -4.05530 },

   far c5[31]  = { 0.0,   25.00000,
                    -4.40370, -4.40370, -4.40370, -4.40370, -4.40370,
                    -4.40370, -4.40370, -4.40370, -4.40370, -4.40370,
                    -4.40370, -4.40370, -4.40370, -4.40370, -4.40370,
                    -4.40370, -4.40370, -4.40370, -4.40370, -4.40370,
                    -4.40370, -4.40370, -4.40370, -4.40370, -4.40370,
                    -4.40370, -4.40370, -4.40370, -4.40370 },

   far c6[31]  = { 0.0,   30.00000,
                    -4.74450, -4.74450, -4.74450, -4.74450, -4.74450,
```

```
                -4.74450,  -4.74450,  -4.74450,  -4.74450,  -4.74450,
                -4.74450,  -4.74450,  -4.74450,  -4.74450,  -4.74450,
                -4.74450,  -4.74450,  -4.74450,  -4.74450,  -4.74450,
                -4.74450,  -4.74450,  -4.74450,  -4.74450,  -4.74450,
                -4.74450,  -4.74450,  -4.74450,  -4.74450 },

far c7[31]  = { 0.0,   35.00000,
                -5.08540,  -5.08540,  -5.08540,  -5.08540,  -5.08540,
                -5.08540,  -5.08540,  -5.08540,  -5.08540,  -5.08540,
                -5.08540,  -5.08540,  -5.08540,  -5.08540,  -5.08540,
                -5.08540,  -5.08540,  -5.08540,  -5.08540,  -5.08540,
                -5.08540,  -5.08540,  -5.08540,  -5.08540,  -5.08540,
                -5.08540,  -5.08540,  -5.08540,  -5.08540 },

far c8[31]  = { 0.0,   40.00000,
                -5.41440,  -5.41440,  -5.41440,  -5.41440,  -5.41440,
                -5.41440,  -5.41440,  -5.41440,  -5.41440,  -5.41440,
                -5.41440,  -5.41440,  -5.41440,  -5.41440,  -5.41440,
                -5.41440,  -5.41440,  -5.41440,  -5.41440,  -5.41440,
                -5.41440,  -5.41440,  -5.41440,  -5.41440,  -5.41440,
                -5.41440,  -5.41440,  -5.41440,  -5.41440 },

far c9[31]  = { 0.0,   45.00000,
                -5.72560,  -5.72560,  -5.72560,  -5.72560,  -5.72560,
                -5.72560,  -5.72560,  -5.72560,  -5.72560,  -5.72560,
                -5.72560,  -5.72560,  -5.72560,  -5.72560,  -5.72560,
                -5.72560,  -5.72560,  -5.72560,  -5.72560,  -5.72560,
                -5.72560,  -5.72560,  -5.72560,  -5.72560,  -5.72560,
                -5.72560,  -5.72560,  -5.72560,  -5.72560 },

far c10[31] = { 0.0,   50.00000,
                -6.00990,  -6.00990,  -6.00990,  -6.00990,  -6.00990,
                -6.00990,  -6.00990,  -6.00990,  -6.00990,  -6.00990,
                -6.00990,  -6.00990,  -6.00990,  -6.00990,  -6.00990,
                -6.00990,  -6.00990,  -6.00990,  -6.00990,  -6.00990,
                -6.00990,  -6.00990,  -6.00990,  -6.00990,  -6.00990,
                -6.00990,  -6.00990,  -6.00990,  -6.00990},

far c11[31] = { 0.0,   55.00000,
                -6.27610,  -6.27610,  -6.27610,  -6.27610,  -6.27610,
                -6.27610,  -6.27610,  -6.27610,  -6.27610,  -6.27610,
                -6.27610,  -6.27610,  -6.27610,  -6.27610,  -6.27610,
                -6.27610,  -6.27610,  -6.27610,  -6.27610,  -6.27610,
                -6.27610,  -6.27610,  -6.27610,  -6.27610,  -6.27610,
                -6.27610,  -6.27610,  -6.27610,  -6.27610},

far c12[31] = { 0.0,   60.00000,
                -6.54520,  -6.54520,  -6.54520,  -6.54520,  -6.54520,
                -6.54520,  -6.54520,  -6.54520,  -6.54520,  -6.54520,
                -6.54520,  -6.54520,  -6.54520,  -6.54520,  -6.54520,
```

```
                -6.54520,  -6.54520,  -6.54520,  -6.54520,  -6.54520,
                -6.54520,  -6.54520,  -6.54520,  -6.54520,  -6.54520,
                -6.54520,  -6.54520,  -6.54520,  -6.54520},

far c13[31] = { 0.0,   65.00000,
                -6.81450,  -6.81450,  -6.81450,  -6.81450,  -6.81450,
                -6.81450,  -6.81450,  -6.81450,  -6.81450,  -6.81450,
                -6.81450,  -6.81450,  -6.81450,  -6.81450,  -6.81450,
                -6.81450,  -6.81450,  -6.81450,  -6.81450,  -6.81450,
                -6.81450,  -6.81450,  -6.81450,  -6.81450,  -6.81450,
                -6.81450,  -6.81450,  -6.81450,  -6.81450},

far c14[31] = { 0.0,   70.00000,
                -7.10550,  -7.10550,  -7.10550,  -7.10550,  -7.10550,
                -7.10550,  -7.10550,  -7.10550,  -7.10550,  -7.10550,
                -7.10550,  -7.10550,  -7.10550,  -7.10550,  -7.10550,
                -7.10550,  -7.10550,  -7.10550,  -7.10550,  -7.10550,
                -7.10550,  -7.10550,  -7.10550,  -7.10550,  -7.10550,
                -7.10550,  -7.10550,  -7.10550,  -7.10550},

far c15[31] = { 0.0,   75.00000,
                -7.42760,  -7.42760,  -7.42760,  -7.42760,  -7.42760,
                -7.42760,  -7.42760,  -7.42760,  -7.42760,  -7.42760,
                -7.42760,  -7.42760,  -7.42760,  -7.42760,  -7.42760,
                -7.42760,  -7.42760,  -7.42760,  -7.42760,  -7.42760,
                -7.42760,  -7.42760,  -7.42760,  -7.42760,  -7.42760,
                -7.42760,  -7.42760,  -7.42760,  -7.42760},

far c16[31] = { 0.0,   80.00000,
                -7.78070,  -7.78070,  -7.78070,  -7.78070,  -7.78070,
                -7.78070,  -7.78070,  -7.78070,  -7.78070,  -7.78070,
                -7.78070,  -7.78070,  -7.78070,  -7.78070,  -7.78070,
                -7.78070,  -7.78070,  -7.78070,  -7.78070,  -7.78070,
                -7.78070,  -7.78070,  -7.78070,  -7.78070,  -7.78070,
                -7.78070,  -7.78070,  -7.78070,  -7.78070},

far c17[31] = { 0.0,   85.00000,
                -8.16990,  -8.16990,  -8.16990,  -8.16990,  -8.16990,
                -8.16990,  -8.16990,  -8.16990,  -8.16990,  -8.16990,
                -8.16990,  -8.16990,  -8.16990,  -8.16990,  -8.16990,
                -8.16990,  -8.16990,  -8.16990,  -8.16990,  -8.16990,
                -8.16990,  -8.16990,  -8.16990,  -8.16990,  -8.16990,
                -8.16990,  -8.16990,  -8.16990,  -8.16990},

far c18[31] = { 0.0,   90.00000,
                -8.46100,  -8.46100,  -8.46100,  -8.46100,  -8.46100,
                -8.46100,  -8.46100,  -8.46100,  -8.46100,  -8.46100,
                -8.46100,  -8.46100,  -8.46100,  -8.46100,  -8.46100,
                -8.46100,  -8.46100,  -8.46100,  -8.46100,  -8.46100,
                -8.46100,  -8.46100,  -8.46100,  -8.46100,  -8.46100,
```

```
                -8.46100,  -8.46100,  -8.46100,  -8.46100},

far c19[31] = { 0.0,   92.00000,
                -8.62000,  -8.62000,  -8.62000,  -8.62000,  -8.62000,
                -8.62000,  -8.62000,  -8.62000,  -8.62000,  -8.62000,
                -8.62000,  -8.62000,  -8.62000,  -8.62000,  -8.62000,
                -8.62000,  -8.62000,  -8.62000,  -8.62000,  -8.62000,
                -8.62000,  -8.62000,  -8.62000,  -8.62000,  -8.62000,
                -8.62000,  -8.62000,  -8.62000,  -8.62000},

far c20[31] = { 0.0,   94.00000,
                -8.77900,  -8.77900,  -8.78000,  -8.78000,  -8.78000,
                -8.78000,  -8.78000,  -8.78000,  -8.78000,  -8.78000,
                -8.78000,  -8.78000,  -8.78000,  -8.78000,  -8.78000,
                -8.78000,  -8.78000,  -8.78100,  -8.78100,  -8.78100,
                -8.78100,  -8.78100,  -8.78100,  -8.78100,  -8.78100,
                -8.78100,  -8.78100,  -8.78100,  -8.78100},

far c21[31] = { 0.0,   96.00000,
                -8.93900,  -8.94000,  -8.94000,  -8.94000,  -8.94000,
                -8.94100,  -8.94100,  -8.94100,  -8.94100,  -8.94100,
                -8.94100,  -8.94200,  -8.94200,  -8.94200,  -8.94200,
                -8.94200,  -8.94200,  -8.94200,  -8.94200,  -8.94200,
                -8.94200,  -8.94200,  -8.94200,  -8.94200,  -8.94300,
                -8.94300,  -8.94300,  -8.94300,  -8.94300},

far c22[31] = { 0.0,   98.00000,
                -9.09900,  -9.10000,  -9.10000,  -9.10100,  -9.10100,
                -9.10100,  -9.10200,  -9.10200,  -9.10200,  -9.10200,
                -9.10300,  -9.10300,  -9.10300,  -9.10300,  -9.10300,
                -9.10400,  -9.10400,  -9.10400,  -9.10400,  -9.10400,
                -9.10400,  -9.10400,  -9.10400,  -9.10400,  -9.10500,
                -9.10500,  -9.10500,  -9.10500,  -9.10500},

far c23[31] = { 0.0,  100.00000,
                -9.25800,  -9.25900,  -9.25900,  -9.26000,  -9.26100,
                -9.26100,  -9.26200,  -9.26200,  -9.26200,  -9.26300,
                -9.26300,  -9.26300,  -9.26400,  -9.26400,  -9.26400,
                -9.26400,  -9.26400,  -9.26500,  -9.26500,  -9.26500,
                -9.26500,  -9.26500,  -9.26500,  -9.26500,  -9.26600,
                -9.26600,  -9.26600,  -9.26600,  -9.26600},

far c24[31] = { 0.0,  102.00000,
                -9.41500,  -9.41600,  -9.41700,  -9.41800,  -9.41800,
                -9.41900,  -9.42000,  -9.42000,  -9.42100,  -9.42100,
                -9.42100,  -9.42200,  -9.42200,  -9.42200,  -9.42300,
                -9.42300,  -9.42300,  -9.42300,  -9.42400,  -9.42400,
                -9.42400,  -9.42400,  -9.42400,  -9.42400,  -9.42500,
                -9.42500,  -9.42500,  -9.42500,  -9.42500},
```

```
far c25[31] = { 0.0,  104.00000,
                -9.57000, -9.57100, -9.57200, -9.57300, -9.57400,
                -9.57400, -9.57500, -9.57500, -9.57600, -9.57600,
                -9.57700, -9.57700, -9.57700, -9.57800, -9.57800,
                -9.57800, -9.57900, -9.57900, -9.57900, -9.57900,
                -9.58000, -9.58000, -9.58000, -9.58000, -9.58000,
                -9.58000, -9.58000, -9.58100, -9.58100},

far c26[31] = { 0.0,  106.00000,
                -9.72200, -9.72300, -9.72400, -9.72500, -9.72500,
                -9.72600, -9.72600, -9.72700, -9.72700, -9.72800,
                -9.72800, -9.72800, -9.72900, -9.72900, -9.72900,
                -9.73000, -9.73000, -9.73000, -9.73000, -9.73000,
                -9.73100, -9.73100, -9.73100, -9.73100, -9.73100,
                -9.73100, -9.73200, -9.73200, -9.73200},

far c27[31] = { 0.0,  108.00000,
                -9.87000, -9.87100, -9.87100, -9.87200, -9.87200,
                -9.87300, -9.87300, -9.87300, -9.87400, -9.87400,
                -9.87400, -9.87500, -9.87500, -9.87500, -9.87500,
                -9.87600, -9.87600, -9.87600, -9.87600, -9.87600,
                -9.87600, -9.87600, -9.87700, -9.87700, -9.87700,
                -9.87700, -9.87700, -9.87700, -9.87700},

far c28[31] = { 0.0,  110.00000,
                -10.01400, -10.01400, -10.01400, -10.01400, -10.01400,
                -10.01500, -10.01500, -10.01500, -10.01500, -10.01500,
                -10.01500, -10.01500, -10.01500, -10.01500, -10.01500,
                -10.01600, -10.01600, -10.01600, -10.01600, -10.01600,
                -10.01600, -10.01600, -10.01600, -10.01600, -10.01600,
                -10.01600, -10.01600, -10.01600, -10.01600},

far c29[31] = { 0.0,  115.00000,
                -10.35000, -10.34800, -10.34600, -10.34500, -10.34400,
                -10.34300, -10.34200, -10.34100, -10.34000, -10.33900,
                -10.33900, -10.33800, -10.33800, -10.33700, -10.33700,
                -10.33600, -10.33600, -10.33600, -10.33600, -10.33500,
                -10.33500, -10.33500, -10.33500, -10.33400, -10.33400,
                -10.33400, -10.33400, -10.33400, -10.33300},

far c30[31] = { 0.0,  120.00000,
                -10.65000, -10.64500, -10.64100, -10.63700, -10.63400,
                -10.63100, -10.62800, -10.62600, -10.62400, -10.62200,
                -10.62100, -10.61900, -10.61800, -10.61700, -10.61500,
                -10.61400, -10.61300, -10.61300, -10.61200, -10.61100,
                -10.61000, -10.61000, -10.60900, -10.60800, -10.60800,
                -10.60700, -10.60700, -10.60600, -10.60600},

far c31[31] = { 0.0,  125.00000,
                -10.91400, -10.90500, -10.89700, -10.89100, -10.88500,
```

```
                -10.88000, -10.87600, -10.87200, -10.86800, -10.86500,
                -10.86200, -10.86000, -10.85700, -10.85500, -10.85300,
                -10.85100, -10.85000, -10.84800, -10.84700, -10.84600,
                -10.84400, -10.84300, -10.84200, -10.84100, -10.84000,
                -10.83900, -10.83800, -10.83800, -10.83700},

far c32[31] = { 0.0,  130.00000,
                -11.14300, -11.13000, -11.11900, -11.10900, -11.10100,
                -11.09400, -11.08700, -11.08100, -11.07600, -11.07200,
                -11.06800, -11.06400, -11.06100, -11.05800, -11.05500,
                -11.05200, -11.05000, -11.04800, -11.04600, -11.04400,
                -11.04200, -11.04100, -11.03900, -11.03800, -11.03600,
                -11.03500, -11.03400, -11.03300, -11.03200},

far c33[31] = { 0.0,  135.00000,
                -11.34000, -11.32400, -11.30900, -11.29700, -11.28600,
                -11.27700, -11.26900, -11.26100, -11.25500, -11.24900,
                -11.24400, -11.23900, -11.23500, -11.23100, -11.22800,
                -11.22500, -11.22200, -11.21900, -11.21600, -11.21400,
                -11.21200, -11.21000, -11.20800, -11.20600, -11.20500,
                -11.20300, -11.20100, -11.20000, -11.19900},

far c34[31] = { 0.0,  140.00000,
                -11.51300, -11.49200, -11.47500, -11.46000, -11.44700,
                -11.43600, -11.42600, -11.41700, -11.41000, -11.40300,
                -11.39700, -11.39100, -11.38600, -11.38200, -11.37800,
                -11.37400, -11.37100, -11.36800, -11.36500, -11.36200,
                -11.35900, -11.35700, -11.35500, -11.35300, -11.35100,
                -11.34900, -11.34700, -11.34600, -11.34400},

far c35[31] = { 0.0,  145.00000,
                -11.66700, -11.64200, -11.62200, -11.60400, -11.58900,
                -11.57600, -11.56400, -11.55400, -11.54600, -11.53800,
                -11.53100, -11.52500, -11.51900, -11.51500, -11.51000,
                -11.50600, -11.50200, -11.49900, -11.49500, -11.49200,
                -11.49000, -11.48700, -11.48500, -11.48200, -11.48000,
                -11.47800, -11.47600, -11.47400, -11.47200},

far c36[31] = { 0.0,  150.00000,
                -11.80800, -11.77900, -11.75500, -11.73400, -11.71700,
                -11.70100, -11.68900, -11.67700, -11.66700, -11.65900,
                -11.65100, -11.64400, -11.63800, -11.63300, -11.62800,
                -11.62300, -11.61900, -11.61500, -11.61200, -11.60900,
                -11.60600, -11.60300, -11.60000, -11.59800, -11.59600,
                -11.59300, -11.59100, -11.58900, -11.58700},

far c37[31] = { 0.0,  155.00000,
                -11.94000, -11.90600, -11.87800, -11.85400, -11.83400,
                -11.81700, -11.80200, -11.78900, -11.77800, -11.76800,
                -11.76000, -11.75200, -11.74600, -11.74000, -11.73400,
```

```
                -11.72900, -11.72500, -11.72100, -11.71700, -11.71400,
                -11.71100, -11.70800, -11.70500, -11.70200, -11.70000,
                -11.69700, -11.69500, -11.69300, -11.69100},

far c38[31] = { 0.0,  160.00000,
                -12.06400, -12.02500, -11.99300, -11.96600, -11.94300,
                -11.92300, -11.90700, -11.89200, -11.88000, -11.86900,
                -11.86000, -11.85100, -11.84400, -11.83700, -11.83100,
                -11.82600, -11.82100, -11.81700, -11.81300, -11.80900,
                -11.80600, -11.80300, -11.80000, -11.79700, -11.79500,
                -11.79200, -11.79000, -11.78800, -11.78600},

far c39[31] = { 0.0,  170.00000,
                -12.29600, -12.24800, -12.20700, -12.17200, -12.14300,
                -12.11900, -12.09800, -12.08000, -12.06400, -12.05100,
                -12.03900, -12.02900, -12.02000, -12.01200, -12.00500,
                -11.99800, -11.99300, -11.98800, -11.98300, -11.97900,
                -11.97500, -11.97200, -11.96800, -11.96500, -11.96200,
                -11.96000, -11.95700, -11.95500, -11.95300},

far c40[31] = { 0.0,  180.00000,
                -12.51200, -12.45300, -12.40400, -12.36200, -12.32700,
                -12.29700, -12.27100, -12.24900, -12.23000, -12.21300,
                -12.19900, -12.18600, -12.17500, -12.16600, -12.15700,
                -12.14900, -12.14300, -12.13700, -12.13100, -12.12600,
                -12.12200, -12.11800, -12.11400, -12.11100, -12.10800,
                -12.10500, -12.10200, -12.09900, -12.09700},


far c41[31] = { 0.0,  190.00000,
                -12.71400, -12.64500, -12.58800, -12.53900, -12.49800,
                -12.46200, -12.43100, -12.40500, -12.38200, -12.36200,
                -12.34500, -12.33000, -12.31700, -12.30500, -12.29500,
                -12.28600, -12.27700, -12.27000, -12.26400, -12.25800,
                -12.25300, -12.24800, -12.24400, -12.24000, -12.23600,
                -12.23300, -12.23000, -12.22700, -12.22400},

far c42[31] = { 0.0,  200.00000,
                -12.90400, -12.82700, -12.76200, -12.70600, -12.65800,
                -12.61700, -12.58200, -12.55100, -12.52400, -12.50100,
                -12.48100, -12.46300, -12.44700, -12.43300, -12.42100,
                -12.41000, -12.40100, -12.39200, -12.38400, -12.37700,
                -12.37100, -12.36600, -12.36100, -12.35600, -12.35200,
                -12.34800, -12.34400, -12.34100, -12.33800},

far c43[31] = { 0.0,  210.00000,
                -13.08500, -13.00000, -12.92700, -12.86400, -12.81100,
                -12.76400, -12.72400, -12.68900, -12.65800, -12.63200,
                -12.60800, -12.58700, -12.56900, -12.55300, -12.53900,
                -12.52600, -12.51500, -12.50500, -12.49500, -12.48700,
```

```
                -12.48000,  -12.47300,  -12.46700,  -12.46200,  -12.45700,
                -12.45200,  -12.44800,  -12.44400,  -12.44100},

far c44[31] = { 0.0,  220.00000,
                -13.25800,  -13.16400,  -13.08400,  -13.01500,  -12.95600,
                -12.90400,  -12.85900,  -12.82000,  -12.78600,  -12.75500,
                -12.72900,  -12.70500,  -12.68400,  -12.66600,  -12.64900,
                -12.63500,  -12.62100,  -12.61000,  -12.59900,  -12.59000,
                -12.58100,  -12.57300,  -12.56600,  -12.56000,  -12.55400,
                -12.54900,  -12.54400,  -12.53900,  -12.53500},

far c45[31] = { 0.0,  230.00000,
                -13.42200,  -13.32100,  -13.23400,  -13.16000,  -13.09500,
                -13.03800,  -12.98900,  -12.94500,  -12.90700,  -12.87300,
                -12.84400,  -12.81700,  -12.79400,  -12.77300,  -12.75400,
                -12.73700,  -12.72200,  -12.70900,  -12.69700,  -12.68600,
                -12.67600,  -12.66700,  -12.65900,  -12.65100,  -12.64400,
                -12.63800,  -12.63200,  -12.62700,  -12.62200},

far c46[31] = { 0.0,  240.00000,
                -13.58100,  -13.47200,  -13.37900,  -13.29800,  -13.22800,
                -13.16700,  -13.11300,  -13.06500,  -13.02300,  -12.98600,
                -12.95300,  -12.92400,  -12.89800,  -12.87500,  -12.85400,
                -12.83500,  -12.81800,  -12.80300,  -12.78900,  -12.77600,
                -12.76500,  -12.75500,  -12.74500,  -12.73700,  -12.72900,
                -12.72200,  -12.71500,  -12.70900,  -12.70400},

far c47[31] = { 0.0,  250.00000,
                -13.73300,  -13.61700,  -13.51800,  -13.43100,  -13.35600,
                -13.29000,  -13.23200,  -13.18100,  -13.13500,  -13.09500,
                -13.05900,  -13.02700,  -12.99800,  -12.97200,  -12.94900,
                -12.92800,  -12.90900,  -12.89200,  -12.87700,  -12.86300,
                -12.85000,  -12.83800,  -12.82800,  -12.81800,  -12.80900,
                -12.80100,  -12.79300,  -12.78600,  -12.78000},

far c48[31] = { 0.0,  260.00000,
                -13.88100,  -13.75800,  -13.65200,  -13.56000,  -13.48000,
                -13.40900,  -13.34700,  -13.29200,  -13.24300,  -13.19900,
                -13.16000,  -13.12600,  -13.09400,  -13.06600,  -13.04100,
                -13.01800,  -12.99700,  -12.97800,  -12.96100,  -12.94500,
                -12.93100,  -12.91800,  -12.90600,  -12.89500,  -12.88500,
                -12.87600,  -12.86700,  -12.85900,  -12.85200},

far c49[31] = { 0.0,  270.00000,
                -14.02400,  -13.89400,  -13.78200,  -13.68400,  -13.59900,
                -13.52400,  -13.45800,  -13.39900,  -13.34700,  -13.30000,
                -13.25900,  -13.22100,  -13.18700,  -13.15700,  -13.12900,
                -13.10400,  -13.08100,  -13.06000,  -13.04200,  -13.02400,
                -13.00900,  -12.99400,  -12.98100,  -12.96900,  -12.95800,
                -12.94700,  -12.3800,  -12.92900,  -12.92000},
```

```
far c50[31] = { 0.0,  280.00000,
                -14.16400,  -14.02600,  -13.90800,  -13.80500,  -13.71500,
                -13.63600,  -13.56600,  -13.50400,  -13.44800,  -13.39800,
                -13.35300,  -13.31300,  -13.27700,  -13.24400,  -13.21400,
                -13.18700,  -13.16300,  -13.14000,  -13.12000,  -13.10100,
                -13.08400,  -13.06800,  -13.05300,  -13.04000,  -13.02700,
                -13.01600,  -13.00500,  -12.99500,  -12.98600},

far c51[31] = { 0.0,  290.00000,
                -14.30000,  -14.15500,  -14.03000,  -13.92200,  -13.82800,
                -13.74400,  -13.67000,  -13.60500,  -13.54600,  -13.49300,
                -13.44600,  -13.40300,  -13.36400,  -13.32900,  -13.29700,
                -13.26800,  -13.24200,  -13.21700,  -13.19500,  -13.17500,
                -13.15600,  -13.13900,  -13.12300,  -13.10800,  -13.09400,
                -13.08200,  -13.07000,  -13.05900,  -13.04900},

far c52[31] = { 0.0,  300.00000,
                -14.43400,  -14.28100,  -14.15000,  -14.03700,  -13.93700,
                -13.85000,  -13.77200,  -13.70300,  -13.64100,  -13.58500,
                -13.53500,  -13.49000,  -13.44900,  -13.41100,  -13.37700,
                -13.34600,  -13.31800,  -13.29200,  -13.26800,  -13.24600,
                -13.22600,  -13.20800,  -13.19000,  -13.17400,  -13.16000,
                -13.14600,  -13.13300,  -13.12100,  -13.11000},

far c53[31] = { 0.0,  310.00000,
                -14.56500,  -14.40500,  -14.26800,  -14.14900,  -14.04400,
                -13.95300,  -13.87100,  -13.79900,  -13.73400,  -13.67500,
                -13.62200,  -13.57500,  -13.53100,  -13.49200,  -13.45600,
                -13.42300,  -13.39300,  -13.36500,  -13.34000,  -13.31600,
                -13.29400,  -13.27400,  -13.25600,  -13.23900,  -13.22300,
                -13.20800,  -13.19400,  -13.18100,  -13.16900},

far c54[31] = { 0.0,  320.00000,
                -14.69400,  -14.52700,  -14.38300,  -14.25800,  -14.14900,
                -14.05300,  -13.96800,  -13.89200,  -13.82400,  -13.76300,
                -13.70700,  -13.65700,  -13.61200,  -13.57000,  -13.53200,
                -13.49700,  -13.46500,  -13.43600,  -13.40900,  -13.38400,
                -13.36100,  -13.34000,  -13.32000,  -13.30100,  -13.28400,
                -13.26800,  -13.25300,  -13.23900,  -13.22600},

far c55[31] = { 0.0,  330.00000,
                -14.82100,  -14.64600,  -14.49600,  -14.36600,  -14.25200,
                -14.15200,  -14.06300,  -13.98400,  -13.91300,  -13.84900,
                -13.79100,  -13.73800,  -13.69000,  -13.64700,  -13.60700,
                -13.57000,  -13.53600,  -13.50500,  -13.47700,  -13.45000,
                -13.42600,  -13.40300,  -13.38200,  -13.36200,  -13.34400,
                -13.32700,  -13.31100,  -13.29600,  -13.28200},

far c56[31] = { 0.0,  340.00000,
                -14.94700,  -14.76400,  -14.60700,  -14.47200,  -14.35300,
```

```
                -14.24900,  -14.15600,  -14.07300,  -13.99900,  -13.93200,
                -13.87200,  -13.81700,  -13.76700,  -13.72100,  -13.67900,
                -13.64100,  -13.60600,  -13.57300,  -13.54300,  -13.51500,
                -13.48900,  -13.46500,  -13.44300,  -13.42200,  -13.40200,
                -13.38400,  -13.36700,  -13.35100,  -13.33600},

far c57[31] = { 0.0,  350.00000,
                -15.07000,  -14.88000,  -14.71700,  -14.57600,  -14.45200,
                -14.34400,  -14.24700,  -14.16100,  -14.08400,  -14.01500,
                -13.95200,  -13.89400,  -13.84200,  -13.79400,  -13.75100,
                -13.71100,  -13.67300,  -13.63900,  -13.60800,  -13.57800,
                -13.55100,  -13.52600,  -13.50200,  -13.48000,  -13.46000,
                -13.44000,  -13.42200,  -13.40500,  -13.38900},

far c58[31] = { 0.0,  360.00000,
                -15.19200,  -14.99500,  -14.82600,  -14.67900,  -14.55000,
                -14.43700,  -14.33700,  -14.24800,  -14.16800,  -14.09500,
                -14.03000,  -13.97000,  -13.91600,  -13.86600,  -13.82100,
                -13.77900,  -13.74000,  -13.70400,  -13.67100,  -13.64000,
                -13.61200,  -13.58500,  -13.56000,  -13.53700,  -13.51600,
                -13.49500,  -13.47600,  -13.45800,  -13.44100},

far c59[31] = { 0.0,  370.00000,
                -15.31300,  -15.10900,  -14.93300,  -14.78000,  -14.64700,
                -14.53000,  -14.42600,  -14.33300,  -14.25000,  -14.17500,
                -14.10700,  -14.04500,  -13.98800,  -13.93700,  -13.88900,
                -13.84600,  -13.80500,  -13.76800,  -13.73300,  -13.70100,
                -13.67100,  -13.64400,  -13.61800,  -13.59300,  -13.57100,
                -13.54900,  -13.52900,  -13.51000,  -13.49200},

far c60[31] = { 0.0,  380.00000,
                -15.43100,  -15.22100,  -15.03800,  -14.88000,  -14.74200,
                -14.62000,  -14.51300,  -14.41700,  -14.33000,  -14.25200,
                -14.18200,  -14.11800,  -14.05900,  -14.00600,  -13.95700,
                -13.91100,  -13.86900,  -13.83100,  -13.79500,  -13.76100,
                -13.73000,  -13.70100,  -13.67400,  -13.64800,  -13.62400,
                -13.60200,  -13.58100,  -13.56100,  -13.54200},

far c61[31] = { 0.0,  390.00000,
                -15.54800,  -15.33100,  -15.14300,  -14.97900,  -14.83600,
                -14.71000,  -14.59900,  -14.49900,  -14.41000,  -14.32900,
                -14.25600,  -14.19000,  -14.12900,  -14.07400,  -14.02300,
                -13.97600,  -13.93200,  -13.89200,  -13.85500,  -13.82000,
                -13.78700,  -13.75700,  -13.72900,  -13.70200,  -13.67700,
                -13.65400,  -13.63200,  -13.61100,  -13.59200},

far c62[31] = { 0.0,  400.00000,
                -15.66200,  -15.44000,  -15.24600,  -15.07700,  -14.92900,
                -14.79900,  -14.68400,  -14.58100,  -14.48800,  -14.40500,
                -14.33000,  -14.26100,  -14.19800,  -14.14100,  -14.08800,
```

```
                -14.03900, -13.99400, -13.95200, -13.91400, -13.87800,
                -13.84400, -13.81300, -13.78300, -13.75500, -13.73000,
                -13.70500, -13.68200, -13.66000, -13.64000},

far c63[31] = { 0.0,  420.00000,
                -15.88400, -15.65400, -15.44900, -15.27000, -15.11200,
                -14.97400, -14.85100, -14.74100, -14.64200, -14.55300,
                -14.47300, -14.40000, -14.33300, -14.27200, -14.21500,
                -14.16300, -14.11500, -14.07100, -14.02900, -13.99000,
                -13.95400, -13.92100, -13.88900, -13.85900, -13.83100,
                -13.80500, -13.78000, -13.75700, -13.73400},

far c64[31] = { 0.0,  440.00000,
                -16.09400, -15.86000, -15.64700, -15.45800, -15.29200,
                -15.14500, -15.01400, -14.89700, -14.79300, -14.69800,
                -14.61300, -14.53500, -14.46400, -14.39900, -14.33900,
                -14.28400, -14.23300, -14.18500, -14.14100, -14.10000,
                -14.06200, -14.02600, -13.99200, -13.96000, -13.93000,
                -13.90200, -13.87500, -13.85000, -13.82600},

far c65[31] = { 0.0,  460.00000,
                -16.29000, -16.05700, -15.83900, -15.64200, -15.46700,
                -15.31200, -15.17400, -15.05100, -14.94000, -14.84000,
                -14.75000, -14.66800, -14.59200, -14.52400, -14.46000,
                -14.40200, -14.34700, -14.29700, -14.25000, -14.20700,
                -14.16600, -14.12800, -14.09200, -14.05800, -14.02600,
                -13.99600, -13.96800, -13.94100, -13.91500},

far c66[31] = { 0.0,  480.00000,
                -16.46800, -16.24400, -16.02400, -15.82100, -15.63900,
                -15.47700, -15.33200, -15.20200, -15.08500, -14.98000,
                -14.88400, -14.79700, -14.71800, -14.64500, -14.57800,
                -14.51700, -14.45900, -14.40600, -14.35700, -14.31100,
                -14.26800, -14.22700, -14.18900, -14.15300, -14.12000,
                -14.08800, -14.05800, -14.02900, -14.00200},

*cc[67] =
{
     c1,  c1,  c2,  c3,  c4,  c5,  c6,  c7,  c8,  c9, c10, c11, c12, c13,
    c14, c15, c16, c17, c18, c19, c20, c21, c22, c23, c24, c25, c26, c27,
    c28, c29, c30, c31, c32, c33, c34, c35, c36, c37, c38, c39, c40, c41,
    c42, c43, c44, c45, c46, c47, c48, c49, c50, c51, c52, c53, c54, c55,
    c56, c57, c58, c59, c60, c61, c62, c63, c64, c65, c66
};

/*=============================================================
=========================
 *-------------------------------------------------------------*/
void jacchia_data(double D[68][31])
{
```

```
    /*-------------------------------------------------------------
     */
    int   i,j;

    /*-------------------------------------------------------------
     */
    for(i=0 ; i<=66 ; i++)
    {
        for(j=0 ; j<=29 ; j++)
        {
           D[i+1][j+1] = cc[i][j];
        }
    }
}
```

```
/*****************************************************************************
 This is the function compute_rnp

*****************************************************************************/
#include <math.h>
#include "rnpm50.h"
#include "rnpd.h"

      void    compute_rnp (reference_input, epoch_input, m50)
    RNP_REFERENCE *reference_input;
    EPOCH_REF     *epoch_input;
    RNP           *m50 ;
 {

    void precession_generator() ;
    void nutation_generator() ;
    void rotation_generator() ;
      /*-------------------------------------------------//
      // Include static data required for the rnp_m50 function.//
      //-------------------------------------------------*/

      double  T, TU, angle_modulo () ;
      double  argument;
      double  centuries_midnight_from_1900_ut1;
      double  centuries_ref_since_1900_tdt;
      double  centuries_ref_since_1950_trop;
      double  coef;
      double  greenwich_mean_siderial_time_ref;
      double  interim_argument;
      double  julian_date_midnight_tdt;
      double  julian_date_midnight_ut1;
      double  julian_date_tdt;
      double  julian_date_ut1;
      double  mean_angle[7];
      double  seconds_epoch_since_midnight_ut1;
      double  t_cubed;
      double  t_squared;
      double  temp;

      int     i;
      int     j;
      int     l;
      int     m;

      double  nutation_precession[4][4];
```

```
  /*-----------------------------------------//
  // Get a copy of the reference epoch and the   //
  // EPOCH_REF structure into the RNP structure. //
  //-----------------------------------------*/

m50->reference = *reference_input;
  m50->epoch     = *epoch_input;

  /*-----------------------------------------------------------//
  // Compute the time parameters in the TDT and UT1 time scales. //
  //-----------------------------------------------------------*/

  if(m50->epoch.time_scale == TDT)
   {
     julian_date_tdt = m50->epoch.julian_date;
     julian_date_ut1 = m50->epoch.julian_date
                         - m50->epoch.delta_time_tdt_ut1 / DAY;

     julian_date_midnight_tdt = m50->epoch.julian_date_midnight;
     julian_date_midnight_ut1 = m50->epoch.julian_date_midnight
                 - m50->epoch.delta_time_tdt_ut1_midnight / DAY;
   }
  else
   {
     julian_date_tdt = m50->epoch.julian_date
                         + m50->epoch.delta_time_tdt_ut1 / DAY;
     julian_date_ut1 = m50->epoch.julian_date;

     julian_date_midnight_tdt = m50->epoch.julian_date_midnight
                 + m50->epoch.delta_time_tdt_ut1_midnight / DAY;
     julian_date_midnight_ut1 = m50->epoch.julian_date_midnight;
   }
centuries_midnight_from_1900_ut1 =
                 (julian_date_midnight_ut1 - JULIAN_DATE_1900) /
                 JULIAN_CENTURY;
seconds_epoch_since_midnight_ut1 = (julian_date_ut1
                         - julian_date_midnight_ut1) * DAY;

  /*-----------------------------------------------------------//
  // Compute Greenwich mean sidereal time at midnight in seconds.//
  //-----------------------------------------------------------*/

TU = centuries_midnight_from_1900_ut1 ;
t_squared = TU * TU;

  temp = gmst_ut_relation_ddot * TU;

m50->greenwich_mean_siderial_time_midnight  = gmst_ut_relation_const
                         + gmst_ut_relation_dot * TU
                         + temp * TU;
```

```
  /*-------------------------------//
  // Ratio of siderial to solar time in //
  // revolutions per mean solar day.    //
  //-------------------------------*/

m50->ratio_siderial_to_solar_time = (DAY
      + (2 * temp + gmst_ut_relation_dot) / JULIAN_CENTURY) / DAY;

  /*-----------------------------------------------------//
  // Compute Greenwich mean sidereal time at epoch in seconds.//
  //-----------------------------------------------------*/

m50->greenwich_mean_siderial_time =
        m50->greenwich_mean_siderial_time_midnight
      + (m50->ratio_siderial_to_solar_time
        * seconds_epoch_since_midnight_ut1);

m50->greenwich_mean_siderial_time_midnight =
      angle_modulo ( m50->greenwich_mean_siderial_time_midnight
                  * RAD_PER_SEC_TIME );
m50->greenwich_mean_siderial_time =
      angle_modulo ( m50->greenwich_mean_siderial_time
                        * RAD_PER_SEC_TIME );

  /*-------------------------------------//
  // Convert the ratio of siderial to solar //
  // time from revolutions to radians.      //
  //-------------------------------------*/

m50->ratio_siderial_to_solar_time *= TWO_PI;

  /*-----------------------------------------------------------//
  // Set the time since the desired reference since the M50 epoch //
  // in centuries and the reference Greenwich mean siderial time. //
  //-----------------------------------------------------------*/

  if( m50->reference == AT_EPOCH )
   {
     centuries_ref_since_1900_tdt =
          (julian_date_tdt - JULIAN_DATE_1900)/JULIAN_CENTURY;
     centuries_ref_since_1950_trop =
          (julian_date_tdt - JULIAN_DATE_1950)/TROPICAL_CENTURY;
     greenwich_mean_siderial_time_ref =
                  m50->greenwich_mean_siderial_time;
   }

  else
   {
     centuries_ref_since_1900_tdt =
          (julian_date_midnight_tdt - JULIAN_DATE_1900)
```

```
        / JULIAN_CENTURY;
    centuries_ref_since_1950_trop =
        (julian_date_midnight_tdt - JULIAN_DATE_1950)
        / TROPICAL_CENTURY;
    greenwich_mean_siderial_time_ref =
        m50->greenwich_mean_siderial_time_midnight;
  }

  /*-------------------------------------------------------------//
  // Evaluate mean_angle from one to seven terms depending on the    //
  // value of poly_index after initialzing the mean_angle array to 0.//
  //-------------------------------------------------------------*/

  T = centuries_ref_since_1900_tdt;

  for(i = 0; i < 7; ++i)
  {
    mean_angle[i] = 0.0;
  }

t_squared = T * T;
t_cubed   = t_squared * T;
  for(i = 0; i < MEAN_ANGLE_INDEX; ++i)
  {
    mean_angle[i] = ( mean_angle_coef[i][0]
                    + mean_angle_coef[i][1] * T
                    + mean_angle_coef[i][2] * t_squared
                    + mean_angle_coef[i][3] * t_cubed )
                    * ARC_SEC;
  }
m50->mean_obliquity = mean_angle[0];

  /*-----------------------------------//
  // Compute the nutation in longitude.//
  //-----------------------------------*/

m50->nutation_in_longitude = 0.0;
  for(j = 0; j < mode_limits[0]; ++j)
  {

  /*-------------------------------------------------------------//
  // Since original ca, cb, longitude_lib_coef, inclination_lib_coef,//
  // node_lib_coef were in six character literal strings increment,  //
  // counter must be advanced by six each time through loop.         //
  //-------------------------------------------------------------*/

        m = j * 6;

        if(j < dimension[1][0])
        {
```

```
            coef = adot[j] * T + a[j];
          }

        else
          {
            coef = a[j];
          }

    argument = 0.0;
        for(l = 0; l <= data_increment[0]; ++l)
          {
            if(l != 0) ++m;

            interim_argument = ca[m];
            if(interim_argument != 0)
             {
                  if( interim_argument >= 4 )
                   {
                     interim_argument -= 7;
                   }
                  argument += interim_argument * mean_angle[l+1];
             }
          }
    m50->nutation_in_longitude += coef * sin(argument);
}

/*-----------------------------------//
// Compute the nutation of obliquity. //
//-----------------------------------*/

m50->nutation_of_obliquity = 0.0;
for(j = 0; j < mode_limits[1]; ++j)
 {

/*-------------------------------------------------------------------//
// Since original ca, cb, longitude_lib_coef, inclination_lib_coef,//
// node_lib_coef were in six character literal strings increment,  //
// counter must be advanced by six each time through loop.        //
//-------------------------------------------------------------------*/

   m = j * 6;

   if(j < dimension[1][1] )
    {
        coef = bdot[j] * T + b[j];
    }

   else
    {
        coef = b[j];
```

```
    }

    argument = 0.0;
    for(l = 0; l <= data_increment[l]; ++l)
     {
         if (l != 0) ++m;

         interim_argument = cb[m];
         if(interim_argument != 0)
          {
            if(interim_argument >= 4)
             {
                 interim_argument -= 7;
             }
            argument += interim_argument * mean_angle[l+1];
          }
     }
    m50->nutation_of_obliquity += coef * cos(argument);
  }


  /*-----------------------------------------------------------------//
  // Produce the true greenwich hour angle by adding the equation of //
  // the equinoxes to the mean hour angle calculated previously.     //
  //-----------------------------------------------------------------*/

m50->nutation_of_obliquity *= ARC_SEC;
  m50->true_obliquity = m50->mean_obliquity + m50->nutation_of_obliquity;

m50->equation_of_equinoxes = m50->nutation_in_longitude
                  * cos (m50->true_obliquity);
m50->nutation_in_longitude *= ARC_SEC;

m50->greenwich_app_siderial_time_ref = angle_modulo (
        greenwich_mean_siderial_time_ref
           + m50->equation_of_equinoxes * ARC_SEC );


  /*-----------------------------------------------------------------//
  // Compute the following precession angles:                        //
  //                                                                 //
  // zeta    Negative of the right ascension of the mean             //
  //           celestial pole of date (Z-MOD axis) refered           //
  //           to the mean equator and pole of interest.             //
  //                                                                 //
  // theta   The complement of the declination of the mean           //
  //           celestial pole of data refered to the mean equator    //
  //           and pole of interest.                                 //
  //                                                                 //
  // z       The negative of the supplement of the right             //
```

```
  //          ascension of the mean celestial pole of date        //
  //          refered to the mean equator and pole of interest.   //
  //--------------------------------------------------------------*/

t_squared = centuries_ref_since_1950_trop
          * centuries_ref_since_1950_trop;

  t_cubed   = t_squared * centuries_ref_since_1950_trop;

m50->zeta  = ( precession_coef[0] * centuries_ref_since_1950_trop
         + precession_coef[1] * t_squared
           + precession_coef[2] * t_cubed ) * ARC_SEC;

m50->theta = ( precession_coef[3] * centuries_ref_since_1950_trop
         + precession_coef[4] * t_squared
         + precession_coef[5] * t_cubed ) * ARC_SEC;

m50->z     = ( precession_coef[6] * centuries_ref_since_1950_trop
         + precession_coef[7] * t_squared
           + precession_coef[8] * t_cubed ) * ARC_SEC;


  /*-------------------------------//
  // Calculate the precession matrix.//
  //-------------------------------*/

precession_generator ( m50->zeta, m50->theta, m50->z, m50->precession);

  /*-----------------------------//
  // Calculate the nutation matrix.//
  //-----------------------------*/

nutation_generator ( m50->mean_obliquity,
                     m50->nutation_in_longitude,
                m50->true_obliquity,m50->nutation);

  /*-------------------------------------------------------------//
  // Calculate the m50_matrix by premultiplying the precession matrix //
  // by the nutation matrix, and then premultiplying that matrix with //
  // the rotation matrix.                                          //
  //-------------------------------------------------------------*/

  matrix_x_matrix ( m50->nutation,
                    m50->precession,
                    nutation_precession );

  rotation_generator ( m50->greenwich_app_siderial_time_ref,
                       m50->rotation);

  matrix_x_matrix ( m50->rotation, nutation_precession, m50->rnp);
```

```
}

/*****************************************************************************/
/* This is the function                                                      */
/* Input arguments are -                                                     */
/*                                                                           */
/* Return arguments are -                                                    */
/*****************************************************************************/

        void    precession_generator(zeta, theta, z, precession)

        double precession[4][4];
        double  zeta;           /* Negative of the right ascension of
                                   // the mean celestial pole of date
                                   // (Z-MOD axis) refered to the mean
                                   // equator and pole of interest. */

        double  theta;          /* The complement of the declination
                                   // of the mean celestial pole of data
                                   // (Z-MOD axis) refered to the mean
                                   // equator and pole of interest. */

        double  z;              /* The negative of the supplement of
                                   // the right ascension of the mean
                                   // celestial pole of date (Z-MOD axis)
                                   // refered to the mean equator and
                                   // pole of interest. */
{
        double  c_theta_c_z;
        double  c_theta_s_z;
        double  cos_theta;
        double  cos_z;
        double  cos_zeta;
        double  sin_theta;
        double  sin_z;
        double  sin_zeta;

        /*-------------------------------------------------------//
        // Compute the sines and cosines of the Z-MOD axis pole angles.//
        //-------------------------------------------------------*/

        cos_zeta = cos( zeta );
        sin_zeta = sin( zeta );

        cos_theta = cos( theta );
        sin_theta = sin( theta );

        cos_z = cos( z );
        sin_z = sin( z );
```

```
        c_theta_c_z = cos_theta * cos_z;
        c_theta_s_z = cos_theta * sin_z;

        /*-----------------------------//
        // Compute the precession matrix.//
        //-----------------------------*/

        precession[1][1] =   cos_zeta  * c_theta_c_z - sin_zeta * sin_z;
        precession[1][2] = - sin_zeta  * c_theta_c_z - cos_zeta * sin_z;
        precession[1][3] = - sin_theta * cos_z;

        precession[2][1] =   cos_zeta  * c_theta_s_z + sin_zeta * cos_z;
        precession[2][2] = - sin_zeta  * c_theta_s_z + cos_zeta * cos_z;
        precession[2][3] = - sin_theta * sin_z;

        precession[3][1] =   cos_zeta  * sin_theta;
        precession[3][2] = - sin_zeta  * sin_theta;
        precession[3][3] =   cos_theta;
  }
/*****************************************************************************/
/* This is the function                                                      */
/* Input arguments are -                                                     */
/*                                                                           */
/* Return arguments are -                                                    */
/*****************************************************************************/
        void     nutation_generator (mean_obliquity, nutation_in_longitude,
                                     true_obliquity, nutation)

        double mean_obliquity;
        double nutation_in_longitude;
        double true_obliquity;
        double nutation[4][4] ;
  {
        double  c_true_obliq_c_nut_long;
        double  cos_mean_obliq;
        double  cos_nutation_in_long;
        double  cos_true_obliq;
        double  s_true_obliq_c_nut_long;
        double  sin_mean_obliq;
        double  sin_nutation_in_long;
        double  sin_true_obliq;

        /*-------------------------------------------------//
        // Compute the sines and cosines of the nutation angles. //
        //-------------------------------------------------*/

        cos_true_obliq = cos(true_obliquity);
        sin_true_obliq = sin(true_obliquity);

        cos_nutation_in_long = cos(nutation_in_longitude);
```

```
        sin_nutation_in_long = sin(nutation_in_longitude);

        cos_mean_obliq = cos(mean_obliquity);
        sin_mean_obliq = sin(mean_obliquity);

        c_true_obliq_c_nut_long = cos_true_obliq * cos_nutation_in_long;
        s_true_obliq_c_nut_long = sin_true_obliq * cos_nutation_in_long;

        /*----------------------------//
        // Compute  the nutation matrix.//
        //----------------------------*/

        nutation[1][1] =   cos_nutation_in_long;
        nutation[1][2] = - cos_mean_obliq * sin_nutation_in_long;
        nutation[1][3] = - sin_mean_obliq * sin_nutation_in_long;

        nutation[2][1] =   cos_true_obliq * sin_nutation_in_long;
        nutation[2][2] =   cos_mean_obliq * c_true_obliq_c_nut_long
                         + sin_mean_obliq * sin_true_obliq;
        nutation[2][3] =   sin_mean_obliq * c_true_obliq_c_nut_long
                         - cos_mean_obliq * sin_true_obliq;

        nutation[3][1] =   sin_true_obliq * sin_nutation_in_long;
        nutation[3][2] =   cos_mean_obliq * s_true_obliq_c_nut_long
                         - sin_mean_obliq * cos_true_obliq;
        nutation[3][3] =   sin_mean_obliq * s_true_obliq_c_nut_long
                         + cos_mean_obliq * cos_true_obliq;
  }
/*****************************************************************************/
/* This is the function                                                      */
/* Input arguments are -                                                     */
/*                                                                           */
/* Return arguments are -                                                    */
/*****************************************************************************/
        void    rotation_generator(greenwich_app_siderial_time,rotation)
        double  greenwich_app_siderial_time,  rotation[4][4] ;
  {
        double  cos_greenwich_app_siderial_time;
        double  sin_greenwich_app_siderial_time;

        /*-----------------------------------------//
        // Compute the sine and cosine of the greewich //
        // apparent siderial time angle.           //
        //-----------------------------------------*/

        cos_greenwich_app_siderial_time = cos ( greenwich_app_siderial_time );
        sin_greenwich_app_siderial_time = sin ( greenwich_app_siderial_time );

        /*----------------------------//
        // Compute the rotation matrix. //
```

```
        //-------------------------*/

        rotation[1][1] = cos_greenwich_app_siderial_time;
        rotation[1][2] = sin_greenwich_app_siderial_time;
        rotation[1][3] = 0.0;

        rotation[2][1] = -sin_greenwich_app_siderial_time;
        rotation[2][2] =  cos_greenwich_app_siderial_time;
        rotation[2][3] =  0.0;

        rotation[3][1] = 0.0;
        rotation[3][2] = 0.0;
        rotation[3][3] = 1.0;
  }
/*****************************************************************************/
/*****************************************************************************/
        double angle_modulo( angle )
        double  angle;
  {
        int       revolutions;

        revolutions = (int) ( fabs( angle ) / TWO_PI );
        if( angle > TWO_PI )
         {
           return( angle - (double) revolutions * TWO_PI );
         }
        else if( angle < 0.0 )
            {
                return( angle + (double) ( revolutions + 1 ) * TWO_PI );
            }
           else
            {
                return ( angle );
            }
  }
/*****************************************************************************/
```

```
/*****************************************************************************
This is the Runge-Kutta fourth order integrator.
Input arguments are -
        tz = current time in seconds
        xz[8] = initial state array consisting of the following elements -
                xz[0] = not used
                xz[1] = position in x (km)
                xz[2] = position in y (km)
                xz[3] = position in z (km)
                xz[4] = velocity in x (km/s)
                xz[5] = velocity in y (km/s)
                xz[6] = velocity in z (km/s)
                xz[7] = magnitude of position vector (km)
        step =  time step size in seconds
Return arguments are -
        x[8] =  final state array consisting of the same elements as xz[8]
          with x[7] = rdot

Runge-Kutta is called by 'Cowell'
Runge-Kutta calls 'Derivatives'
*****************************************************************************/
#include <orbmech1.h>
#include "iers.h"
#include "rnpm50.h"

        void    Runge_Kutta_4(tz, xz, jd_midnight, m50, P, IERS, step, x)
        double  tz;                                     /*current time */
        double  xz[8];                                  /*initial state */
        double  jd_midnight;                            /*julian date of midnight */
        double  step;                                   /*time step size */
        double  x[8];                                   /*integrated state */
      Perturbations *P ;
      struct  IERS_DATA *IERS ;
      RNP     *m50 ;
  {
        int     i ;                             /*loop counter */
        double  fz[8];
        double  x1[8];
        double  f1[8];
        double  x2[8];
        double  f2[8];
        double  x3[8];
        double  f3[8];
        double  t1, t2, t3 ;

        Derivatives(m50, P, IERS, jd_midnight, tz, xz, fz) ;
        t1 = tz + 0.4*step ;
        for(i=1 ; i<=7 ; i++)
         {
           x1[i] = xz[i] + 0.4*step*fz[i] ;
```

```
    }

    Derivatives(m50, P, IERS, jd_midnight, t1, x1, f1) ;
    t2 = tz + 0.6*step ;
    for(i=1 ; i<=7 ; i++)
    {
      x2[i] = xz[i] + step*(-0.15*fz[i] + 0.75*f1[i]) ;
    }

    Derivatives(m50, P, IERS, jd_midnight, t2, x2, f2) ;
    t3 = tz + step ;
    for(i=1 ; i<=7 ; i++)
    {
      x3[i] = xz[i] + step*(19.0*fz[i] - 15.0*f1[i] +
              40.0*f2[i])/44.0 ;
    }

    Derivatives(m50, P, IERS, jd_midnight, t3, x3, f3) ;
    for(i=1 ; i<=7 ; i++)
    {
      x[i] = xz[i] + step*(55.0*(fz[i] + f3[i]) +
             125.0*(f1[i] + f2[i]))/360.0 ;
    }
}
```

```
#ifndef RNP_M50_DATA_INCLUDED
#define RNP_M50_DATA_INCLUDED


/*
 ************************************************************************
 *
 *     Data for the Woolard Nutation Model
 *
 ************************************************************************
 */

static double   a[] = {
                        -17.2327 , -1.2729 ,  0.2088 ,
                         -0.2037 ,  0.1261 ,  0.0675 , -0.0497 ,
                    -0.0342 , -0.0261 ,  0.0214 , -0.0149 ,  0.0124 ,
                     0.0114 ,  0.0060 ,  0.0058 , -0.0057 , -0.0052 ,
                     0.0045 ,  0.0045 , -0.0044 , -0.0032 ,  0.0028 ,
                     0.0026 , -0.0026 ,  0.0025 , -0.0021 ,  0.0019 ,
                     0.0016 , -0.0015 , -0.0015 ,  0.0014 , -0.0013 ,
                     0.0010 , -0.0010 , -0.0009 , -0.0007 ,  0.0007 ,
                     0.0006 , -0.0006 , -0.0006 , -0.0006 ,  0.0006 ,
                    -0.0005 , -0.0005 , -0.0005 , -0.0005 ,  0.0005 ,
                     0.0004 , -0.0004 , -0.0004 , -0.0004 ,  0.0004 ,
                     0.0004 , -0.0004 ,  0.0003 , -0.0003 , -0.0003 ,
                     0.0003 , -0.0003 , -0.0003 , -0.0002 , -0.0002 ,
                    -0.0002 ,  0.0002 , -0.0002 , -0.0002 , -0.0002 ,
                     0.0002 , -0.0002 };

static double   adot[] = {
                         -0.01737 , -0.00013 ,  0.00002 , -0.00002 ,
                         -0.00031 ,  0.00001 ,  0.00012 , -0.00004 ,
                   0.0    , -0.00005 ,  0.0    ,  0.00001 ,  0.0    ,
                   0.0    ,  0.0    ,  0.0    ,  0.0    ,  0.0    ,
                   0.0    ,  0.0    ,  0.0    ,  0.0    ,  0.0    ,
                   0.0    ,  0.0    ,  0.0    ,  0.0    , -0.00001 ,
                   0.00001 };

static double   b[] = {
                          9.2100 ,  0.5522 , -0.0904 ,  0.0884 ,
                          0.0216 ,  0.0183 ,  0.0113 , -0.0093 ,
                    -0.0066 , -0.0050 , -0.0031 ,  0.0030 , -0.0024 ,
                     0.0023 ,  0.0022 ,  0.0014 , -0.0011 ,  0.0011 ,
                    -0.0010 ,  0.0008 ,  0.0007 , -0.0007 ,  0.0007 ,
                     0.0005 ,  0.0005 ,  0.0003 ,  0.0003 , -0.0003 ,
                     0.0003 ,  0.0003 ,  0.0003 ,  0.0003 ,  0.0003 ,
                    -0.0003 ,  0.0002 ,  0.0002 , -0.0002 , -0.0002 ,
                    -0.0002 ,  0.0002 };

static double   bdot[] = {
```

```
                0.00091 , -0.00029 , 0.00004 , -0.00005 ,
                -0.00006 , 0.0     , -0.00001 , 0.00003 };

static int      ca[] = {
                    0,0,0,0,1,0,0,0,2,5,2,
                    0,0,0,0,0,2,0,0,0,2,0,
                2,0,0,1,0,0,0,0,1,0,0,0,0,
                0,0,1,2,5,2,0,0,0,2,0,1,0,
                1,0,2,0,2,0,0,6,2,5,2,0,1,
                0,0,5,0,0,0,0,2,5,1,0,6,0,
                2,0,2,0,0,0,0,2,0,0,1,0,0,
                0,1,0,6,0,0,0,1,0,6,0,2,2,
                2,0,5,0,2,0,1,0,2,0,0,5,0,
                0,1,0,2,0,1,0,0,0,2,2,2,0,
                2,0,0,0,0,0,1,0,2,5,2,0,2,
                0,2,0,2,0,0,0,2,0,0,0,0,0,
                2,5,0,0,6,0,2,0,1,0,0,2,0,
                0,0,0,0,2,2,5,2,0,0,1,0,0,
                1,0,6,0,0,2,1,0,1,0,0,5,1,
                0,2,0,5,0,0,0,0,6,0,0,1,0,
                6,0,2,2,1,0,1,1,0,5,0,0,0,
                1,2,0,2,0,1,0,0,2,0,0,0,0,
                0,2,1,0,0,6,2,0,2,0,1,0,2,
                2,2,0,2,0,2,5,2,0,5,0,0,2,
                1,0,0,6,2,5,1,0,0,0,0,5,1,
                0,0,0,2,2,1,0,1,0,2,5,1,0,
                2,0,0,5,1,0,0,5,2,5,1,0,0,
                0,0,1,0,0,0,1,0,5,0,0,1,6,
                0,0,0,0,1,0,5,0,0,0,2,0,2,
                0,1,0,0,1,2,5,1,0,1,0,0,6,
                0,0,5,0,2,0,2,0,1,0,2,0,0,
                0,1,1,0,0,0,0,1,6,2,0,2,0,
                1,6,0,6,0,0,5,0,0,0,1,0,6,
                0,2,5,1,0,2,0,0,0,1,0,6,6,
                2,2,2,0,0,6,2,2,2,0,1,0,0,
                0,2,0,1,1,2,0,2,0,3,0,2,0,
                2,0};

static int      cb[] = {
                    0,0,0,0,1,0,0,0,2,5,2,
                    0,0,0,0,0,2,0,0,0,2,0,
                2,0,0,1,2,5,2,0,0,0,2,0,1,
                0,1,0,2,0,2,0,0,6,2,5,2,0,
                0,0,2,5,1,0,6,0,2,0,2,0,1,
                0,0,0,1,0,6,0,0,0,1,0,5,0,
                2,0,1,0,1,0,2,0,1,0,6,0,2,
                2,2,0,0,0,2,2,2,0,1,0,2,5,
                2,0,2,0,2,0,2,0,6,0,2,0,1,
                0,0,1,0,0,1,0,0,2,2,5,2,0,
                6,0,0,2,1,0,1,0,0,5,1,0,0,
```

```
        6 , 0 , 0 , 1 , 0 , 6 , 0 , 2 , 2 , 1 , 0 , 5 , 0 ,
        0 , 2 , 1 , 0 , 0 , 6 , 2 , 5 , 1 , 0 , 0 , 1 , 2 ,
        0 , 2 , 0 , 0 , 0 , 0 , 2 , 1 , 0 , 0 , 6 , 2 , 0 ,
        2 , 0 , 1 , 0 , 2 , 2 , 2 , 0 , 0 , 0 , 0 , 5 , 1 ,
        0 , 0 , 0 , 2 , 2 , 1 , 0 , 1 , 0 , 2 , 5 , 1 , 0 ,
        0 , 5 , 2 , 5 , 1 , 0 , 5 , 0 , 2 , 0 , 2 , 0 , 2 ,
        0 , 0 , 5 , 1 , 0 , 0 , 1 , 2 , 5 , 1 , 0 , 2 , 0 ,
        2 , 5 , 2 , 0 , 2 , 0 , 2 , 0 , 1 , 0 };


static int      dimension[2][2] = {
                        { 69 , 40 },    /* dimensions ca & cb */
                        { 29 ,  8 } };  /* dimensions adot & bdo: */



    static int      mode_limits[2] = {
                        69 , 40 };      /* Execution limits for
                                           operation mode set to 1.
                                           Provides the appropriate
                                           number of data points from
                                           the ca & cb data arrays from
                                           the nutation 1 mode */

    static int      data_increment[2] = {
                        4 , 4 };        /* Provides the number of
                                           coefficient data points to
                                           be used in the calculation
                                           of the two nutation angle */



/*
 ************************************************************************
 *
 *      Coefficients of the polynomials for mean angles
 *
 ************************************************************************
 */


static double   mean_angle_coef[7][4] = {

/*
 *      Coefficients for the mean obliquity of the celestial equator
 *      equivalent to epsilon bar (seconds of arc )
 */

    { 84428.26    ,      -46.845 ,  -0.0059 ,  0.00181 } ,
```

```
/*
 *    Coefficients for the mean anomaly of the moon equivalent to
 *    lower case l ( seconds of arc )
 */

      { 1065976.59  , 1717915856.79  ,  33.09   ,  0.0518  } ,


/*
 *    Coefficients for the mean anomaly of the sun equivalent to
 *    lower case l prime ( seconds of arc )
 */

      { 1290513.00  ,  129596579.10  ,  -0.54   , -0.0120  } ,


/*
 *    Coefficients for the geocentric angular distance from the ascending
 *    node of the lunar orbit to the moon equivalent to F (seconds of arc )
 */

      {  40503.20  , 1739527290.54  , -11.56   , -0.0012  } ,


/*
 *    Coefficients for the geocentric angular separation of the sun and moon
 *    equivalent to D (seconds of arc )
 */

      { 1262654.95  , 1602961611.18  ,  -5.17   ,  0.0068  } ,


/*
 *    Coeficients for the longitude of the mean ascending node of the lunar
 *    orbit measured in the elitic plane from the mean equinox of date
 *    equivalent to omega ( seconds of arc )
 */

      {  933059.79  ,  -6962911.23  ,   7.48   ,  0.0080  } ,


/*
 *    Coefficients for the polynomial that produces the angular argument
 *    of Koziel's free libration term in longitude
 */

      {  696385.257 ,   -478333.469 ,   0.0    ,  0.0    } };


/*
```

```
 *      Set the mean angle index such that the free lirbation polynomial goes
 *      unused.
 */

static int      MEAN_ANGLE_INDEX = { 6 };


/*
 *      Coefficients of precession polyomials
 */

static double   precession_coef[] = {
                    2304.951665 , 0.302165 , 0.0180 ,
                    2004.258258 , -0.426885 , -0.0418 ,
                    2304.951615 , 1.095195 , 0.01832 };


/*
 *      Set the parameters that relate Greenwich mean siderial time
 *      to universal time.
 */

static double   gmst_ut_relation_const  = { 23925.836 };
static double   gmst_ut_relation_dot    = { 8640184.542 };
static double   gmst_ut_relation_ddot   = { 0.0929 };


#endif  /* Endif for RNP_M50_DATA_INCLUDED */
```

```
/*****************************************************************************
 built by Mike Fraietta Jan 1993

 Program modified by Lee Barker

 Function declarations contained in "util.h"

  date          mod #  description
 23 Feb 93      1      addition of seconds_in_day,jd_time functions

 24 Feb 93      2      addition of doy_utc_to_julian_date function
*****************************************************************************/
#include <math.h>

#include "util.h"

#define jd_jan_6_1980_midnight 2444243.5          /*GPS clock start */
                                                  /*julian date */
/*****************************************************************************
 This is the function 'julian_date'
 Input arguments are -
        year    =       julian calender year, ie 1993
        month   =       month number (1-12)
        day     =       day of month (1-31)
        hours   =       hours since midnight (military time) in UTC
        minutes =       minutes past hour
        seconds =       seconds past minute
 Return arguments are -
         jd(double)=     julian date
*****************************************************************************/

        double  julian_date(year,month,day,hours,minutes,seconds)
        int     year,month,day,hours,minutes ;
        double  seconds ;
  {
        double  jd_noon, jd ;

        jd_noon=367.0*year -
        floor(7.0*(year + floor((month+9)/12))/4.0) +
        floor(275.0*month/9.0) + (double)day + 1721014.0 ;

        jd = (double)jd_noon - 0.5 +
                hours/24.0 +
                minutes/1440.0 +
                seconds/86400.0 ;
        return jd ;
  }
```

```
/*****************************************************************************
This is the function 'doy_utc_to_julian_date'
 Input arguments are -
        doy   = day of year (1-365)
        hours = hours since midnight (military time) in UTC
        minutes   = minutes past hour
        seconds   = seconds past minute
 Return arguments are -
    julian_date(double)= julian date

 Year is assumed to be 1993. For different year, change year in call
 to 'julian_date'.
*****************************************************************************/
#ifdef GIGO
        double  doy_utc_to_julian_date(doy,hour,min,sec)
        doub[illegible] ;
        in[illegible] y,hour,min;

  {
        double  seconds_since_midnight;
        double  julian_date_of_year;
        double  jd;

        seconds_since_midnight = 3600.0*hour + 60.0*min + sec;

        julian_date_of_year = julian_date(1993, 1, 1, 0, 0, 0.0);

        jd = julian_date_of_year +
                        doy + seconds_since_midnight/86400.0;

        return(jd);
  }
#endif GIGO

/*****************************************************************************
 This is the function 'magnitude'
 Input arguments are -
        a = 4x1 vector where the zero element is not used
           and a 3x1 vector is held in the respective elements
 Return arguments are -
     (double) the magnitude of the vector 'a'
*****************************************************************************/

        double  magnitude(a)
        double  a[4] ;
  {
        double  mag;
        mag=sqrt(a[1]*a[1]+a[2]*a[2]+a[3]*a[3]);
        return (mag);
  }
```

```
/*****************************************************************************
 This is the function 'dot'
 Input arguments are -
        a = 4x1 vector where the zero element is not used
           and a 3x1 vector is held in the respective elements
        b = 4x1 vector where the zero element is not used
           and a 3x1 vector is held in the respective elements
 Return arguments are -
        (double) the dot product of 'a' and 'b'
*****************************************************************************/

        double dot(a,b)
        double a[],b[] ;
  {
        return (a[1]*b[1]+a[2]*b[2]+a[3]*b[3]) ;
  }
/*****************************************************************************
 This is the function 'cross'
 Input arguments are -
        a = 4x1 vector where the zero element is not used
           and a 3x1 vector is held in the respective elements
        b = 4x1 vector where the zero element is not used
           and a 3x1 vector is held in the respective elements
 Return arguments are -
        c = 4x1 vector where the zero elements are not used
           and a 3x1 vector of the cross product is held in
           the respective elements
*****************************************************************************/

        void    cross(a,b,c)
        double a[],b[],c[] ;
  {
        c[1] = a[2]*b[3] - a[3]*b[2] ;
        c[2] = a[3]*b[1] - a[1]*b[3] ;
        c[3] = a[1]*b[2] - a[2]*b[1] ;
  }
/*****************************************************************************
 This is the function 'unit'
 Input arguments are -
        a = 4x1 vector where the zero element is not used and
           a 3x1 vector is held in the respective elements
 Return arguments are -
        a = 4x1 vector where the zero element is not used and
           the normalized 3x3 matrix is held in the respective elements
*****************************************************************************/

        void    unit(a,b)
        double  a[],b[] ;
  {
        double a_mag_i  ;
```

```
        a_mag_i = 1.0/sqrt(dot(a,a)) ;
        b[1] = a[1]*a_mag_i ;
        b[2] = a[2]*a_mag_i ;
        b[3] = a[3]*a_mag_i ;
  }
/*****************************************************************************
 This is the function  factorial_ratio
 Input arguments are -
        n,k = integers
 Return arguments are -
     ans(double) = the factorial ratio of n to k
*****************************************************************************/

        double  factorial_ratio(n,k)
        int     n,k;
  {
        double  ans;
        int     i,stop,start;

        if(n == k) return(1.0);
        if (n > k)
         {
           start = k+1;
           stop  = n;
         }
        else
         {
           start = n+1;
           stop  = k;
         }
        ans = (double)start;
        for(i = start+1; i <= stop; i++) ans = ans*(double)i;
        if(k > n) ans = 1.0/ans;

        return(ans);
  }
/*****************************************************************************
 This is the function 'sqr'
 Input arguments are -
        a = (double)
 Return arguments are -
     (double) the square of 'a'
*****************************************************************************/

        double  sqr(a)
        double  a ;
  {
        return a*a ;
  }
```

```
/*****************************************************************************
 This is the function 'frac'
 Input arguments are -
     x = (double)
 Return arguments are -
     (double) the fractional part of 'x'
*****************************************************************************/

        double  frac(x)
        double  x ;
  {
        double  floor() ;

        return(x-(double)floor(x)) ;
  }
/*****************************************************************************
 This is the function *jdtocd
*****************************************************************************/

        void    jdtocd(jd,montho,dayo,yearo,houro,minuteo,secondo)
        double  jd ;
        double  *secondo ;
        int     *montho,*dayo,*yearo,*houro,*minuteo ;
  {
        double  floor(),ceil(),frac() ;
        double  jdtemp,f,alpha,z,a,b,c,d,e ;
        double  day,month,year,hour,minute,second ;

        jdtemp=jd+0.5 ;
        f=frac(jdtemp) ;
        z=jdtemp-f ;

        if(z < 2299161.0)
         {
           a=z ;
         }
        else
         {
           alpha=(double)floor((z-1867216.25)/36524.25) ;
           a=z+1.0+alpha-(double)floor(alpha/4.0) ;
         }

        b=a+1524.0 ;
        c=(double)floor((b-122.1)/365.25) ;
        d=365.25*c-frac(365.25*c) ;
        e=(double)floor((b-d)/30.6001) ;

        day=b-d-(30.6001*e-frac(30.6001*e))+f ;
        *dayo=floor(day) ;
```

```
        if (e < 13.5)  month=e-1.0 ; else month=e-13.0 ;
        if (month > 2.5)  year=c-4716.0 ; else year=c-4715.0 ;
        *montho=floor(month) ;
        *yearo=floor(year) ;

        hour=frac(day)*24.0 ;
        minute=frac(hour)*60.0 ; second=frac(minute)*60.0 ;
        *houro=(int)floor(hour) ;
        *minuteo=(int)floor(minute) ;
        *secondo=second ;

        if (*secondo >= 60.0)
         {
           *secondo=0.0 ; *minuteo = *minuteo+1 ;
         }
        if (*minuteo >= 60)
         {
           *minuteo=0 ; *houro = *houro+1 ;
         }
  }
/*****************************************************************************
 This is the function matrix_x_matrix
 Input arguments are -
        a = 4x4 matrix where the zero elements are not used
           and a 3x3 matrix is held in the respective elements
        b = 4x4 matrix where the zero elements are not used
           and a 3x3 matrix is held in the respective elements
 Return arguments are -
        c = 4x4 matrix where the zero elements are not used and
           the product 3x3 matrix is held in the respective elements
*****************************************************************************/

        void    matrix_x_matrix(a,b,c)
        double  a[4][4],b[4][4],c[4][4] ;
  {
        int     i,j,k ;

        for(i=1 ; i<=3 ; i++)
        for(k=1 ; k<=3 ; k++)
         {
           c[i][k]=0.0 ;
           for(j=1 ; j<=3 ; j++) c[i][k]=c[i][k] + a[i][j]*b[j][k] ;
         }
  }
/*****************************************************************************
 This is the function matrix_x_vector
 Input arguments are -
        a = 4x4 matrix where the zero elements are not used
           and a 3x3 matrix is held in the respective elements
        v = array containing a 3x1 vector where the zero element
```

```
          is not used
 Return arguments are -
        v_out = array containing a 3x1 vector product of 'a'
                and 'v' where the zero element is not used
*****************************************************************************/

        void    matrix_x_vector(a,v,v_out)
        double  a[4][4],v[4],v_out[4] ;
  {
        int     i,j ;

        for(i=1 ; i<=3 ; i++)
        {
          v_out[i]=0.0 ;
          for(j=1 ; j<=3 ; j++) v_out[i] = v_out[i] + a[i][j]*v[j];
        }
  }
/*****************************************************************************
 This is the function matrixTranspose_x_vector
 Input arguments are -
        a = 4x4 matrix where the zero elements are not used
          and a 3x3 matrix is held in the respective elements
        v = array containing a 3x1 vector where the zero element
          is not used
 Return arguments are -
        v_out = array containing a 3x1 vector product of 'a'transpose
                and 'v' where the zero element is not used
*****************************************************************************/

        void    matrixTranspose_x_vector(a,v,v_out)
        double  a[4][4],v[4],v_out[4] ;
  {
        int     i,j ;

        for(i=1 ; i<=3 ; i++)
        {
          v_out[i]=0.0 ;
          for(j=1 ; j<=3 ; j++) v_out[i] = v_out[i] + a[j][i]*v[j] ;
        }
  }
/*****************************************************************************/
 This is the function jd_time
 Input arguments are -
        gps_week_number = GPS week number starting at week 0 on 6 Jan 1980
        gps_seconds_in_week = seconds starting Sunday morning UTC
 Return arguments are -
     (jd) = julian date
*****************************************************************************/

        double  jd_time( gps_week_number, gps_seconds_in_week)
```

```
        double  gps_week_number,gps_seconds_in_week;
    {
        double  jd;

        jd = jd_jan_6_1980_midnight + 7*gps_week_number +
                gps_seconds_in_week/86400;

        return  (jd);

    }

/*****************************************************************************
 This is the function 'seconds_in_day'
 Input arguments are -
        gps_seconds_in_week  =  seconds starting Sunday morning UTC
 Return arguments are -
        (seconds) = seconds in day since midnight
*****************************************************************************/

        double  seconds_in_day(gps_seconds_in_week)
        double  gps_seconds_in_week;
    {
        double  seconds;
        seconds = fmod(gps_seconds_in_week,86400);
        return  (seconds);
    }
/*****************************************************************************/
```

```
/*
#ifndef UTIL_INCLUDED
#define UTIL_INCLUDED
*/
void    jdtocd() ;

double  julian_date() ;
double  dot() ;
double  sqr() ;
double  frac() ;
double  magnitude() ;
double  factorial_ratio() ;
double  jd_time();
double  seconds_in_day();
double  doy_utc_to_julian_date();

void    matrixTranspose_x_vector();
void    matrix_x_vector();
void    matrix_x_matrix();
void    unit();
void    cross();
void    rotation_generator();
void    nutation_generator ();
void    precession_generator();
void    rnp_m50 ();
void    instantiate_gotpot();
void    gem_9_model_data();
void    get_gravity_data();
void    get_rotation_angles();
void    gotpot();
void    Derivatives();
void    get_inertial_to_wgs84_matrix();
void    compute_a_matrix();
void    Runge_Kutta_4();
void    cowell();
void    bgprop();


/*
#endif
*/
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>

#include <edit.h>
#include <keys.h>

/*=============================================================
============================
 *-----------------------------------------------------------*/
#define YES 1
#define NO  0

enum edit_types { EDIT_alpha, EDIT_numb, EDIT_text };

#define WORKSIZE 80

static char work[WORKSIZE+1];


/*=============================================================
============================
 *-----------------------------------------------------------*/
static int edit(char *label, enum edit_type type)
{
   char *next = work + strlen(work);
   printf("%s%s", label, work);

   for (;;)
   {
        int key = key_get();

        if (key == KEY_esc)  return NO;
        if (key == KEY_cr)
        {
           putch(key);  putch(KEY_lf);
           return YES;
        }
        if (key == KEY_bs)
        {
           if (next == work)  continue;
           *(--next) = 0;
           putch('\b');  putch(' ');  putch('\b');
           continue;
        }
```

```
        if (! isascii(key))  continue;

        switch (type)
        {
           case EDIT_alpha:
                if (!isalpha(key))  continue;
                break;
           case EDIT_numb:
                if (!isdigit(key)  &&  key != '.'  &&  key != '-')  continue;
                break;
           case EDIT_text:
                if (!isgraph(key))  continue;
                break;
        }
        *next++ = key;
        putch(key);
    }
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
char *edit_text(char *label, char *initial)
{
    int size = strlen(initial);

    if (size > WORKSIZE)  initial[WORKSIZE] = 0;
    strcpy(work, initial);

    if (edit(label, EDIT_text) == NO  ||  work[0] == 0)  strcpy(work, initial);
    return work;
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
int edit_integer(char *label, int initial)
{
    sprintf(work, "%d", initial);

    return (edit(label, EDIT_numb) == NO  ||  work[0] == 0 ? initial : atoi(work));
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
long edit_integer_long(char *label, long initial)
{
    sprintf(work, "%ld", initial);
```

```
    return (edit(label, EDIT_numb) == NO  ||  work[0] == 0 ? initial : atol(work));
}
```

```
/*=============================================================
==========================
 *-----------------------------------------------------------*/
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>

#include <edit_rt.h>
#include <keys.h>
#include <times.h>

/*=============================================================
==========================
 *-----------------------------------------------------------*/
int edit_field_get_int(char *work, void *data)
{
   sprintf(work, "%d", *(int *)data);
   return -1;
}

int edit_field_put_int(char *work, void *data)
{
   *(int *)data = atoi(work);
   return YES;
}

int edit_field_key_int(Edit_field_info *info, int key)
{
   if ((key >= '0' && key <= '9') || key == '-')
   {
        edit_field_insert(info, key);
        return YES;
   }
   return edit_field_key_handler(info, key);
}

/*=============================================================
==========================
 *-----------------------------------------------------------*/
int edit_field_get_dbl(char *work, void *data)
{
   sprintf(work, "%.12lg", *(double *)data);
   return -1;
}

int edit_field_put_dbl(char *work, void *data)
{
   *(double *)data = atof(work);
   return YES;
}
```

```
int edit_field_key_dbl(Edit_field_info *info, int key)
{
   if ((key >= '0' && key <= '9')
    || key == '-' || key == '.' || key == 'e' || key == 'E')
   {
        edit_field_insert(info, key);
        return YES;
   }
   return edit_field_key_handler(info, key);
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
int edit_field_get_yesno(char *work, void *data)
{
   work[0] = (*(int *)data ? 'Y' : 'N');
   work[1] = 0;
   return 0;
}

int edit_field_put_yesno(char *work, void *data)
{
   *(int *)data = (work[0] == 'Y' ? YES : NO);
   return YES;
}

int edit_field_key_yesno(Edit_field_info *info, int key)
{
   int x, y;

   switch (key)
   {
        case 'Y': case 'y': case 'N': case 'n':
           if (key > 'a') key -= 'a'-'A';
           if (info->work[0] != key) info->changed = YES;
           x = wherex() - info->cursor_offset; y = wherey();
           info->cursor_offset = 0;
           gotoxy(x, y); putch(info->work[0] = key); gotoxy(x, y);
           return YES;

        case KEY_esc: return (info->changed ? 2 : 0);

        default: return NO;
   }
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
```

```
int edit_field_get_dhms1(char *work, void *data)
{
   double temp = *(double *)data + 86400.0;
   return edit_field_get_dhms0(work, &temp);
}

int edit_field_put_dhms1(char *work, void *data)
{
   edit_field_put_dhms0(work, data);
   if ((*(double *)data -= 86400.0) < 0.0)  *(double *)data = 0.0;
   return YES;
}

/*=============================================================
==========================
 *-------------------- ----------------------------------------*/
int edit_field_get_dhms0(char *work, void *data)
{
   time_dbl_to_string((double *)data, work);
   return 0;
}

int edit_field_put_dhms0(char *work, void *data)
{
   Time_dhms dhms;

   work[3] =  work[6] =  work[9] =  work[12] =  0;
   dhms.days  = atoi(work);
   dhms.hours = atoi(work+4);
   dhms.mins  = atoi(work+7);
   dhms.secs  = atoi(work+10);
   dhms.msecs = atoi(work+13);

   time_dhms_to_dbl(&dhms, (double *)data);
   return YES;
}

int edit_field_key_dhms(Edit_field_info *info, int key)
{
   char *where;
   int rc, key_max;

   /*-------------------------------------------------------------
    */
   if (key >= '0'  &&  key <= '9')
   {
        where = info->work + info->cursor_offset;

        if (info->cursor_offset == 4)                                  key_max = '2';
        else if (info->cursor_offset == 7  ||  info->cursor_offset == 10)  key_max = '5';
```

```
        else if (info->cursor_offset == 5  &&  where[-1] == '2')        key_max = '3';
        else                                                            key_max = '9';
        if (key > key_max)  return YES;

        if (*where != key)  info->changed = YES;
        putch(*where = key);

        if (info->cursor_offset == 4  &&  where[1] > '3')
        {
           info->cursor_offset++;  where++;
           putch(*where = '3');
        }

        if (info->cursor_offset < 15)
        {
           info->cursor_offset++;  where++;
           if (*where == '/'  ||  *where == ':'  ||  *where == '.')
                edit_field_key_handler(info, KEY_right);
        }
        else gotoxy(wherex() - 1, wherey());

        return YES;
}

/*---------------------------------------------------------------------
 */
switch (key)
{
        /*---------------------------------------------------------------
         *  I can't allow backspace/delete, they have no meaning for the fixed
         * position time string.
         */
        case KEY_bs:
        case KEY_del:  return YES;

        /*---------------------------------------------------------------
         *  I don't want the default key handler moving the cursor beyond the
         * end of the time string.
         */
        case KEY_end:
           gotoxy(wherex() + info->size - info->cursor_offset - 1, wherey());
           info->cursor_offset = info->size - 1;
           return YES;

        case KEY_right:
           if (info->cursor_offset == 15)  return YES;
}

/*---------------------------------------------------------------------
 *  I have to add an extra left/right keystroke when the cursor lands on a
```

```
     * separator character (/:).
     */
    if ((rc = edit_field_key_handler(info, key)) == 1)
    {
        where = info->work + info->cursor_offset;
        if (*where == '/' || *where == ':' || *where == '.')
            edit_field_key_handler(info, key);
    }
    return rc;
}
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>

#include <edit_rt.h>
#include <keys.h>

/*=============================================================
============================
 *-----------------------------------------------------------*/
static char work[80];

static Edit_screen *prev_screen;

/*=============================================================
============================
 *  I force the field to regenerate its string, then I draw the string and
 * position the cursor.
 *-----------------------------------------------------------*/
static void field_reset(Edit_field_info *info, Edit_field *field, int preclear)
{
   int first_offset;

   if (preclear  &&  info->size)
   {
        memset(work, ' ', info->size);
        gotoxy(field->col, field->row);
        cputs(work);
   }

   first_offset = (*field->get)(work, field->data);
   gotoxy(field->col, field->row);
   cputs(work);

   info->size = strlen(work);

   if (first_offset < 0  || (info->cursor_offset = first_offset) > info->size)
        info->cursor_offset = info->size;
   gotoxy(field->col + info->cursor_offset, field->row);

   info->changed = NO;
}

/*=============================================================
============================
 *-----------------------------------------------------------*/
```

```
void edit_screen_init(Edit_screen *screen)
{
    int count;
    Edit_label *label;
    Edit_field *field;

    /*-----------------------------------------------------------------
     *  Clean up the previous screen, if it's still up.
     */
    if (prev_screen != NULL && prev_screen->post != NULL)
        (*prev_screen->post)(prev_screen);

    /*-----------------------------------------------------------------
     *  Show all labels and fields.
     */
    clrscr();
    prev_screen = screen;
    screen->changed = NO;
    if (screen->pre != NULL) (*screen->pre)(screen);

    for (label = screen->labels, count = screen->label_count; count > 0; label++, count--)
    {
        gotoxy(label->col, label->row);
        cputs(label->string);
    }

    for (field = screen->fields, count = screen->field_count; count > 0; field++, count--)
    {
        (*field->get)(work, field->data);
        gotoxy(field->col, field->row);
        cputs(work);
    }

    /*-----------------------------------------------------------------
     *  Convert first field to text, save its size.
     */
    screen->current_field.info.work = work;
    field_reset(&screen->current_field.info, screen->current_field.ptr = screen->fields, NO);
}

/*=================================================================
==========================
 *  I'm called by some piece of code that wants to take over the screen.
 *-----------------------------------------------------------------*/
void edit_screen_stop()
{
    Edit_screen *screen = prev_screen;

    if (screen == NULL) return;
    prev_screen = NULL;
```

```
    if (screen->current_field.info.changed)
    {
        Edit_field *field = screen->current_field.ptr;

        if ((*field->put)(work, field->data) == NO)
            field_reset(&screen->current_field.info, field, YES);
        else screen->changed = YES;
    }
    if (screen->post != NULL) (*screen->post)(screen);
}

/*=============================================================
==========================
 *  My return values are:
 *      0  I didn't use the key
 *      1  I used the key
 *      2  I'm done
 *-------------------------------------------------------------*/
int edit_screen_key_handler(Edit_screen *screen, int key)
{
    Edit_field *field = screen->current_field.ptr;
    Edit_field *last_field;
    int rc;

    /*---------------------------------------------------------
     *  Call the field's key handler, check its return value for:
     *    0  It didn't use the key, I will check the key myself
     *    1  It used the key, I can return
     *    2  I must reset the field to original condition
     */
    rc = (*field->key)(&screen->current_field.info, key);

    if (rc > 0)
    {
        if (rc == 2) field_reset(&screen->current_field.info, field, YES);
        return 1;
    }

    /*---------------------------------------------------------
     *  The field didn't use the key, I'll check the key.
     */
    last_field = screen->fields + (screen->field_count - 1);

    switch (key)
    {
        case KEY_esc: rc = 2; break;

        case KEY_up:
        case KEY_backtab:
            if (--screen->current_field.ptr < screen->fields)
```

```
                screen->current_field.ptr = last_field;
            rc = 1;
            break;

        case KEY_down:
        case KEY_tab:
        case KEY_cr:
            if (++screen->current_field.ptr > last_field)
                screen->current_field.ptr = screen->fields;
            rc = 1;
            break;

        default: return 0;
    }

    /*-----------------------------------------------------------------
     *  Check if the field was changed.  I only reach this code after I've
     * picked a new field, or I'm done with this screen.
     *  Then move to and reshow the new current field.
     */
    if (screen->current_field.info.changed)
    {
        if ((*field->put)(work, field->data) == NO)
            field_reset(&screen->current_field.info, field, YES);
        else screen->changed = YES;
    }

    if (rc == 2)
    {
        if (screen->post != NULL) (*screen->post)(screen);
        prev_screen = NULL;
    }
    else if (rc == 1)
        field_reset(&screen->current_field.info, screen->current_field.ptr, NO);

    return rc;
}

/*=================================================================
=======================
   0 Didn't use the key
   1 Used the key
   2 Used the key (reset my field)
 *-----------------------------------------------------------------*/
int edit_field_key_handler(Edit_field_info *info, int key)
{
    int x = wherex(), y = wherey();
    char *where;

    /*-----------------------------------------------------------------
```

```
 */
switch (key)
{
     /*-------------------------------------------------------------
      *  If this field was changed, then I'll use the key and request the
      * field be restored to its original value.  Otherwise I don't use the
      * key, I'll let some upper key-handler use the key.
      */
     case KEY_esc:  return (info->changed ? 2 : 0);

     /*-------------------------------------------------------------
      *  Movement keys only affect the cursor location.
      */
     case KEY_left:
        if (--info->cursor_offset < 0)
             info->cursor_offset++;
        else  x--;
        break;

     case KEY_right:
        if (++info->cursor_offset > info->size)
             info->cursor_offset--;
        else  x++;
        break;

     case KEY_home:
        x -= info->cursor_offset;
        info->cursor_offset = 0;
        break;

     case KEY_end:
        x += info->size - info->cursor_offset;
        info->cursor_offset = info->size;
        break;

     /*-------------------------------------------------------------
      *  Deletion keys affect content, and sometimes cursor location.  The
      * backspace is turned into an equivalent delete.
      */
     case KEY_bs:
        if (info->cursor_offset == 0)  break;
        info->cursor_offset--;  x--;
        gotoxy(x, y);

     case KEY_del:
        if (info->size == 0  ||  info->cursor_offset >= info->size)  break;
        where = info->work + info->cursor_offset;
        memmove(where, where+1, info->size - (where - info->work));
        cputs(where);  putch(' ');
        info->size--;  info->changed = YES;
```

```
            break;

        default: return NO;
    }

    /*-------------------------------------------------------------
     */
    gotoxy(x, y);
    return YES;
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
void edit_field_insert(Edit_field_info *info, int key)
{
    char *where = info->work + info->cursor_offset;

    /*-------------------------------------------------------------
     */
    if (info->cursor_offset < info->size)
    {
        int x = wherex(), y = wherey();
        memmove(where+1, where, info->size - (where - info->work) + 1);
        cputs(where);
        gotoxy(x, y);
    }

    /*-------------------------------------------------------------
     */
    putch(*where = key);
    info->size++; info->cursor_offset++; info->changed = YES;
    info->work[info->size] = 0;
}
```

```
/*=============================================================================
=============================
 *  Public Domain
 *      Thomas J. Silva and The Telemetry Workshop, Inc.
 *-----------------------------------------------------------------*/
#include <dos.h>
#include <mem.h>

#include <emm.h>

/*=============================================================================
=============================
 *-----------------------------------------------------------------*/
void emm_info_construct(Emm_info *this)
{
   static char emm_name[] = "EMMXXXX0";

   ubyte far*emm_dev;
   char *name;
   ushort version, frame_seg, count;

   /*-----------------------------------------------------------------
    *  Make sure phys.pages is zero (in case I return for error).  Then get
    * the interrupt vector for the EMM device driver.
    *  The device driver has its name at offset 0x0a, so use the segment of
    * the interrupt vector, but force the offset.  Check the 8 byte name.
    */
   memset(this, 0, sizeof(Emm_info));

   emm_dev = MK_FP(FP_SEG(getvect(EMM_INT)),0x0a);

   for (name = emm_name;  *name  &&  *name == *emm_dev;  name++, emm_dev++);
   if (*name != 0)  return;

   /*-----------------------------------------------------------------
    *  Now I know that the EMM driver exists, so go get its version number.
    */
   if ((this->last_error = _emm_version(&version)) != 0)  return;
   if (version != EMM_VER4)  return;

   /*-----------------------------------------------------------------
    *  Get the total number of logical pages, and the free page count.  Then
    * get the page-frame segment and fill in the page segment array.
    */
   if ((this->last_error = _emm_pages(&this->pages_total,  &this->pages_free)) != 0)  return;

   if ((this->last_error = _emm_frame_seg(&frame_seg)) != 0)  return;

   this->phys.pages = 4;
   for (count = 0;  count < 4;  count++)
```

```
    {
        this->phys.segments[count] = frame_seg + (count * EMM_PAGE_PARAS);
        this->map_0123[count].page = this->map_0123[count].phys = count;
    }
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
void emm_construct_new(Emm_handle *this, Emm_info *info, ushort pages)
{
    this->info = info;
    /*---OLD this->pages_mapped = 0; ---*/
    this->pages = pages;

    if ((info->last_error = _emm_alloc(&this->handle, pages)) != 0) this->handle = 0;
    _emm_pages(&info->pages_total, &info->pages_free);
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
void emm_construct_known(Emm_handle *this, Emm_info *info, ushort handle)
{
    this->info = info;
    /*---OLD this->pages_mapped = 0; ---*/
    this->handle = handle;
    if ((info->last_error = _emm_handle_pages(handle, &this->pages)) != 0) this->handle = 0;
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
void emm_destroy(Emm_handle *this)
{
    /*---OLD Emm_mapped *map, *map_end; ---*/

    if (this->handle == 0) return;
    this->info->last_error = _emm_free(this->handle);
    this->handle = 0;

    this->info->pages_free += this->pages;

    /*--- OLD ---
    if (this->pages_mapped == 0) return;
    for (map = this->info->mapped, map_end = map + 4; map < map_end; map++)
        if (map->emm == this) map->emm = NULL;
    --- OLD ---*/
}
```

```
;=============================================================================
; *   Public Domain
; *              Thomas J. Silva  and  The Telemetry Workshop, Inc.
; *
ideal
model large,C

;=============================================================================
codeseg

;=============================================================================
; int _emm_version(ushort *version)
; returns EMM return code
;-----------------------------------------------------------------------------
proc _emm_version
        arg arg_VERS:ptr word

        mov             ah,46h
        int             67h                     ;Get version, al = version, ah = rc
        xor             cx,cx
        xchg            cl,ah                   ;cx = return code
        les             bx,[arg_VERS]
        mov             [es:bx],ax              ;ax = version
        mov             ax,cx
        ret
endp

;=============================================================================
; int _emm_pages(ushort *total, ushort *free)
; returns EMM return code
;-----------------------------------------------------------------------------
proc _emm_pages
        arg arg_TOTAL:ptr word, arg_FREE:ptr word

        mov             ah,42h
        int             67h                     ;Get logical page count, dx total, bx free
        mov             cx,bx
        les             bx,[arg_TOTAL]
        mov             [es:bx],dx              ;dx = total
        les             bx,[arg_FREE]
        mov             [es:bx],cx              ;cx = free
        mov             al,ah
        xor             ah,ah
        ret
endp
```

```
;=============================================================
; int _emm_frame_seg(ushort *seg)
; returns EMM return code
;-------------------------------------------------------------
proc _emm_frame_seg
        arg arg_SEG:ptr word

        mov             ah,41h
        int             67h                     ;Get page frame segment, bx = segment
        mov             cx,bx
        les             bx,[arg_SEG]
        mov             [es:bx],cx              ;cx = segment
        mov             al,ah
        xor             ah,ah
        ret
endp

;=============================================================
; int _emm_alloc(ushort *handle, ushort pages)
; returns EMM return code
;-------------------------------------------------------------
proc _emm_alloc
        arg arg_HANDLE:ptr word, arg_PAGES:word

        mov             bx,[arg_PAGES]          ;bx = pages needed
        mov             ah,43h
        int             67h                     ;Allocate new handle, dx = handle
        les             bx,[arg_HANDLE]
        mov             [es:bx],dx              ;dx = handle
        mov             al,ah
        xor             ah,ah
        ret
endp

;=============================================================
; int _emm_realloc(ushort handle, ushort pages)
; returns EMM return code
;-------------------------------------------------------------
proc _emm_realloc
        arg arg_HANDLE:word, arg_PAGES:word

        mov             bx,[arg_PAGES]          ;bx = pages needed
        mov             dx,[arg_HANDLE]         ;dx = handle
        mov             ah,51h
        int             67h                     ;Resize the handle
        mov             al,ah
        xor             ah,ah
```

```
        ret
endp

;=============================================================
; int _emm_free(ushort handle)
; returns EMM return code
;-------------------------------------------------------------
proc _emm_free
        arg arg_HANDLE:word

        mov             dx,[arg_HANDLE]         ;dx = handle
        mov             ah,45h
        int             67h                     ;Free the handle
        mov             al,ah
        xor             ah,ah
        ret
endp

;=============================================================
; int _emm_handle_pages(ushort handle, ushort *pages)
; returns EMM return code
;-------------------------------------------------------------
proc _emm_handle_pages
        arg arg_HANDLE:word, arg_PAGES:ptr word

        mov             dx,[arg_HANDLE]         ;dx = handle
        mov             ah,4ch
        int             67h                     ;Get handle's page count, bx = pages
        mov             dx,bx
        les             bx,[arg_PAGES]
        mov             [es:bx],dx              ;dx = pages
        mov             al,ah
        xor             ah,ah
        ret
endp

;=============================================================
; int _emm_map(ushort phys_page, ushort handle, ushort page)
; returns EMM return code
;-------------------------------------------------------------
proc _emm_map
        arg arg_PHYS:word, arg_HANDLE:word, arg_PAGE:word

        mov             ax,[arg_PHYS]           ;al = physical page number
        mov             dx,[arg_HANDLE]         ;dx = handle
        mov             bx,[arg_PAGE]           ;bx = page number
        mov             ah,44h
```

```
        int             67h                     ;Map one page
        mov             al,ah
        xor             ah,ah
        ret
endp

;=============================================================================
; int _emm_map_multi(int phys_type, ushort handle, ushort count, Emm_map *maps)
; returns EMM return code
;-----------------------------------------------------------------------------
proc _emm_map_multi uses si ds
        arg arg_TYPE:word, arg_HANDLE:word, arg_CNT:word, arg_MAPS:ptr word

        mov             ax,[arg_TYPE]           ;al = type of map structure
        mov             dx,[arg_HANDLE]         ;dx = handle
        mov             cx,[arg_CNT]            ;cx = page count
        lds             si,[arg_MAPS]           ;ds:si -> mapping structures
        mov             ah,50h
        int             67h                     ;Map multiple pages
        mov             al,ah
        xor             ah,ah
        ret
endp

;=============================================================================
; int _emm_name_get(ushort handle, char *name)
; returns EMM return code
;-----------------------------------------------------------------------------
proc _emm_name_get uses di
        arg arg_HANDLE:word, arg_NAME:ptr byte

        mov             dx,[arg_HANDLE]         ;dx = handle
        les             di,[arg_NAME]           ;es:di -> name area
        mov             ax,ds
        mov             es,ax
        mov             ax,5300h
        int             67h                     ;Get name of handle
        mov             al,ah
        xor             ah,ah
        ret
endp

;=============================================================================
; int _emm_name_set(ushort handle, char *name)
; returns EMM return code
;-----------------------------------------------------------------------------
proc _emm_name_set uses si ds
```

```
        arg arg_HANDLE:word, arg_NAME:ptr byte

        mov             dx,[arg_HANDLE]     ;dx = handle
        lds             si,[arg_NAME]       ;ds:si -> name string
        mov             ax,5301h
        int             67h                 ;Set name of handle
        mov             al,ah
        xor             ah,ah
        ret
endp

;=============================================================================
; int _emm_name_find(ushort *handle, char *name)
; returns EMM return code
;-----------------------------------------------------------------------------
proc _emm_name_find uses si ds
        arg arg_HANDLE:ptr word, arg_NAME:ptr byte

        lds             si,[arg_NAME]           ;ds:si -> name string
        mov             ax,5401h
        int             67h                     ;Find handle by name, dx = handle
        les             bx,[arg_HANDLE]
        mov             [es:bx],dx              ;dx = handle
        mov             al,ah
        xor             ah,ah
        ret
endp

public _emm_version, _emm_pages, _emm_frame_seg, _emm_alloc, _emm_realloc
public _emm_free, _emm_handle_pages, _emm_map, _emm_map_multi, _emm_name_get
public _emm_name_set, _emm_name_find
end
```

```
/*=============================================================
============================
 *  Public Domain
 *      Thomas J. Silva and The Telemetry Workshop, Inc.
 *-----------------------------------------------------------*/
#include <emm.h>

/*=============================================================
============================
 *-----------------------------------------------------------*/
int emm_map(Emm_handle *this, ushort phys_page, ushort page)
{
   Emm_info *info = this->info;

   if (phys_page >= info->phys.pages)  return info->last_error = 0x8b;

   if ((info->last_error = _emm_map(phys_page, this->handle, page)) == 0)
   {
        /*--- OLD ---
        if (info->mapped[phys_page].emm != NULL)
           info->mapped[phys_page].emm->pages_mapped--;

        info->mapped[phys_page].emm = this;
        info->mapped[phys_page].page = page;
        this->pages_mapped++;
        --- OLD ---*/
   }

   return info->last_error;
}
```

```
/*=============================================================================
============================
 *  Public Domain
 *      Thomas J. Silva  and  The Telemetry Workshop, Inc.
 *-----------------------------------------------------------------------*/
#include <emm.h>

/*=============================================================================
============================
 *-----------------------------------------------------------------------*/
int emm_map_multi(Emm_handle *this, int type, ushort count, Emm_map *maps)
{
   Emm_map *map, *map_end = maps + count;
   Emm_info *info = this->info;
   ushort phys_page;
   /*---OLD Emm_mapped *mapped; ---*/

   /*-----------------------------------------------------------------------
    */
   for (map = maps;  map < map_end;  map++)
   {
        if (map->page >= this->pages)  return info->last_error = 0x8a;

        phys_page = (type == EMM_phys_segment
                    ? (map->phys - info->phys.segments[0]) / EMM_PAGE_PARAS
                    : map->phys);

        if (phys_page >= info->phys.pages)  return info->last_error = 0x8b;

        /*--- OLD ---
        mapped = info->mapped + phys_page;
        if (mapped->emm != NULL)
        {
           mapped->emm->pages_mapped--;
           mapped->emm = NULL;
        }
        --- OLD ---*/
   }

   /*-----------------------------------------------------------------------
    */
   info->last_error = _emm_map_multi(type, this->handle, count, maps);
   if (info->last_error == 0)
   {
        /*--- OLD ---
        for (map = maps;  map < map_end;  map++)
        {
           mapped = info->mapped
             + (type == EMM_phys_segment
                ? (map->phys - info->phys.segments[0]) / EMM_PAGE_PARAS
```

```
                : map->phys);

            mapped->emm = this;
            mapped->page = map->page;
            this->pages_mapped++;
        }
        --- OLD ---*/
    }

    return info->last_error;
}
```

```
;=============================================================
============================
ideal
model large,C

;=============================================================
============================
codeseg

;=============================================================
============================
; key_hit()
;-------------------------------------------------------------
proc key_hit
        mov             ah,11h
        int             16h
        jnz             hit_it          ;if ZR ... don't got
        xor             ax,ax           ;return 0
        ret
        hit_it:                         ;else ... got one
        or              ax,ax
        jnz             hit_ok          ;if it has 0 value
        mov             al,3            ;make it 3 (control-C)
        hit_ok:
        ret
endp

;=============================================================
============================
; key_get()
;-------------------------------------------------------------
proc key_get
        mov             ah,10h
        int             16h
        or              al,al
        jnz             not_special     ;if low byte 0
        mov             al,1
        xchg            ah,al           ;high is special code, move to low byte
        ret
        not_special:
        js              got_extend      ;else if normal ascii
        mov             ah,0            ;clear high byte (scan code)
        ret
        got_extend:                     ;else if extended key
        mov             al,1            ;clear low byte (00e0)
        xchg            ah,al
        ret
endp

;=============================================================
```

```
;=============================
; key_flags()
;-----------------------------------------------------------------
proc key_flags
        mov             ah,12h
        int             16h
        ret
endp

public key_hit, key_get, key_flags
end
```

```
;=============================================================================
=============================
ideal
model large,C

;=============================================================================
=============================
codeseg
bios_seg        dw 40h
BIOS_KEY1       = 17h
BIOS_KEY2       = 18h
KEY2_PAUSE      = 08h

label old_09 dword
old_09_off      dw 0
old_09_seg      dw 0

;=============================================================================
=============================
;-----------------------------------------------------------------------------
proc _kybd_my_09
        pushf
        call            [cs:old_09]
        cli
        push            ds
        mov             ds,[cs:bios_seg]
        and             [BIOS_KEY2],not KEY2_PAUSE
        pop             ds
        iret
endp

;=============================================================================
=============================
;-----------------------------------------------------------------------------
proc kybd_start
        push            ds
        mov             ax,3509h
        int             21h                     ;get and save current int-09 handler
        mov             [cs:old_09_off],bx
        mov             [cs:old_09_seg],es
        Smov            ds,ax,cs
        mov             dx,offset _kybd_my_09   ;install my_09 as int-09 handler
        mov             ax,2509h
        int             21h
        pop             ds
        ret
endp

proc kybd_stop
        push            ds
```

```
        lds             dx,[cs:old_09]
        mov             ax,2509h                ;restore old handler
        int             21h
        pop             ds
        ret
endp

public _kybd_my_09, kybd_start, kybd_stop
end
```

```
;=============================================================================
=============================
ideal
model large,C

;=============================================================================
=============================
codeseg

;=============================================================================
=============================
; void swap_double(void *to, void *from)
;-----------------------------------------------------------------
proc swap_double uses si ds
        arg arg_TO:ptr word, arg_FROM:ptr word

        ;-----------------------------------------------------------------
        lds             si,[arg_FROM]
        mov             ax,[si]
        mov             bx,[si+2]
        mov             cx,[si+4]
        mov             dx,[si+6]
        xchg            al,dh
        xchg            ah,dl
        xchg            bl,ch
        xchg            bh,cl

        ;-----------------------------------------------------------------
        lds             si,[arg_TO]
        mov             [si],ax
        mov             [si+2],bx
        mov             [si+4],cx
        mov             [si+6],dx

        ret
endp

;=============================================================================
=============================
; double swap_double_out(void *from)
;-----------------------------------------------------------------
proc swap_double_out
        arg arg_FROM:ptr word
        local temp:qword

        ;-----------------------------------------------------------------
        lea             bx,[temp]
        call            swap_double, bx, ss, [arg_FROM]
        fld             [temp]
```

```
        ret
endp

public swap_double, swap_double_out
end
```

```
;=============================================================================
ideal
model large,C

;=============================================================================
codeseg

;=============================================================================
; void swap_float(void *to, void *from)
;-----------------------------------------------------------------------------
proc swap_float uses si ds
        arg arg_TO:ptr word, arg_FROM:ptr word

        ;-------------------------------------------------------------------
        lds             si,[arg_FROM]
        mov             ax,[si]
        mov             dx,[si+2]
        xchg            al,dh
        xchg            ah,dl

        ;-------------------------------------------------------------------
        lds             si,[arg_TO]
        mov             [si],ax
        mov             [si+2],dx

        ret
endp

;=============================================================================
; float swap_float_out(void *from)
;-----------------------------------------------------------------------------
proc swap_float_out
        arg arg_FROM:ptr word
        local temp:dword

        ;-------------------------------------------------------------------
        lea             bx,[temp]
        call            swap_float, bx, ss, [arg_FROM]
        fld             [temp]

        ret
endp

public swap_float, swap_float_out
end
```

```
;=============================================================================
=============================
ideal
model large,C

;=============================================================================
=============================
codeseg

;=============================================================================
=============================
; void swap_short(void *to, void *from)
;-----------------------------------------------------------------------------
proc swap_short
        arg arg_TO:ptr word, arg_FROM:ptr word

        ;-------------------------------------------------------------------
        les             bx,[arg_FROM]
        mov             ax,[es:bx]
        xchg            al,ah

        les             bx,[arg_TO]
        mov             [es:bx],ax
        ret
endp

;=============================================================================
=============================
; ushort swap_short_out(void *from)
;-----------------------------------------------------------------------------
proc swap_short_out
        arg arg_FROM:ptr word

        ;-------------------------------------------------------------------
        les             bx,[arg_FROM]
        mov             ax,[es:bx]
        xchg            al,ah

        ret
endp

public swap_short, swap_short_out
end
```

```
;=============================================================
============================
; *  Public Domain
; *              Thomas J. Silva and Bruce G. Jackson & Associates, Inc.
; *
ideal
model large,C

;=============================================================
============================
dataseg
secs_per_day    dd      86400
secs_per_min    dw      60
one_thousand    dw      1000

temp_word               dw      0

;=============================================================
============================
codeseg

;=============================================================
============================
; void time_dbl_to_dhms(double far*time, Time_dhms far*dhms)
;
; NOTES:
;       This routine was adapted from the dhms routine used by the PCDecom display
; application.
;
;       I assume the time (in seconds) starts at day 0, and I add one to the day,
; thus days are 1-365.
;       Note: I no longer increment the day since elapsed time tends to get screwed
; up that way.  I just leave it as is and Caveat Emptor!
;
;-------------------------------------------------------------------
proc time_dbl_to_dhms uses di
        arg arg_TIME:dword, arg_DHMS:dword
        local var_OLD_CW:word, var_NEW_CW:word

        ;---------------------------------------------------------------
        ;  Load the value into math register st(0).
        ;
        les             bx,[arg_TIME]           ;es:bx -> time
        fld             [qword ptr es:bx]       ;load double into st(0)

        ;---------------------------------------------------------------
        ;  Save the math control word, it contains the current rounding mode.  Turn
        ; on round-towards-0 mode.
        ;  Make a pointer to the start of the work buffer, I'll fill in from the
        ; front first.
```

```
        ;
        rounding:
        fstcw           [var_OLD_CW]                ;save rounding mode
        fwait
        mov             ax,[var_OLD_CW]
        or              ax,0c00h                    ;turn on round-to-0 mode
        mov             [var_NEW_CW],ax
        fldcw           [var_NEW_CW]

        les             di,[arg_DHMS]

        ;-----------------------------------------------------------------
        ;  Load a divide-constant into math register st(0), then call the crank
        ; routine, it extracts the next integer.
        ;
        fild            [secs_per_day]
        call            crank                       ;extract days
;       inc             ax                          ;days++  ... make days 1-365
        stosw

        fild            [secs_per_min]
        call            crank                       ;extract total minutes in day
        mov             bx,60
        div             bl                          ;al = mins/60 (hours)
        xchg            bh,ah                       ;bh = mins%60 (minutes)
        stosw
        mov             al,bh
        stosw

        call            crank_half                  ;extract seconds
        stosw

        ;-----------------------------------------------------------------
        ;  Restore the rounding mode.  Convert and store the fraction.
        ;
        fldcw   [var_OLD_CW]        ;reset rounding mode

        fimul   [one_thousand]      ;st(0) = st(0) * 1000
        fist    [word es:di]        ;store milli-seconds

        ;-----------------------------------------------------------------
        ;  Pop math register st(0).
        ;
        ffree   st(0)
        ret
endp

;=================================================================
=========================.=
; inputs:       value in st(1), constant in st(0)
```

```
; returns:      adjusted value in st(0)
;                       ax = extracted integer (also in temp_word)
;-------------------------------------------------------------------------
crank:
        fld     st(0)                   ;duplicate the constant
        fdivr   st(0),st(2)             ;st(0) = st(2) / st(0) ... int = value / const
        frndint
        fist    [temp_word]             ;store rounded integer
        fmulp   st(1),st(0)             ;st(1) = st(1) * st(0) ... part = const * int, pop
        fsubp   st(1),st(0)             ;st(1) = st(1) - st(0) ... value -= part, pop
        mov     ax,[temp_word]
        ret

;=========================================================================
=========================
; inputs:       value in st(0)
; returns:      adjusted value in st(0)
;                       ax = extracted integer (also in temp_word)
;-------------------------------------------------------------------------
crank_half:
        fld     st(0)                   ;duplicate the value
        frndint
        fist    [temp_word]             ;store rounded integer
        fsubp   st(1),st(0)             ;st(1) = st(1) - st(0) ... value -= int, pop
        mov     ax,[temp_word]
        ret

public time_dbl_to_dhms
end
```

```
;=============================================================================
; *  Public Domain
; *              Thomas J. Silva and Bruce G. Jackson & Associates, Inc.
; *
ideal
model large,C

include "times.inc"

extrn time_dbl_to_dhms:far, time_dhms_to_string:far

;=============================================================================
codeseg

;=============================================================================
; char far*time_dbl_to_string(double far*time, char far*work)
;
; NOTES:
;       This routine makes this fixed size string: "ddd:hh:mm:ss.123", and needs 18
; bytes with which to work, so size your work area accordingly.
;
;-----------------------------------------------------------------------------
proc time_dbl_to_string
        arg arg_TIME:dword, arg_WORK:dword
        local var_DHMS:time_dhms

        ;-------------------------------------------------------------------
        ;  Convert the time into dhms integers, then convert that into a string.
        ;
        lea             bx,[var_DHMS]
        push            bx
        call            time_dbl_to_dhms, [arg_TIME], bx, ss

        pop             bx
        call            time_dhms_to_string, bx, ss, [arg_WORK]

        ret
endp

public time_dbl_to_string
end
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#include <types.h>
#include <times.h>

/*=============================================================
============================
 *  The base date is: 3 Jan 93, julian day 3, GPS week 678.
 *-----------------------------------------------------------*/
#define BASE_YEAR 1993
#define BASE_LEAP   NO
#define BASE_WEEK  678
#define BASE_JDAY    3
#define SECS_IN_DAY 86400

/*=============================================================
============================
 *-----------------------------------------------------------*/
void time_gps_to_utc(Time_gps *gps, double gps_leads_utc, Time_utc *utc)
{
   int year = BASE_YEAR,
       leap = BASE_LEAP,
       jday = BASE_JDAY,
       weeks = gps->week - BASE_WEEK,
       wday;
   double secs;

   /*-------------------------------------------------------------
    *  Compute seconds in current day, and day of week.
    */
   if ((secs = gps->secs - gps_leads_utc) < 0)
   {
       weeks--;
       secs += SECS_IN_DAY * 7;
   }

   if ((wday = secs / SECS_IN_DAY) > 0)
       secs -= (long)wday * SECS_IN_DAY;

   /*-------------------------------------------------------------
    *  Compute julian day, loop until it fits in the proper year.
    */
   jday += weeks * 7 + wday;

   while (jday < 0)
   {
       year--;
       leap = (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0) ? YES : NO);
       jday += 365 + leap;
```

```
    }

    while (jday > 365 + leap)
    {
        jday -= 365 + leap;
        year++;
        leap = (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0) ? YES : NO);
    }

    /*-----------------------------------------------------------
     */
    utc->secs = (long)jday * SECS_IN_DAY + secs;
    utc->year = year;
}
```

```
;=============================================================================
; *  Public Domain
; *              Thomas J. Silva and Bruce G. Jackson & Associates, Inc.
; *
ideal
model large,C

include "times.inc"

;=============================================================================
dataseg
one_thousand    dw      1000

;=============================================================================
codeseg

DHMS  equ (time_dhms es:bx)

union long_int
        long    dd ?
        struc
                lsw             dw ?
                msw             dw ?
        ends
ends

;=============================================================================
; void time_dhms_to_dbl(Time_dhms far*dhms, double far*time)
;-----------------------------------------------------------------------------
proc time_dhms_to_dbl uses di
        arg dhms_ptr:ptr time_dhms, time_ptr:dword
        local seconds:long_int

        ;---------------------------------------------------------------
        ;  Pre-load seconds accumulator with seconds.
        ;
        les             bx,[dhms_ptr]

        mov             ax,[DHMS.secs]
        mov             [seconds.lsw],ax
        xor             ax,ax
        mov             [seconds.msw],ax        ;seconds = seconds

        ;---------------------------------------------------------------
        ;  Turn days into seconds.
        ;
```

```
        mov     ax,[DHMS.days]
        mov     dx,24
        mul     dx                      ;ax = days * 24 ... hours
        mov     dx,3600
        mul     dx                      ;ax = (days * 24) * 60 * 60 ... seconds
        add     [seconds.lsw],ax
        adc     [seconds.msw],dx        ;seconds += (days * 24) * 60 * 60

        ;---------------------------------------------------------------
        ;  Turn hours and minutes into seconds.
        ;
        mov     ax,[DHMS.hours]
        mov     dx,60
        mul     dx                      ;ax = hrs * 60 ... minutes
        add     ax,[DHMS.mins]
        mov     dx,60
        mul     dx                      ;ax = ((hrs * 60) + mins) * 60 ... seconds
        add     [seconds.lsw],ax
        adc     [seconds.msw],dx        ;seconds += ((hrs * 60) + mins) * 60

        ;---------------------------------------------------------------
        ;  Load total seconds into math chip, load msecs and adjust it, then add
        ; to get a total time.
        ;
        fild    [seconds.long]
        fild    [DHMS.msecs]
        fidiv   [one_thousand]          ;st(0) = st(0) / 1000
        faddp   st(1),st(0)             ;st(1) = st(1) + st(0), pop

        ;---------------------------------------------------------------
        ;  Store the final result
        ;
        les     bx,[time_ptr]           ;es:bx -> time
        fstp    [qword ptr es:bx]       ;store double, pop
        fwait

        ret
endp

public time_dhms_to_dbl
end
```

```
;=============================================================================
; *  Public Domain
; *              Thomas J. Silva and Bruce G. Jackson & Associates, Inc.

ideal
model large,C

include "times.inc"

;=============================================================================
codeseg

DHMS  equ (time_dhms bx)

;=============================================================================
; char far*time_dhms_to_string(Time_dhms far*dhms, char far*work)
;
; NOTES:
;       This routine makes this fixed size string: "ddd:hh:mm:ss.123", and needs 18
; bytes with which to work, so size your work area accordingly.
;-----------------------------------------------------------------------
proc time_dhms_to_string uses di ds
        arg arg_DHMS:ptr time_dhms, arg_WORK:ptr byte

        ;-----------------------------------------------------------------
        ;  Make a pointer to the start of the work buffer, I'll fill in from the
        ; front first.
        ;
        les             di,[arg_WORK]           ;es:di -> front of the work buffer
        lds             bx,[arg_DHMS]

        mov             ax,[DHMS.days]
        cmp             ax,999
        jbe             do_day
        mov             ax,999
        do_day:
        call            fill_3digit             ;convert days, 3 digits

        dec             di                      ;backup to trailing colon
        mov             al,'/'
        stosb                                   ;add a slash

        mov             ax,[DHMS.hours]
        call            fill_2digit             ;convert hours, 2 digits

        mov             ax,[DHMS.mins]
        call            fill_2digit             ;convert minutes, 2 digits
```

```
        mov             ax,[DHMS.secs]
        call            fill_2digit             ;convert seconds, 2 digits

        ;-----------------------------------------------------------------
        ;  Convert the fraction.
        ;
        dec             di                      ;backup to trailing colon
        mov             al,'.'
        stosb                                   ;add a dot

        mov             ax,[DHMS.msecs]
        call            fill_3digit             ;convert milli-secs, 3 digits

        ;-----------------------------------------------------------------
        ;  Add a NULL terminator, return a pointer to work.
        ;
        dec             di                      ;backup to trailing colon
        xor             al,al
        stosb
        mov             dx,es
        mov             ax,[word arg_WORK]
        ret
endp

;=================================================================
;=========================
; inputs:       ax = integer value
;                       es:di -> work area
; returns:      work area filled with 2 or 3 digits, di bumped
; wasted:       ax, cl
;-----------------------------------------------------------------
fill_3digit:
        mov             cl,100
        div             cl              ;al = ax / 100, ah = ax % 100
        or              al,'0'
        stosb                           ;*work++ = al + '0'
        shr             ax,8            ;ax = ah

fill_2digit:
        mov             cl,10
        div             cl              ;al = ax / 10, ah = ax % 10
        or              ax,3030h
        stosw                           ;*work++ = al + '0', *work++ = ah + '0'
        mov             al,':'
        stosb                           ;*work++ = :
        ret

public time_dhms_to_string
end
```

```
;=============================================================================
============================
ideal
model large,C

;=============================================================================
============================
codeseg

;=============================================================================
============================
; void rtc_timer_set(ushort msec_count, ubyte *flag)
;-----------------------------------------------------------------------
proc rtc_timer_set
        arg arg_MSECS:word, arg_FLAG:ptr byte

        ;-----------------------------------------------------------------
        les             bx,[arg_FLAG]           ;es:bx -> flag, non-zero when done

        mov             ax,[arg_MSECS]
        mov             dx,1000
        mul             dx
        mov             cx,dx
        mov             dx,ax                   ;cx_dx = micro-seconds

        mov             ax,8300h                ;start timer service
        int             15h

        ret
endp

public rtc_timer_set
end
```

```
/*=============================================================
=========================
 *-------------------------------------------------------------*/
#include <math.h>

#include <vector.h>

/*=============================================================
=========================
 *-------------------------------------------------------------*/
int vector_check(Vector *v, Vector_limits *limit)
{
   double mag;

   if (fabs(v->x) > limit->hi
    || fabs(v->y) > limit->hi
    || fabs(v->z) > limit->hi)   return NO;

   mag = vector_magnitude(v);
   if (mag < limit->low  ||  mag > limit->hi)  return NO;

   return YES;
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
int state_vector__check(State_vector *v, State_vector_limits *limit)
{
   return  vector_check(&v->pos, &limit->pos) && vector_check(&v->vel, &limit->vel);
}
```

```
/*=============================================================
============================
 *  These functions convert the Shuttle state vector between M50 Inertial
 * coordinates and J2000 inertial coordinates (and back again).
 *
 * NOTE:  This really does B50, not M50, but they are close.
 *-------------------------------------------------------------*/
#include <math.h>

#include <vector.h>
#include <const.h>

/*=============================================================
============================
 *-------------------------------------------------------------*/
static double j2000_b50[3][3] =
{
    .9999256794956877,  .0111814832391717,  .0048590037723143,
   -.0111814832204662,  .9999374848933135, -.0000271702937440,
   -.0048590038153592, -.0000271625947142,  .9999881946023742
};

/*=============================================================
============================
 *-------------------------------------------------------------*/
void
state_vector__m50_to_j2000(State_vector *m50, State_vector *j2000)
{
   double b50_j2000[3][3];

   matrix_transpose(j2000_b50, b50_j2000);
   vector_transform(&m50->pos, &j2000->pos, b50_j2000);
   vector_transform(&m50->vel, &j2000->vel, b50_j2000);
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
void
state_vector__j2000_to_m50(State_vector *j2000, State_vector *m50)
{
   vector_transform(&j2000->pos, &m50->pos, j2000_b50);
   vector_transform(&j2000->vel, &m50->vel, j2000_b50);
}
```

```
/*=============================================================
============================
 *  These functions convert the Shuttle state vector between M50 Inertial
 * coordinates and ECEF coordinates (and back again).
 *
 * NOTE:  Values may be in any units, and time must be UTC.
 *-------------------------------------------------------------*/
#include <math.h>

#include <vector.h>
#include <const.h>

/*=============================================================
============================
 *-------------------------------------------------------------*/
void
state_vector__rotate_m50_ecef(State_vector_transform  *m50_ecef,
                              double new_time,
                              State_vector_transform *ecef_m50)
{
   double c_lam, s_lam, lambda;

   ecef_m50->time = new_time;

   lambda = EARTH_OMEGA * (new_time - m50_ecef->time);
   c_lam = cos(lambda);
   s_lam = sin(lambda);

   ecef_m50->matrix[0][0] =  m50_ecef->matrix[0][0]*c_lam  + m50_ecef->matrix[1][0]*s_lam;
   ecef_m50->matrix[0][1] = -m50_ecef->matrix[0][0]*s_lam  + m50_ecef->matrix[1][0]*c_lam;
   ecef_m50->matrix[0][2] =  m50_ecef->matrix[2][0];

   ecef_m50->matrix[1][0] =  m50_ecef->matrix[0][1]*c_lam  + m50_ecef->matrix[1][1]*s_lam;
   ecef_m50->matrix[1][1] = -m50_ecef->matrix[0][1]*s_lam  + m50_ecef->matrix[1][1]*c_lam;
   ecef_m50->matrix[1][2] =  m50_ecef->matrix[2][1];

   ecef_m50->matrix[2][0] =  m50_ecef->matrix[0][2]*c_lam  + m50_ecef->matrix[1][2]*s_lam;
   ecef_m50->matrix[2][1] = -m50_ecef->matrix[0][2]*s_lam  + m50_ecef->matrix[1][2]*c_lam;
   ecef_m50->matrix[2][2] =  m50_ecef->matrix[2][2];
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
#ifdef GIGO
static show_vector(char *title, Vector *v)
{
   printf("%s\t%20lf %20lf %20lf\n", title, v->x, v->y, v->z);
}
```

```
static show_matrix(char *title, double m[3][3])
{
   static char fmt[] =
        "%s\t%20lf %20lf %20lf\n"
        "\t\t%20lf %20lf %20lf\n"
        "\t\t%20lf %20lf %20lf\n";

   printf(fmt, title,
     m[0][0], m[0][1], m[0][2],
     m[1][0], m[1][1], m[1][2],
     m[2][0], m[2][1], m[2][2]);
}
#endif GIGO

/*=============================================================
==========================
 *  Note: Velocity may be an inertial velocity expressed in ECEF coordinates.
 * Otherwise it is an ECEF velocity, expressed in ECEF coordinates.
 *-------------------------------------------------------------*/
void
state_vector__m50_to_ecef(State_vector *m50, State_vector *ecef,
                          State_vector_transform *m50_ecef_epoch, int vel_is_ecef)
{
   Vector omega, omegaXr, vrel;
   State_vector_transform ecef_m50, m50_ecef;

   /*-----------------------------------------------------------
    */
   state_vector__rotate_m50_ecef(m50_ecef_epoch,  m50->time, &ecef_m50);

   if (vel_is_ecef)
   {
        omega.x = EARTH_OMEGA * ecef_m50.matrix[0][2];
        omega.y = EARTH_OMEGA * ecef_m50.matrix[1][2];
        omega.z = EARTH_OMEGA * ecef_m50.matrix[2][2];

        vector_cross(&omega, &m50->pos, &omegaXr);
        vector_diff(&m50->vel, &omegaXr, &vrel);        // m50 - omegaXr
   }
   else  vrel = m50->vel;

   /*-----------------------------------------------------------
    */
   matrix_transpose(ecef_m50.matrix,  m50_ecef.matrix);
   vector_transform(&m50->pos, &ecef->pos, m50_ecef.matrix);
   vector_transform(&vrel, &ecef->vel, m50_ecef.matrix);
}

/*=============================================================
==========================
```

```
 *-------------------------------------------------------------------*/
void
state_vector__ecef_to_m50(State_vector  *ecef, State_vector *m50,
                          State_vector_transform *m50_ecef_epoch, int vel_is_ecef)
{
   Vector omega, omegaXr;
   State_vector_transform ecef_m50;

   /*-----------------------------------------------------------------
    */
   state_vector__rotate_m50_ecef(m50_ecef_epoch,  ecef->time, &ecef_m50);

   vector_transform(&ecef->pos,  &m50->pos, ecef_m50.matrix);
   vector_transform(&ecef->vel,  &m50->vel, ecef_m50.matrix);

   /*-----------------------------------------------------------------
    */
   if (vel_is_ecef)
   {
        omega.x = EARTH_OMEGA * ecef_m50.matrix[0][2];
        omega.y = EARTH_OMEGA * ecef_m50.matrix[1][2];
        omega.z = EARTH_OMEGA * ecef_m50.matrix[2][2];

        vector_cross(&omega,  &m50->pos, &omegaXr);
        m50->vel.x += omegaXr.x;      // m50 + omegaXr
        m50->vel.y += omegaXr.y;
        m50->vel.z += omegaXr.z;
   }
}
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#include <math.h>

#include <vector.h>

/*=============================================================
============================
 *  Both m_in and m_out may be the same matrix.
 *-----------------------------------------------------------*/
void matrix_transpose(double m_in[3][3], double m_out[3][3])
{
   if (m_in == m_out)
   {
        double temp;
        temp = m_out[0][1];  m_out[0][1] = m_out[1][0];  m_out[1][0] = temp;
        temp = m_out[0][2];  m_out[0][2] = m_out[2][0];  m_out[2][0] = temp;
        temp = m_out[1][2];  m_out[1][2] = m_out[2][1];  m_out[2][1] = temp;
   }
   else
   {
        int row, col;

        for (row = 0;  row < 3;  row++)
           for (col = 0;  col < 3;  col++)
                m_out[row][col] = m_in[col][row];
   }
}

/*=============================================================
============================
 *  I compute the INERTIAL to LVLH transform matrix given an inertial vector.
 * This is a standard post-multiply matrix, where Vl = M Vi, and Vi is a column.
 *-----------------------------------------------------------*/
int state_vector__itol_matrix(State_vector *sv, double matrix[3][3])
{
   Vector temp;

   /*-------------------------------------------------------------
    */
   temp.x = -sv->pos.x;
   temp.y = -sv->pos.y;
   temp.z = -sv->pos.z;

   if (vector_unit(&temp, (Vector *)matrix[2]) == NO)  return NO;

   vector_cross(&sv->vel, &sv->pos, &temp);
   if (vector_unit(&temp, (Vector *)matrix[1]) == NO)  return NO;
```

```
    vector_cross((Vector *)matrix[1], (Vector *)matrix[2], &temp);
    if (vector_unit(&temp, (Vector *)matrix[0]) == NO)  return NO;

    return YES;
}
```

```
/*=============================================================================
=============================
 *  See vector.h for a discussion of quaternion order and direction.
 *-----------------------------------------------------------------------------*/
#include <math.h>

#include <vector.h>

/*=============================================================================
=============================
 *-----------------------------------------------------------------------------*/
void quaternion_conjugate(Quaternion *q)
{
   q->x = -q->x;
   q->y = -q->y;
   q->z = -q->z;
}

/*=============================================================================
=============================
 *  Quaternion multiply order is reversed from matrix multiply, but matrix
 * multiply is not straightforward either, so read on.
 *
 *  If you have transforms from A-to-B and B-to-C, then the matrix equation
 * will be: Vc = M_btoc * M_atob * Va, remember: the vector is post-multiplied
 * as a column, and the result Vb is post-multiplied again.
 *  The corresponding quaternion equation is: Vc = Va * Q_atob * Q_btoc.
 *-----------------------------------------------------------------------------*/
//void quaternion_mult(Quaternion *q1, Quaternion *q2, Quaternion *qout)
//{
//   qout->s = (q1->s * q2->s) - (q1->x * q2->x) - (q1->y * q2->y) - (q1->z * q2->z);
//   qout->x = (q1->s * q2->x) + (q1->x * q2->s) + (q1->y * q2->z) - (q1->z * q2->y);
//   qout->y = (q1->s * q2->y) - (q1->x * q2->z) + (q1->y * q2->s) + (q1->z * q2->x);
//   qout->z = (q1->s * q2->z) + (q1->x * q2->y) - (q1->y * q2->x) + (q1->z * q2->s);
//}

/*=============================================================================
=============================
 *  This routine creates a direction-cosine matrix from a quaternion.  The
 * matrix is computed in the "direction" of the quaternion: if the quaternion
 * is body-to-inertial then the matrix denotes the rotation from the body frame
 * to the inertial frame.
 *
 *  NOTE: See vector.h about NASA quaternion twist direction.  As as result of
 * the reversed definition, all the orbiter code that I've seen has reversed
 * signs on the non-squared elements (the same as transposing the matrix).
 *
 *          2  2  2  2
 *  m00 = s + x - y - z      m01 = 2xy + 2sz          m02 = 2xz - 2sy
 *
```

```
 *                                  2   2   2   2
 *  m10 = 2xy - 2sz         m11 = s - x + y - z      m12 = 2yz + 2sx
 *
 *                                                        2   2   2   2
 *  m20 = 2xz + 2sy         m21 = 2yz - 2sx          m22 = s - x - y + z
 *
 * Equivalent diagonal elements:
 *              2   2                    2   2            2   2
 *  m00 = 1 - 2y - 2z       m11 = 1 - 2x - 2z        m22 = 1 - 2x - 2y
 *-----------------------------------------------------------------------*/
void quaternion_to_matrix(Quaternion *q, double matrix[3][3])
{
   double ss = q->s * q->s;
   double xx = q->x * q->x;
   double yy = q->y * q->y;
   double zz = q->z * q->z;
   double xy = q->x * q->y;
   double xz = q->x * q->z;
   double yz = q->y * q->z;
   double sx = q->s * q->x;
   double sy = q->s * q->y;
   double sz = q->s * q->z;

   matrix[0][0] = ss + xx - yy - zz;
   matrix[0][1] = 2 * (xy + sz);
   matrix[0][2] = 2 * (xz - sy);
   matrix[1][0] = 2 * (xy - sz);
   matrix[1][1] = ss - xx + yy - zz;
   matrix[1][2] = 2 * (yz + sx);
   matrix[2][0] = 2 * (xz + sy);
   matrix[2][1] = 2 * (yz - sx);
   matrix[2][2] = ss - xx - yy + zz;
}
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#include <math.h>

#include <vector.h>

/*=============================================================
============================
 *  This routine produces the position vector from one state vector (target) to
 * another state vector (chaser).  The inputs are assumed to be in GCI, and the
 * output is returned in the LVLH of the principle state vector (target).
 *
 *  Vector rv
 *       x = vbar, positive forward
 *       y = hbar, positive "north"
 *       z = rbar, positive down
 *
 *  Vector rvdot
 *       x = magnitude of (position difference vector)
 *       y = ydot
 *       z = closure rate
 *-------------------------------------------------------------*/
void vector_rvbar(State_vector *target, State_vector *chaser, Vector *rv, Vector *rvdot)
{
   double mag_tpos, magxxv, magvbar, lcg[3][3], w;
   Vector xtxvt, hhat, mhhat, vb, xop, dif, rvbdif, omega, vdifi, vdifl, rdot, wxr;
   double hbar, vbar, rbar, cos_theta, eta;

   /*-----------------------------------------------------------------
    */
   mag_tpos = vector_magnitude(&target->pos);

   lcg[2][0] = -target->pos.x / mag_tpos;
   lcg[2][1] = -target->pos.y / mag_tpos;
   lcg[2][2] = -target->pos.z / mag_tpos;

   /*-----------------------------------------------------------------
    */
   vector_cross(&target->pos, &target->vel, &xtxvt);
   magxxv = vector_magnitude(&xtxvt);

   lcg[1][0] =  mhhat.x =  -(hhat.x = xtxvt.x / magxxv);
   lcg[1][1] =  mhhat.y =  -(hhat.y = xtxvt.y / magxxv);
   lcg[1][2] =  mhhat.z =  -(hhat.z = xtxvt.z / magxxv);

   hbar = vector_dot(&hhat, &chaser->pos);
   vector_cross(&target->pos, &mhhat, &vb);

   xop.x = chaser->pos.x - hhat.x * hbar;
```

```
xop.y = chaser->pos.y - hhat.y * hbar;
xop.z = chaser->pos.z - hhat.z * hbar;

/*-----------------------------------------------------------------
 */
cos_theta = vector_dot(&xop, &target->pos) / mag_tpos / vector_magnitude(&xop);

if ((cos_theta > 1.0) && (cos_theta < 1.05))
     eta = 0.0;
else
     eta = acos(vector_dot(&xop,&target->pos) / mag_tpos / vector_magnitude(&xop));

vbar = mag_tpos * eta;
rbar = mag_tpos - vector_magnitude(&chaser->pos);

/*-----------------------------------------------------------------
 */
magvbar = vector_magnitude(&vb);

lcg[0][0] = vb.x / magvbar;  dif.x = chaser->pos.x - target->pos.x;
lcg[0][1] = vb.y / magvbar;  dif.y = chaser->pos.y - target->pos.y;
lcg[0][2] = vb.z / magvbar;  dif.z = chaser->pos.z - target->pos.z;

vector_transform(&dif, &rvbdif, lcg);

/*-----------------------------------------------------------------
 */
if (rvbdif.x <= 0)  vbar = -vbar;

rv->x = vbar;
rv->y = -hbar;
rv->z = rbar;

/*-----------------------------------------------------------------
 */
vdifi.x = chaser->vel.x - target->vel.x;
vdifi.y = chaser->vel.y - target->vel.y;
vdifi.z = chaser->vel.z - target->vel.z;

vector_transform(&vdifi, &vdifl, lcg);

w = magxxv / vector_dot(&target->pos, &target->pos);
omega.x =  omega.z  = 0.0;  omega.y = w;

vector_cross(&omega, &rvbdif, &wxr);

rdot.x = vdifl.x + wxr.x;
rdot.y = vdifl.y + wxr.y;
rdot.z = vdifl.z + wxr.z;
```

```
    /*-------------------------------------------------------------
     */
    rvdot->y = rdot.y;

    rvdot->x = vector_magnitude(&rvbdif);

    if (rvdot->x > 1.0e-8)        rvdot->z = -vector_dot(&rdot, &rvbdif) / rvdot->x;
    else                          rvdot->z = 0.0;
}
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#include <math.h>

#include <vector.h>

/*=============================================================
============================
 *-------------------------------------------------------------*/
double vector_magnitude(Vector *v)
{
   return sqrt(v->x * v->x + v->y * v->y + v->z * v->z);
}

int vector_unit(Vector *vin, Vector *vout)
{
   double mag;
   if ((mag = vector_magnitude(vin)) == 0.0)  return NO;
   vout->x = vin->x / mag;
   vout->y = vin->y / mag;
   vout->z = vin->z / mag;
   return YES;
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
double vector_dot(Vector *v1, Vector *v2)
{
   return (v1->x * v2->x + v1->y * v2->y + v1->z * v2->z);
}

void vector_cross(Vector *v1, Vector *v2, Vector *vout)
{
   vout->x = v1->y*v2->z - v1->z*v2->y;
   vout->y = v1->z*v2->x - v1->x*v2->z;
   vout->z = v1->x*v2->y - v1->y*v2->x;
}

/*=============================================================
============================
 *  This post-multiplies a matrix with a COLUMN vector: Vout = M Vin.
 *-------------------------------------------------------------*/
void vector_transform(Vector *vin, Vector *vout, double transform[3][3])
{
   vout->x = vector_dot(vin, (Vector *)&transform[0][0]);
   vout->y = vector_dot(vin, (Vector *)&transform[1][0]);
   vout->z = vector_dot(vin, (Vector *)&transform[2][0]);
}
```

```
/*=============================================================
=======================
 *-------------------------------------------------------------*/
void vector_diff(Vector *v1, Vector *v2, Vector *vout)
{
   vout->x = v1->x - v2->x;
   vout->y = v1->y - v2->y;
   vout->z = v1->z - v2->z;
}

//void vector_add(Vector *v1, Vector *v2, Vector *vout)
//{
//   vout->x = v1->x + v2->x;
//   vout->y = v1->y + v2->y;
//   vout->z = v1->z + v2->z;
//}
```

```
/*=============================================================
============================
 *  Public Domain
 *      Thomas J. Silva and Bruce G. Jackson & Associates, Inc.
 *
 *-------------------------------------------------------------*/
#ifndef _COM_PORT_
#define _COM_PORT_

#include <types.h>

/*=============================================================
============================
 *  These are the only user-callable routines, all the others are for internal
 * use only.
 *-------------------------------------------------------------*/
extern int
   com_port_start(int port_id, ushort rx_bsize, ushort tx_bsize),
   com_port_stop(int port_id),
   com_port_set(int port_id, long baud, int parity_flag, int stop_bits),
   com_port_get(int port_id),
   com_port_put(int port_id, ubyte value),
   com_port_read(int port_id, ubyte *where, int size),
   com_port_write(int port_id, ubyte *where, int size);

#define COM_PORT_par_none   0x00
#define COM_PORT_par_even   0x18
#define COM_PORT_par_odd    0x08

#define COM_PORT_stop_1             0x00
#define COM_PORT_stop_2             0x04

/*=============================================================
============================
 *  Head and tail are offsets from the buffer start.
 *-------------------------------------------------------------*/
typedef struct
{
   ushort size, used, head, tail;
   ubyte *start;
} Com_port_buffer;

/*=============================================================
============================
 *  This structure is used for actual interrupt-handler entry points.  The
 * structure contains several instructions, all of which must be initialized,
 * and one of which must be set with the common-handler address.
 *      06  push es
 *      53  push bx
 *      90  nop
```

```
 *       9a  callf SEG:OFF
 *-------------------------------------------------------------------------*/
typedef struct
{
   ubyte op_es, op_bx, op_nop, op_call;
   ushort offset, segment;
} Com_port_entry;

/*=========================================================================
==========================
 *  Note: If you change COM_PORT_max, then you must change an array in the C
 * source file.
 *-------------------------------------------------------------------------*/
#define COM_PORT_max                4

typedef struct
{
   Com_port_entry entry;
   ushort status, port;
   ubyte irq, irq_mask;
   Com_port_buffer rx, tx;
   void (far*old_handler)(void);
} Com_port_info;

#define COM_PORT_in_use             0x0001
#define COM_PORT_irq_was_off        0x0002
#define COM_PORT_rx_enabled  0x0010
#define COM_PORT_tx_enabled  0x0100
#define COM_PORT_tx_stopped  0x0200

#define COM_PORT_clock 1843200L

/*-------------------------------------------------------------------------*/
extern int
   _com_port_reset(Com_port_info *port),
   _com_port_get(Com_port_info *port),
   _com_port_put(Com_port_info *port, ushort value);

extern void
   _com_port_set(Com_port_info *port, ushort divisor, int parity_flag, int stop_bits),
   _com_port_intr_catch(Com_port_info *port),
   _com_port_intr_uncatch(Com_port_info *port);

#endif _COM_PORT_
```

```
%nolist
;=============================================================
============================
; *  Public Domain
; *          Thomas J. Silva and Bruce G. Jackson & Associates, Inc.
; *

;=============================================================
============================
;
union Far_ptr
        ptr             dd ?
        struc
                off             dw ?
                seg             dw ?
        ends
ends

struc Com_port_entry
        op_es           db ?
        op_bx           db ?
        op_nop          db ?
        op_call         db ?
        handler         Far_ptr < >
ends

OP_PUSH_ES   = 06h
OP_PUSH_BX   = 53h
OP_NOP              = 90h
OP_CALLF     = 9ah

;=============================================================
============================
;
struc Com_port_buffer
        bsize           dw ?
        bused           dw ?
        bhead           dw ?
        btail           dw ?
        bstart          Far_ptr < >
ends

struc Com_port_info
        entry           Com_port_entry < >
        status          dw ?
        port            dw ?
        irq                     db ?
        irq_mask        db ?
        rx                      Com_port_buffer < >
        tx                      Com_port_buffer < >
```

```
        old_handler     Far_ptr < >
ends

;-------------------------------------------------------------------
COM_PORT_in_use                 = 0001h
COM_PORT_irq_was_off        = 0002h
COM_PORT_rx_enabled         = 0010h
COM_PORT_tx_enabled         = 0100h
COM_PORT_tx_stopped         = 0200h

;=================================================================
============================
;
macro ACC_WR Reg,Inst,Val
        mov             ah,dl
        add             dl,Reg
        Inst    al,Val
        out             dx,al
        mov             dl,ah
endm

macro ACC_RD Reg
        mov             ah,dl
        add             dl,Reg
        in              al,dx
        mov             dl,ah
endm

;=================================================================
============================
;
PC_int_irq0     = 08h
PC_int_irq8     = 70h

PC_IC0_r0       = 20h
PC_IC0_r1       = 21h

PC_IC1_r0       = 0a0h
PC_IC1_r1       = 0a1h

IC_eoi          = 20h

;=================================================================
============================
;
ACC_rw1_int_mstat       = 00001000b
ACC_rw1_int_lstat       = 00000100b
ACC_rw1_int_tx              = 00000010b
ACC_rw1_int_rx              = 00000001b
```

```
ACC_r2_int_pending      = 00000001b
ACC_r2_int_type                 = 00000110b

ACC_w2_rx_fifo_1        = 00000000b
ACC_w2_rx_fifo_4        = 01000000b
ACC_w2_rx_fifo_8        = 10000000b
ACC_w2_rx_fifo_14       = 11000000b
ACC_w2_dma_mode                 = 00001000b
ACC_w2_tx_clr           = 00000100b
ACC_w2_rx_clr           = 00000010b
ACC_w2_fifo_on                  = 00000001b

ACC_rw3_dlab            = 10000000b
ACC_rw3_break                   = 01000000b
ACC_rw3_par_flip        = 00100000b
ACC_rw3_par_even        = 00010000b
ACC_rw3_par_enab        = 00001000b
ACC_rw3_2stop           = 00000100b
ACC_rw3_8bits           = 00000011b

ACC_rw4_loopback        = 00010000b
ACC_rw4_output2                 = 00001000b
ACC_rw4_output1                 = 00000100b
ACC_rw4_RTS                     = 00000010b
ACC_rw4_DTR                     = 00000001b

ACC_r5_tx_empty         = 00100000b
ACC_r5_rx_avail         = 00000001b

%list
```

```
/*=============================================================
============================
 *  This package provides mathematical constants.
 *
 * Note: Some of these are already defined, with different names, in math.h.
 *
 *  Author:
 *      Robert Gottlieb, August, 1989
 *
 *  Modified:
 *      Lee Barker, Mar 93
 *      TJ Silva, Mar 93
 *
 * Knuth, D.E.,"The Art of Computer Programing: Vol.2 Semi-Numerical
 * Algorithms," Addison-Wesley Pub. Co., 2nd Ed., 1981.
 *
 * Selby, S.M., "Standard Mathematical Tables," The Chemical Rubber Co.,
 * 20th ed., 18901 Cranwood Parkway, Cleveland Ohio, 1972.
 *
 *-------------------------------------------------------------------*/
#ifndef __CONST__
#define __CONST__

/* -- Define machine constants --- */

#define BIG             1.0e+20
#define SMALL           1.0e-20
#define TIME_TOL        1.0e-06

/* -- Defining Constants --*/
#define PI                      3.14159265358979323846
#define NATURAL_LOGARITHM_BASE  2.718281828459045235

#define  EULERS_NUMBER NATURAL_LOGARITHM_BASE;

/* -- Derived Constants --- */

#define SQUARE_ROOT_OF_TWO          1.41421356237309504880
#define SQUARE_ROOT_OF_THREE        1.732050807568877293
#define SQUARE_ROOT_OF_FIVE         2.236067977499789696
#define SQUARE_ROOT_TEN             3.162277660168379331

#define CUBE_ROOT_TWO               1.259921049894873164
#define CUBE_ROOT_THREE             1.442249570307408382
#define FOURTH_ROOT_TWO             1.189207115002721066

#define NATURAL_LOG_TWO             0.693147180559945309
#define NATURAL_LOG_THREE           1.098612288668109691
#define NATURAL_LOG_TEN             2.302585092994045684
#define NATURAL_LOG_PI              1.144729885849400174
```

```
#define BASE_TEN_LOG_PI              0.497149872694133854
#define PI_SQUARED                   9.869604401089358618
#define PI_INVERSE                   0.318309886183790671

#define HALF_PI                      ( PI / 2.0 )
#define TWO_PI                       ( PI * 2.0 )

#define NATURAL_LOGARITHM_BASE_INVERSE  0.367879441171442321
#define NATURAL_LOGARITHM_BASE_SQUARED  7.389056098930650227


#define DEGREES         (PI/180.0)
#define OBLIQUITY       (23.45 * DEGREES)

#define TRUE            1
#define FALSE           0

#define ON          1
#define OFF         0

/*****************************************************************************/
/* Additional constants added by Lee Barker for STS-51 software              */
/*****************************************************************************/
#define YES 1
#define NO  0

#define FT_TO_M                 0.3048
#define KILOFT_TO_KILOM     (FT_TO_M)
#define FT_TO_KILOM  (FT_TO_M/1000.0)

#define M_TO_FT                 (1.0/FT_TO_M)
#define KILOM_TO_KILOFT     (M_TO_FT)
#define KILOM_TO_FT  (M_TO_FT*1000.0)

#define RAD_TO_DEG (180.0/PI)
#define DEG_TO_RAD (PI/180.0)

#define EARTH_RADIUS_KILOM      6378.139
#define EARTH_OMEGA             7.29211514646e-5

#endif __CONST__
```

```
/*=============================================================
=========================
 *-------------------------------------------------------------*/
#ifndef _EDIT_
#define _EDIT_

extern char
   *edit_text(char *label, char *initial);

extern int
   edit_integer(char *label, int initial);

extern long
   edit_integer_long(char *label, long initial);

#endif _EDIT_
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#ifndef _EDIT_RT_
#define _EDIT_RT_

/*=============================================================
============================
 *-----------------------------------------------------------*/
typedef struct
{
   int row, col;
   char *string;
} Edit_label;

/*=============================================================
============================
 *  The field callback functions take these arguments:
 *
 *       get(char *work, void *data)
 *       put(char *work, void *data)
 *       key(Edit_field_info *info, int key)
 *
 *       work: pointer to start of work string, 80 chars max
 *       data: pointer to some piece of data, field specific
 *       info: pointer to field-in-progress information
 *       key:  key value
 *
 *  The get function must convert its data into a string.  It must return the
 * cursor offset as follows:
 *       < 0  Put the cursor past the end of the string
 *       >= 0  Put the cursor somewhere in the string (0 for start of string)
 *
 *  The put function must convert back from a string, it may return:
 *       0  No, string had garbage (reset my field)
 *       1  Ok, convert successful
 *
 *  The key function must behave somewhat like other key handlers:
 *       0  Didn't use the key
 *       1  Used the key
 *       2  Used the key (reset my field)
 *-----------------------------------------------------------*/
typedef struct
{
   char *work;
   int changed, size, cursor_offset;
} Edit_field_info;

typedef struct
{
```

```
    int row, col;
    int (*get)(char *, void *);
    int (*put)(char *,  void *);
    int (*key)(Edit_field_info *, int);
    void *data;
} Edit_field;

/*-----------------------------------------------------------------------*/
extern void
    edit_field_insert(Edit_field_info *info, int key);

extern int
    edit_field_key_handler(Edit_field_info *info, int key),

    edit_field_get_int(char *work, void *data),
    edit_field_put_int(char *work, void *data),
    edit_field_key_int(Edit_field_info *info, int key),

    edit_field_get_dbl(char *work, void *data),
    edit_field_put_dbl(char *work, void *data),
    edit_field_key_dbl(Edit_field_info *info, int key),

    edit_field_get_yesno(char *work, void *data),
    edit_field_put_yesno(char *work, void *data),
    edit_field_key_yesno(Edit_field_info *info, int key),

    edit_field_get_dhms0(char *work, void *data),
    edit_field_put_dhms0(char *work, void *data),
    edit_field_get_dhms1(char *work, void *data),
    edit_field_put_dhms1(char *work, void *data),
    edit_field_key_dhms(Edit_field_info *info, int key);


/*=======================================================================
=======================
 *  The screen callback functions take these arguments:
 *
 *      pre(Edit_screen *screen)
 *      post(Edit_screen *screen)
 *
 *  The screen callback functions give you control just before the screen goes
 * up, and just after it comes down.  The primary use for these callbacks is to
 * modify all the fields at once, specially when several fields are related.
 *-----------------------------------------------------------------------*/
typedef struct _edit_screen
{
    int label_count;  Edit_label *labels;
    int field_count;  Edit_field *fields;

    void *data;
```

```
    int changed;
    void (*pre)(struct _edit_screen *);
    void (*post)(struct _edit_screen *);

    struct { Edit_field *ptr;  Edit_field_info info; } current_field;
} Edit_screen;

/*-----------------------------------------------------------------------*/
extern void
    edit_screen_init(Edit_screen *screen),
    edit_screen_stop(void);

extern int
    edit_screen_key_handler(Edit_screen *screen, int key);

#endif _EDIT_RT_
```

```
/*=============================================================
============================
 *  Public Domain
 *      Thomas J. Silva and The Telemetry Workshop, Inc.
 *
 *-------------------------------------------------------------*/
#if !defined(_EMM_)
#define _EMM_

#include <types.h>

/*=============================================================
============================
 *  Here are some constants / enumerations / structures, and the prototypes of
 * the low-level EMM calls.
 *-------------------------------------------------------------*/
#define EMM_INT             0x67
#define EMM_VER4    0x40
#define EMM_PAGE_SIZE       (16*1024)
#define EMM_PAGE_PARAS      (EMM_PAGE_SIZE/16)

typedef struct { ushort seg, page; } Emm_phys_map;

typedef struct { ushort page, phys; } Emm_map;

enum emm_phys_type { EMM_phys_page, EMM_phys_segment };

/*-------------------------------------------------------------*/
extern int
   _emm_version(ushort *version),
   _emm_pages(ushort *total, ushort *free),
   _emm_frame_seg(ushort *seg),
   /* _emm_phys_pages(ushort *pages), */
   /* _emm_phys_maps(Emm_phys_map *maps, ushort *pages), */
   _emm_alloc(ushort *handle, ushort pages),
   _emm_realloc(ushort handle, ushort pages),
   _emm_free(ushort handle),
   _emm_map(ushort phys_page, ushort handle, ushort page),
   _emm_map_multi(int type, ushort handle, ushort count, Emm_map *maps),
   _emm_name_get(ushort handle, char *name),
   _emm_name_set(ushort handle, char *name),
   _emm_name_find(ushort *handle, char *name),
   _emm_handle_pages(ushort handle, ushort *pages);

/*=============================================================
============================
 *  An Emm_info structure holds information about the EMM system.  You should
 * make just one of these in your program.
 *
 *  The element .phys.pages will be 0 if EMM not present, is an old version, or
```

```
 * fails in some other manner during emm_info_construct().
 *
 *  NOTES:
 *
 *   The Emm_mapped structure, and the array of these maps in the Emm_info, were
 * from an earlier version of this code.  This version does not try to keep
 * track of things as did the previous version.
 *
 *   This version's Emm_info has the array .map_0123, which you can use to map
 * the first four pages of any handle into the four pages of the page-frame.
 *-----------------------------------------------------------------------*/
/*--- typedef struct { struct _emm_handle *emm; ushort page; } Emm_mapped; ---*/

typedef struct
{
   ushort pages_free, pages_total;
   struct { ushort pages, segments[4]; } phys;
   /*--- Emm_mapped mapped[4]; ---*/
   int last_error;
   Emm_map map_0123[4];
} Emm_info;

/*-----------------------------------------------------------------------*/
extern void
   emm_info_construct(Emm_info *this);

#define emm_info_exists(this) ((this)->phys.pages != 0)

/*=======================================================================
=======================
 *   An Emm_handle structure holds information about one EMM handle.  Make one
 * of these for every EMM allocation.
 *-----------------------------------------------------------------------*/
typedef struct _emm_handle
{
   ushort handle;
   Emm_info *info;
   ushort pages;
   /*--- ushort pages_mapped; ---*/
} Emm_handle;

/*-----------------------------------------------------------------------*/
extern void
   emm_construct_new(Emm_handle *this, Emm_info *info, ushort pages),
   emm_construct_known(Emm_handle *this, Emm_info *emm_info, ushort handle),
   emm_destroy(Emm_handle *this);

#define emm_exists(this) ((this)->handle != 0)

#define emm_name_get(this,name)  _emm_name_get((this)->handle, name)
```

```
#define emm_name_set(this,name)  _emm_name_set((this)->handle, name)

extern int
   emm_map(Emm_handle *this, ushort phys_page, ushort page),
   emm_map_multi(Emm_handle *this, int type, ushort count, Emm_map *maps);

#endif _EMM_
```

```
/*=============================================================
==========================
 *-------------------------------------------------------------*/
#ifndef _KEYS_
#define _KEYS_

extern int key_hit(void);
extern int key_get(void);
extern unsigned int key_flags(void);

#define KEY_flag_lshift  0x0002
#define KEY_flag_rshift  0x0001
#define KEY_flag_shift   0x0003

#define KEY_space        0x0020
#define KEY_esc                  0x001b
#define KEY_cr           0x000d
#define KEY_lf           0x000a
#define KEY_bs           0x0008
#define KEY_del                  0x0153

#define KEY_tab                  0x0009
#define KEY_backtab      0x010f

#define KEY_f1           0x013b
#define KEY_f2           0x013c
#define KEY_f3           0x013d
#define KEY_f4           0x013e
#define KEY_f5           0x013f
#define KEY_f6           0x0140
#define KEY_f7           0x0141
#define KEY_f8           0x0142
#define KEY_f9           0x0143
#define KEY_f10                  0x0144

#define KEY_shft_f1      0x0154
#define KEY_shft_f2      0x0155
#define KEY_shft_f3      0x0156
#define KEY_shft_f4      0x0157
#define KEY_shft_f5      0x0158
#define KEY_shft_f6      0x0159
#define KEY_shft_f7      0x015a
#define KEY_shft_f8      0x015b
#define KEY_shft_f9      0x015c
#define KEY_shft_f10     0x015d

#define KEY_cntl_f1      0x015e
#define KEY_cntl_f2      0x015f
#define KEY_cntl_f3      0x0160
#define KEY_cntl_f4      0x0161
```

```
#define KEY_cntl_f5     0x0162
#define KEY_cntl_f6     0x0163
#define KEY_cntl_f7     0x0164
#define KEY_cntl_f8     0x0165
#define KEY_cntl_f9     0x0166
#define KEY_cntl_f10    0x0167

#define KEY_alt_f1      0x0168
#define KEY_alt_f2      0x0169
#define KEY_alt_f3      0x016a
#define KEY_alt_f4      0x016b
#define KEY_alt_f5      0x016c
#define KEY_alt_f6      0x016d
#define KEY_alt_f7      0x016e
#define KEY_alt_f8      0x016f
#define KEY_alt_f9      0x0170
#define KEY_alt_f10     0x0171

#define KEY_right       0x014d
#define KEY_left        0x014b
#define KEY_up                  0x0148
#define KEY_down        Cx0150
#define KEY_home        0x0147
#define KEY_end                 0x014f
#define KEY_pgup        0x0149
#define KEY_pgdn        0x0151

#define KEY_cntl_right  0x0174
#define KEY_cntl_left   0x0173
#define KEY_cntl_up     0x018d
#define KEY_cntl_down   0x0191

#define KEY_alt_r       0x0113
#define KEY_alt_s       0x011f
#define KEY_alt_t       0x0114
#define KEY_alt_w       0x0111

#define YES 1
#define NO  0

#endif _KEYS_
```

```
/*=============================================================
=======================
 *-------------------------------------------------------------*/
#ifndef _NPS_IF_
#define _NPS_IF_

#include <vector.h>

/*=============================================================
=======================
 *-------------------------------------------------------------*/
extern int
    nps_system_init(void),
    nps_key_handler(int),
    nps_display_init(void);

extern void
    nps_system_stop(void),
    nps_data_flush(void),
    nps_data_update(State_vector *orb_ins_m50,
                    Quaternion *orb_quat,
                    State_vector *orb_targ_m50,
                    State_vector *orb_gps_m50,
                    State_vector *orb_gps_ecef,
                    State_vector *spas_sv_ecef,
                    State_vector *spas_gps_ecef),
    nps_process_data(int have_orb_gps, int orb_pkt_type);

#endif _NPS_IF_
```

```
/*=============================================================================
============================
 *-----------------------------------------------------------------------*/
#ifndef __ORBMECH1__
#define __ORBMECH1__

#include <const.h>
#include <vector.h>

typedef struct
{
   double planet_mu;
   double planet_radius;
   double planet_omega;
   double planet_flattening;
   double planet_j2;
   double cc[530];
   double ss[530];
} Gravity_data;

typedef struct
{
   int mode;
   int number_zonal;
   int number_tesseral;
   Gravity_data GMD;
} Gravity_model_data;

typedef struct
{
   int    mode;
   double solar_flux;
   double mean_solar_flux;
   double geomagnetic_index_ap;
   double ballistic_number;
} Drag_model_data;


typedef struct
{
   Gravity_model_data Gravity;
   Drag_model_data Drag;
} Perturbations;

/*-----------------------------------------------------------------------*/
extern int
   bgprop_init(void);

extern void
   bgprop_year(int year),
```

```
    bgprop(double time_total, Vector *pos, Vector *vel, Perturbations *P,
           double delta_t, double step, Vector *posf, Vector *velf),
    instantiate_gotpot(int model_name, Perturbations *P),
    m50_wgs84_m50(State_vector *svin, State_vector *svout,
                  int direction, State_vector_transform *svt, int vel_is_ecef);

#endif __ORBMECH1__
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#ifndef _PACKET_
#define _PACKET_

#include <vector.h>

/*=============================================================
============================
 *  The packet start sync is EB 90, the stop sync is BE EF.  The type is two
 * bytes instead of one to insure even-byte alignment for the remaining parts of
 * the structure.
 *-------------------------------------------------------------*/
typedef struct
{
   ubyte sync_start[2];
   int type;
} Packet_header;

typedef struct { ubyte sync_stop[2]; } Packet_trailer;

enum packet_types { PKT_orbiter = 1, PKT_spas, PKT_gps };

/*=============================================================
============================
 *  These packets are meant to go from one PC to another.  The receiving PC
 * program may not know where the data comes from, nor should it have to know.
 *
 *  Therefore the sending machine, regardless of its architecture, must build
 * the packets with this format:
 *       All multi-byte elements are in Intel order: LSB first (little-endian).
 *       Doubles are IEEE format.
 *
 *  In addition, the data must be converted in this manner:
 *       Times are floating-point seconds of GMT, note that 001/00:00:01.000
 *        (1 sec past midnight on Jan 1st) is 1 sec + 86400 (1 day).
 *       Angles are in degrees.
 *       Orbiter state vectors are M50, in ft and ft/s.
 *       Spas state vectors are WGS84, in ft and ft/s.
 *
 * Note: packet type 3 (PKT_gps) is used only for testing.
 *-------------------------------------------------------------*/
typedef struct
{
   double range, roll_azimuth, pitch_elevation, range_rate;
   ushort range_good, angles_good, rate_good;
} Ku_info;

typedef struct
```

```
{
   Packet_header header;
   State_vector orbiter_vector;
   Quaternion orbiter_quat;
   Ku_info ku;
   State_vector target_vector;
   Packet_trailer trailer;
} Orb_packet;

typedef struct
{
   Packet_header header;
   State_vector spas_vector, gps_vector;
   Packet_trailer trailer;
} Spas_packet;

typedef struct
{
   Packet_header header;
   State_vector gps_vector;
   Packet_trailer trailer;
} Orb_gps_packet;

#endif _PACKET_
```

```
/*=============================================================
==========================
 *-----------------------------------------------------------*/
#ifndef _PKT_IF_
#define _PKT_IF_

#include <vector.h>

/*=============================================================
==========================
 *-----------------------------------------------------------*/
extern void
   orb_port_packet_check(int on_off);

extern int
   orb_port_init(void),
   orb_port_init_byname(char *filename),
   orb_port_stop(void),
   orb_port_getb(void),
   orb_port_read(unsigned char *where, int bytes),
   orb_port_process(void);

extern int
   tans_port_init(void),
   tans_port_init_byname(char *filename),
   tans_port_stop(void),
   tans_port_getb(void),
   tans_port_sendb(unsigned char value),
   tans_port_read(unsigned char *where, int bytes),
   tans_port_process(void);

#endif _PKT_IF_
```

```
/*=============================================================
==========================
 *-------------------------------------------------------------*/
#if !defined(_SWAP_)
#define _SWAP_

/*=============================================================
==========================
 *-------------------------------------------------------------*/
extern void   swap_double(void *to, void *from);
extern double swap_double_out(void *from);

extern void  swap_float(void *to, void *from);
extern float swap_float_out(void *from);

extern void   swap_short(void *to, void *from);
extern ushort swap_short_out(void *from);

#endif _SWAP_
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#ifndef _TANS_PKT_
#define _TANS_PKT_

#include <vector.h>

/*=============================================================
============================
 *-------------------------------------------------------------*/
typedef struct { float  x, y, z; }                    Vector_float;

typedef struct { ubyte dle, type; } Tans_pkt_head;
typedef struct { ubyte dle, etx; }  Tans_pkt_tail;

/*-------------------------------------------------------------*/
typedef struct
{
   float secs_of_week;
   int week;
   float gps_utc_offset;
} Tans_pkt_21_41;

/*-------------------------------------------------------------*/
typedef struct
{
   ubyte dyn_code;
   float elev_angle_mask, signal_lvl_mask, pdop_mask, pdop_switch;
} Tans_pkt_2c_4c;

/*-------------------------------------------------------------*/
typedef struct { ubyte position, velocity, timing, auxiliary; } Tans_pkt_35_55;

/*-------------------------------------------------------------*/
typedef struct { Vector_float vel; float gigo; float sol_time; } Tans_pkt_43;

typedef struct { Vector pos; double gigo; float sol_time; } Tans_pkt_83;

/*-------------------------------------------------------------*/
typedef struct { ubyte satellite; } Tans_pkt_3c;

typedef struct
{
   ubyte sat_prn, slot_code, acq_flag, ephem_flag;
   float signal_level, time, elevation, azimuth;
   ubyte old_meas, msec_flag, data_bad_flag, data_collect_flag;
} Tans_pkt_5c;

/*-------------------------------------------------------------*/
```

```
typedef struct
{
   ubyte control, ant_select, carrier_track, diff_track;
} Tans_pkt_e0_e1;

/*-----------------------------------------------------------------*/
typedef struct { ubyte snr; int dif_phase; } Tans_pkt_e4_slave;

typedef struct
{
   ubyte slot_code, sat_prn;
   int meas_count;
   ubyte snr;
   float code_phase, doppler;
   double time;
   ulong carrier_beat;
   ubyte master_index;
   Tans_pkt_e4_slave slave_ant[3];
} Tans_pkt_e4;

/*=================================================================
=========================
 *-----------------------------------------------------------------*/
#define TANS_PKT_SIZE_RAW 512
#define TANS_PKT_SIZE_COOKED      256

typedef enum
{
   TANS_state__find_start, TANS_state__had_gigo, TANS_state__packet_start,
   TANS_state__find_end, TANS_state__found_dle, TANS_state__packet_bad,
   TANS_state__packet_ok, TANS_state__oversize
} Tans_packet_state;

typedef struct
{
   ubyte raw[TANS_PKT_SIZE_RAW], cooked[TANS_PKT_SIZE_COOKED];
   int raw_size, cooked_size, state;
} Tans_packet;

/*-----------------------------------------------------------------*/
extern int
   tans_packet_recv(Tans_packet *packet);

#endif _TANS_PKT_
```

```
/*=============================================================
============================
 *  Public Domain
 *      Thomas J. Silva and Bruce G. Jackson & Associates, Inc.
 *
 *-------------------------------------------------------------*/
#if !defined(_TIMES_)
#define _TIMES_

/*=============================================================
============================
 *-------------------------------------------------------------*/
typedef struct { unsigned short days, hours, mins, secs, msecs; } Time_dhms;

typedef struct { int week; double secs; } Time_gps;

typedef struct { int year; double secs; } Time_utc;

/*=============================================================
============================
 * WARNING: These routines are not coded to handle negative time!
 *
 *  The string routines make this fixed size string: "ddd:hh:mm:ss.123", and
 * need 18 bytes with which to work, so size your work area accordingly.
 *-------------------------------------------------------------*/
extern char
   *time_dhms_to_string(Time_dhms *dhms, char *work),
   *time_dbl_to_string(double *time, char *work);

extern void
   time_dbl_to_dhms(double *time, Time_dhms *dhms),
   time_dhms_to_dbl(Time_dhms *dhms, double *time),
   time_gps_to_utc(Time_gps *gps, double gps_leads_utc, Time_utc *utc);

extern void
   rtc_timer_set(unsigned short msec_count, void *flag);

#endif _TIMES_
```

```
%nolist
;============================================================
============================
; *  Public Domain
; *              Thomas J. Silva and Bruce G. Jackson & Associates, Inc.
; *

;============================================================
============================
;
struc time_sec_msec
        secs            dd ?
        msecs           dw ?
ends

struc time_sec_nsec
        secs            dd ?
        nsecs           dd ?
ends

struc time_dhms
        days            dw ?
        hours           dw ?
        mins            dw ?
        secs            dw ?
        msecs           dw ?
ends

%list
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#ifndef _TYPES_
#define _TYPES_

/*=============================================================
============================
 *-------------------------------------------------------------*/
typedef unsigned long    ulong;
typedef unsigned short   ushort;
typedef unsigned char    ubyte;
typedef char             byte;

#define YES 1
#define NO  0

#if !defined(FP_OFF)
#define MK_FP(seg,off)  ((void far *)(((ulong)(seg) << 16) | (ushort)(off)))
#endif

#endif _TYPES_
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#ifndef _VECTOR_
#define _VECTOR_

#include <types.h>

/*=============================================================
============================
 *  I define the quaternion rotation as follows: positive twist is counter-
 * clockwise when looking down the rotation vector towards the origin.  This is
 * consistent with a right-hand rotation where the thumb points out along the
 * rotation vector.  This is also the way the books describe it.
 *
 *  NASA apparently defines positive rotation clockwise, thus all names and
 * labels are "backwards" when used with this code.  For example: "M50 to Body"
 * must be treated as a "Body to M50" rotation.  However, just to confuse us,
 * the names used by the onboard software appear to be correct: the onboard name
 * "Q_BI" corresponds to the ground name "M50 to body".  Go figure.
 *
 *  The physical layout of the quaternion structure corresponds to the Q1-Q4
 * as used by NASA: Q1 is the scalar, Q2 is the x component, etc.
 *-----------------------------------------------------------------*/
typedef struct { double x, y, z; }                  Vector;

typedef struct { double s, x, y, z, time; }         Quaternion;

typedef struct { Vector pos, vel;  double time; }   State_vector;

typedef struct { Vector pos;  double time; }            RV_vector;

// typedef struct { double roll, pitch, yaw; }          Euler_angles;

typedef struct { double low, hi; }                  Vector_limits;
typedef struct { Vector_limits pos, vel; }          State_vector_limits;

/*-----------------------------------------------------------------*/
extern double
   vector_magnitude(Vector *v),
   vector_dot(Vector *v1, Vector *v2);

extern void
   vector_diff(Vector *v1, Vector *v2, Vector *vout),
   vector_add(Vector *v1, Vector *v2, Vector *vout),
   vector_cross(Vector *v1, Vector *v2, Vector *vout),
   vector_transform(Vector *vin, Vector *vout, double transform[3][3]),
   vector_rvbar(State_vector *target, State_vector *chaser, Vector *rv, Vector *rvdot),
   matrix_transpose(double matrix_in[3][3], double matrix_out[3][3]),
   // matrix_pyr(double m[3][3], Euler_angles *euler),
```

```
    quaternion_conjugate(Quaternion *q),
    quaternion_to_matrix(Quaternion *q, double matrix[3][3]),
    quaternion_mult(Quaternion *q1, Quaternion *q2, Quaternion *qout),
    // quaternion_pyr(Quaternion *q, Euler_angles *euler),
    quaternion_normalize(Quaternion *q);

extern int
    vector_unit(Vector *vin, Vector *vout),
    vector_check(Vector *v, Vector_limits *limit),
    matrix_to_quaternion(double matrix[3][3], Quaternion *q),
    state_vector__itol_matrix(State_vector *sv, double matrix[3][3]),
    state_vector__check(State_vector *v, State_vector_limits *limit);

/*=============================================================
==========================
 *  This structure is used to transform one state vector into another.  Some
 * transformations need a time stamp on the matrix.
 *-------------------------------------------------------------*/
typedef struct { double matrix[3][3], time; }          State_vector_transform;

/*-------------------------------------------------------------*/
extern void
    state_vector__rotate_m50_ecef(State_vector_transform  *m50_ecef,
                                  double new_time,
                                  State_vector_transform *ecef_m50),
    state_vector__m50_to_ecef(State_vector  *m50, State_vector *ecef,
                              State_vector_transform *m50_ecef_epoch, int vel_is_ecef),
    state_vector__ecef_to_m50(State_vector  *ecef, State_vector *m50,
                              State_vector_transform *m50_ecef_epoch, int vel_is_ecef),
    state_vector__m50_to_j2000(State_vector *m50, State_vector *j2000),
    state_vector__j2000_to_m50(State_vector *j2000, State_vector *m50);

#endif _VECTOR_
```

```
/*=============================================================
==========================
 *-----------------------------------------------------------*/
#include <stdio.h>
#include <alloc.h>
#include <conio.h>

#include <edit.h>
#include <keys.h>
#include <com_port.h>

/*=============================================================
==========================
 *-----------------------------------------------------------*/
static int port_id, packet_size;

static struct
{
   int in_use, yline, next;
   ubyte *packet;
   ushort cnt;
} tx, rx;

#define INFO_X 5

/*=============================================================
==========================
 *-----------------------------------------------------------*/
static int
   init(void),
   recv_next(void),
   recv_chk(void);

static void
   send_next(void),
   recv_dump(void);

/*=============================================================
==========================
 *-----------------------------------------------------------*/
main(int argc, char **argv)
{
   static char msg[] =
         "\nTest keys:\n"
         "  t  Toggles packet send\n"
         "  r  Toggles packet recv\n"
         " ESC  Stop the test\n"
         "\n"
         " TX\n"
         " RX\n"
```

```
        "Both rx and tx are now off, Hit any key to begin transfer test";

    /*-------------------------------------------------------------------
     */
    if (init() == NO)  return;

    puts(msg);  key_get();
    gotoxy(1, wherey()-1);  clreol();
    tx.yline = (rx.yline = wherey() - 1) - 1;

    for (;;)
    {
        if (key_hit())
        {
            ushort key = key_get();
            if (key == 't')                 tx.in_use = (tx.in_use ? NO : YES);
            else if (key == 'r')   rx.in_use = (rx.in_use ? NO : YES);
            else if (key == KEY_esc)      break;
        }
        if (tx.in_use) send_next();
        if (rx.in_use  &&  recv_next() == NO)  break;
    }

    com_port_stop(port_id);


/*=====================================================================
=====================
 *-------------------------------------------------------------------*/
static int init()
{
    ushort rx_size, tx_size;
    long baud;
    int parity, stops;

    /*-------------------------------------------------------------------
     */
    port_id = edit_integer("Enter port id ", 1);
    rx_size = edit_integer("Enter rx buffer size ", 512);
    tx_size = edit_integer("Enter tx buffer size ", 512);

    if (com_port_start(port_id, rx_size, tx_size) == NO)
    {
        printf("Can't start port %d\n", port_id);
        port_id = 0;
        return NO;
    }

    /*-------------------------------------------------------------------
     */
```

```
    baud = edit_integer_long("Enter baud rate ", 9600);

    parity = edit_integer("Enter parity (0 none, 1 even, 2 odd) ", 2);
    parity = (parity == 0
      ? COM_PORT_par_none
      : (parity == 1
        ? COM_PORT_par_even
          : COM_PORT_par_odd));

    stops  = edit_integer("Enter stop bits, 1 or 2 ", 1);
    stops = (stops == 1 ? COM_PORT_stop_1 : COM_PORT_stop_2);

    com_port_set(port_id, baud, parity, stops);
    com_port_get(port_id);

    /*-------------------------------------------------------------------
     */
    packet_size = edit_integer("Enter packet size ", 32);
    if (packet_size < 4)  packet_size = 4;

    tx.packet = malloc(packet_size);  rx.packet = malloc(packet_size);
    if (tx.packet == NULL || rx.packet == NULL)
    {
          com_port_stop(port_id);
          return NO;
    }
    else
    {
          int cnt;
          memset(tx.packet, 0, packet_size);
          tx.packet[0] = 0xde;  tx.packet[1] = 0xad;          // dead
          tx.packet[2] = 0xbe;  tx.packet[3] = 0xef;          // beef
          for (cnt = 4;  cnt < packet_size;  cnt++)
             tx.packet[cnt] = (ubyte)cnt;
    }

    return YES;
}

/*==========================================================
==========================
 *-----------------------------------------------------------*/
static void send_next()
{
    if (com_port_put(port_id, tx.packet[tx.next]) == YES)
    {
          if (++tx.next >= packet_size)
          {
             gotoxy(INFO_X, tx.yline);  cprintf("%5u", tx.cnt++);
             tx.next = 0;
```

```
        }
    }
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static int recv_next()
{
    int value;

    if ((value = com_port_get(port_id)) != -1)
    {
        rx.packet[rx.next] = (ubyte)value;
        if (++rx.next >= packet_size)
        {
            if (recv_chk() == NO)
            {
                printf("\nrx %u failed:", rx.cnt);
                recv_dump();
                return NO;
            }
            gotoxy(INFO_X, rx.yline);  cprintf("%5u", rx.cnt++);
            rx.next = 0;
        }
    }
    return YES;
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static int recv_chk()
{
    int cnt;

    if (rx.packet[0] != 0xde  ||  rx.packet[1] != 0xad
     || rx.packet[2] != 0xbe  ||  rx.packet[3] != 0xef)  return NO;

    for (cnt = 4;  cnt < packet_size;  cnt++)
        if (rx.packet[cnt] != (ubyte)cnt)  return NO;

    return YES;
}

static void recv_dump()
{
    int cnt;

    for (cnt = 0;  cnt < packet_size;  cnt++)
```

```
    {
        if ((cnt % 16) == 0)  printf("\n %04x ", cnt);
        printf(" %02x", rx.packet[cnt]);
    }
    fputc('\n', stdout);
}
```

```
/*=============================================================
============================
 */
#include <stdio.h>
#include <stdlib.h>
#include <mem.h>
#include <string.h>

#include <emm.h>

/*=============================================================
============================
 *  These structures are global because, well, they need to be!
 */
Emm_info emm_info;
Emm_handle emm_handle;

/*=============================================================
============================
 *-------------------------------------------------------------*/
main(int argc, char **argv)
{
   static char *errs[] =
   {
        "EMM ok",
        "EMM init failed, error %d\n",
        "EMM handle failed, error %d\n",
        "EMM map failed, error %d\n",
        "Usage: emm_name [pages]\n"
   };

   int rc;

   /*-------------------------------------------------------------
    *  Go ahead, make my mem!
    */
   if (argc < 2) { puts(errs[4]);  return; }

   rc = make_my_emm(argv[1], argv[2]);

   if (rc != 0) { printf(errs[rc], emm_info.last_error);  return; }

   /*-------------------------------------------------------------
    */
   printf("EMM handle %d <%s> has %d pages, and is mapped at %04x\n",
     emm_handle.handle, argv[1], emm_handle.pages, emm_info.phys.segments[0]);

   if (yes_no("Delete this handle ?"))  emm_destroy(&emm_handle);
}
```

```
/*=============================================================================
=============================
 * returns:
 * 0    ok
 * 1    Can't find EMM on this machine
 * 2    Can't create memory handle
 * 3    Can't map the memory into physical page 0
 *-----------------------------------------------------------------------*/
make_my_emm(char *raw_name, char *make_pages)
{
   char name[9];
   ushort pages, handle;

   /*--------------------------------------------------------------------
    *  Fill in the emm_info structure, make sure EMM exists on this machine.
    */
   emm_info_construct(&emm_info);

   if (! emm_info_exists(&emm_info))  return 1;

   /*--------------------------------------------------------------------
    *  Look for an existing named memory chunk.  If it doesn't exist then make
    * a new chunk of size 1 (16K), and give it a name.
    */
   memset(name, 0, sizeof(name));
   strncpy(name, raw_name, 8);

   if (_emm_name_find(&handle, name) != 0)
   {
        if ((pages = atoi(make_pages)) > emm_info.pages_free)
           pages = emm_info.pages_free;
        emm_construct_new(&emm_handle, &emm_info, pages);
        if (emm_exists(&emm_handle))  emm_name_set(&emm_handle, name);
   }
   else  emm_construct_known(&emm_handle, &emm_info, handle);

   if (! emm_exists(&emm_handle))  return 2;

   /*--------------------------------------------------------------------
    *  Map the memory somewhere (use the first physical page, page 0).
    */
   if (emm_map(&emm_handle, 0, 0) != 0)  return 3;

   return 0;
}

/*=============================================================================
=============================
 *-----------------------------------------------------------------------*/
yes_no(char *prompt)
```

```
{
   char ans[80];

   fputs(prompt, stdout);
   ans[0] = 0;  scanf("%s", ans);

   return (ans[0] == 'y'  ||  ans[0] == 'Y');
}
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

/*=============================================================
============================
 *-----------------------------------------------------------*/
int main(int argc, char **argv)
{
   static char
        msg1[] = "Usage: %s map_text map_raw\n",
        msg2[] = "Can't open file <%s>\n";

   FILE *in, *out;

   /*------ ---------------------------------------------------
    */
   if (argc < 3)
   {
        printf(msg1, argv[0]);
        return;
   }

   if ((in = fopen(argv[1], "rt")) == NULL)
   {
        printf(msg2, argv[1]);
        return;
   }

   if ((out = fopen(argv[2], "wb")) == NULL)
   {
        printf(msg2, argv[2]);
        return;
   }

   /*-----------------------------------------------------------
    */
   for (;;)
   {
        struct { unsigned char cmd;  float longitude, latitude; } point;
        int rc;

        rc = fscanf(in, "%d %f %f", &point.cmd, &point.longitude, &point.latitude);
        if (rc < 3)  break;
        fwrite(&point, sizeof(point), 1, out);
   }
}
```

```
/*=============================================================
============================
 *  I clean out packets from a file.  I filter packets whose times are too
 * close or too far apart.
 *  Usage: pkt_file_in file_out  pkt_type dt_lt dt_gt
 *-------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>

#include <packet.h>
#include <keys.h>
#include <pkt_if.h>


/*=============================================================
============================
 *  These structures are owned by the orb_port_process() routine, and are in
 * the original units: orbiter vectors are M50 in ft and ft/s, spas vectors are
 * WGS84 in ft and ft/s.
 *
 * The orbiter time is absolute, 001/00:00:01.000 == 1.0 sec + 86400 secs (one
 * day).  The spas time is also absolute.
 *-------------------------------------------------------------*/
extern Orb_packet orbiter_packet;
extern Spas_packet spas_packet;

/*=============================================================
============================
 *  This structure is also owned by the orb_port_process() routine, and is only
 * used for testing.  The vector is WGS84 in m and m/s.  Its time is absolute.
 *-------------------------------------------------------------*/
extern Orb_gps_packet orbiter_gps_packet;


/*=============================================================
============================
 *-------------------------------------------------------------*/
static FILE *file_rx, *file_tx;
static double dt_lt, dt_gt;

static char usage[] =
   "Usage: %s pkt_file_in file_out  pkt_type dt_lt dt_gt\n"
   "    file_in          Input file with packets\n"
   "    file_out Output file for filtered packets\n"
   "    pkt_type         Packet type to filter\n"
   "    dt_lt            Throw out packets < dt_lt\n"
   "    dt_gt            Throw out packets > dt_gt\n";

#define BUFFER_SIZE 0x7000
```

```
/*=============================================================
============================
 *  The array follows the packet_types enumeration.
 *-------------------------------------------------------------*/
typedef struct { int got_one, size;  void *packet;  double time, *new_time; } Time_info;

static Time_info time_infos[] =
{
   { 0 },
   { NO, sizeof(Orb_packet),     &orbiter_packet,     0.0, &orbiter_packet.orbiter_vector.time },
   { NO, sizeof(Spas_packet),    &spas_packet,        0.0, &spas_packet.spas_vector.time       },
   { NO, sizeof(Orb_gps_packet), &orbiter_gps_packet, 0.0, &orbiter_gps_packet.gps_vector.time }
};

/*=============================================================
============================
 *-------------------------------------------------------------*/
main(int argc, char **argv)
{
   int filter_type, type;

   /*-----------------------------------------------------------
    */
   if (argc < 6) { printf(usage, argv[0]);  return; }

   if ((file_rx = fopen(argv[1], "rb")) == NULL)
   {
        printf("Can't open file <%s>\n", argv[1]);
        return;
   }
   setvbuf(file_rx, NULL, _IOFBF, BUFFER_SIZE);

   if ((file_tx = fopen(argv[2], "wb")) == NULL)
   {
        printf("Can't open file <%s>\n", argv[2]);
        return;
   }
   setvbuf(file_tx, NULL, _IOFBF, BUFFER_SIZE);

   filter_type = atoi(argv[3]);
   dt_lt = atof(argv[4]);
   dt_gt = atof(argv[5]);

   /*-----------------------------------------------------------
    */
   while ((type = orb_port_process()) > 0  &&  type <= PKT_gps)
   {
        Time_info *info = time_infos + type;

        if (type == filter_type)
```

```
        {
            if (info->got_one)
            {
                if (*info->new_time < info->time + dt_lt
                 || *info->new_time > info->time + dt_gt)  continue;
            }
            else  info->got_one = YES;

            info->time = *info->new_time;
        }

        fwrite(info->packet, info->size,1, file_tx);
    }

    fclose(file_rx);
    fclose(file_tx);
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
int orb_port_getb() { return getc(file_rx); }
```

```
/*=============================================================
============================
 *  I show packets from a file.
 *  Usage: pkt_file
 *-----------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>

#include <packet.h>
#include <keys.h>
#include <pkt_if.h>


/*=============================================================
============================
 *  These structures are owned by the orb_port_process() routine, and are in
 * the original units: orbiter vectors are M50 in ft and ft/s, spas vectors are
 * WGS84 in ft and ft/s.
 *
 * The orbiter time is absolute, 001/00:00:01.000 == 1.0 sec + 86400 secs (one
 * day).  The spas time is also absolute.
 *-----------------------------------------------------------*/
extern Orb_packet orbiter_packet;
extern Spas_packet spas_packet;

/*=============================================================
============================
 *  This structure is also owned by the orb_port_process() routine, and is only
 * used for testing.  The vector is WGS84 in m and m/s.  Its time is absolute.
 *-----------------------------------------------------------*/
extern Orb_gps_packet orbiter_gps_packet;

extern int orb_port_pkt_error;


/*=============================================================
============================
 *-----------------------------------------------------------*/
static FILE *file_rx;

static char usage[] = "Usage: %s pkt_file\n";

#define BUFFER_SIZE 0x7000

static void
   dump_packet(int type),
   dump_vector(State_vector *sv, char *msg);

/*=============================================================
============================
```

```
 *-------------------------------------------------------------*/
main(int argc, char **argv)
{
   int type;

   /*-------------------------------------------------------------
    */
   if (argc < 2) { printf(usage, argv[0]);  return; }

   if ((file_rx = fopen(argv[1], "rb")) == NULL)
   {
        printf("Can't open file <%s>\n", argv[1]);
        return;
   }
   setvbuf(file_rx, NULL, _IOFBF, BUFFER_SIZE);

   orb_port_packet_check(NO);

   /*-------------------------------------------------------------
    */
   while ((type = orb_port_process()) > 0  &&  type <= PKT_gps)
   {
        dump_packet(type);
   }

   fclose(file_rx);
}

/*=============================================================
=======================
 *-------------------------------------------------------------*/
int orb_port_getb() { return getc(file_rx); }

/*=============================================================
=======================
 *-------------------------------------------------------------*/
#include <times.h>
#include <const.h>

#define POS_LOW_FT 21000000.0
#define POS_HI_FT  23000000.0
#define VEL_LOW_FT 10000.0
#define VEL_HI_FT  40000.0

#define POS_LOW_M  (POS_LOW_FT * FT_TO_M)
#define POS_HI_M   (POS_HI_FT * FT_TO_M)
#define VEL_LOW_M  (VEL_LOW_FT * FT_TO_M)
#define VEL_HI_M   (VEL_HI_FT * FT_TO_M)

static State_vector_limits
```

```
    limit_ft = { { POS_LOW_FT, POS_HI_FT }, { VEL_LOW_FT, VEL_HI_FT } },
    limit_m  = { { POS_LOW_M,  POS_HI_M  }, { VEL_LOW_M,  VEL_HI_M  } };

static void dump_vector(State_vector *sv, char *msg)
{
    char work[32];
    int pos_ok, vel_ok;

    pos_ok = vector_check(&sv->pos, &limit_ft.pos);
    vel_ok = vector_check(&sv->vel, &limit_ft.vel);

    time_dbl_to_string(&sv->time, work);
    fprintf(stderr, "%-6s %s\n", msg, work);

    fprintf(stderr,"\tpos %9.0f %9.0f %9.0f", sv->pos.x, sv->pos.y, sv->pos.z);
    if (pos_ok == NO) fputs(" ERR", stderr);
    fputc('\n', stderr);

    fprintf(stderr,"\tvel %9.0f %9.0f %9.0f", sv->vel.x, sv->vel.y, sv->vel.z);
    if (vel_ok == NO) fputs(" ERR", stderr);
    fputc('\n', stderr);
}

static void dump_packet(int type)
{
    switch (type)
    {
        case PKT_orbiter:
            dump_vector(&orbiter_packet.orbiter_vector, "Orb");
            dump_vector(&orbiter_packet.target_vector, "Tgt");
            break;
        case PKT_spas:
            dump_vector(&spas_packet.gps_vector, "Spas");
            break;
        case PKT_gps:
            break;
        default: break;
    }
}
```

```
/*=============================================================
============================
 */
#include <stdio.h>
#include <stdlib.h>

#include <keys.h>
#include <packet.h>
#include <pkt_if.h>
#include <tans_pkt.h>
#include <times.h>
#include <const.h>

/*=============================================================
============================
 *-------------------------------------------------------------*/
Orb_gps_packet orbiter_gps_packet;

static int my_terminate = NO;

static FILE *my_file;
static int my_file_done = NO;

/*=============================================================
============================
 *-------------------------------------------------------------*/
static void
   do_file(char *name),
   main_key_process(void);

/*=============================================================
============================
 *-------------------------------------------------------------*/
int tans_port_getb()
{
   int rc;

   if ((rc = fgetc(my_file)) == EOF)  my_file_done = YES;

   return rc;
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
main(int argc, char **argv)
{
   static char msg[] =
        "\nThese keys active during program:\n"
        "   ESC   stops program\n"
```

```
        "\nHit any key to begin";

    /*-------------------------------------------------------------
     */
    if (argc < 2)
    {
        printf("Usage: %s tans-file1 ... tans-fileN\n", argv[0]);
        return;
    }

    puts(msg);  key_get();

    for (argv++;  argc > 1;  argv++, argc--)
    {
        do_file(*argv);

        if (my_terminate)  break;
    }
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
#define M_TO_KILOFT  (M_TO_FT/1000.0)

static void do_file(char *name)
{
    if ((my_file = fopen(name, "rb")) == NULL)
    {
        printf("Can't open file <%s>\n", name);
        return;
    }

    printf("File <%s>\n", name);
    my_file_done = NO;

    /*-------------------------------------------------------------
     */
    for (;;)
    {
        char work[24];

        /*---------------------------------------------------------
         */
        if (key_hit())  main_key_process();

        if (my_terminate  ||  my_file_done)  break;

        /*---------------------------------------------------------
         */
```

```
        if (tans_port_process() == NO) continue;

        time_dbl_to_string(&orbiter_gps_packet.gps_vector.time, work);
        printf("Time %s\n", work);

        printf(" Pos (km):  %10.3f %10.3f %10.3f\n",
          orbiter_gps_packet.gps_vector.pos.x / 1000.0,
          orbiter_gps_packet.gps_vector.pos.y / 1000.0,
          orbiter_gps_packet.gps_vector.pos.z / 1000.0);
        printf(" Pos (kft): %10.3f %10.3f %10.3f\n",
          orbiter_gps_packet.gps_vector.pos.x * M_TO_KILOFT,
          orbiter_gps_packet.gps_vector.pos.y * M_TO_KILOFT,
          orbiter_gps_packet.gps_vector.pos.z * M_TO_KILOFT);

        printf(" Vel (m/s):  %10.3f %10.3f %10.3f\n",
          orbiter_gps_packet.gps_vector.vel.x / 1000.0,
          orbiter_gps_packet.gps_vector.vel.y / 1000.0,
          orbiter_gps_packet.gps_vector.vel.z / 1000.0);
        printf(" Vel (kft/s) %10.3f %10.3f %10.3f\n",
         orbiter_gps_packet.gps_vector.vel.x * M_TO_KILOFT,
          orbiter_gps_packet.gps_vector.vel.y * M_TO_KILOFT,
          orbiter_gps_packet.gps_vector.vel.z * M_TO_KILOFT);
    }

    fclose(my_file);
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static void main_key_process()
{
    int key = key_get();

    switch (key)
    {
        case KEY_esc:  my_terminate = YES;  break;
    }
}
```

```
/*=============================================================
============================
 */
#include <stdio.h>
#include <stdlib.h>

#include <keys.h>
#include <tans_pkt.h>
#include <pkt_if.h>
#include <swap.h>

/*=============================================================
============================
 *  Week 697 is 16 May 1993.
 *-----------------------------------------------------------*/
static int my_terminate = NO;

static Tans_packet my_packet;

static FILE *my_file;
static int my_file_done = NO;

static Tans_pkt_21_41 my_packet_41;
static Tans_pkt_43    my_packet_43;
static Tans_pkt_83    my_packet_83;

/*=============================================================
============================
 *-----------------------------------------------------------*/
static void
   do_file(char *name),
   main_key_process(void),
   process_packet_41(void),
   process_packet_43(void),
   process_packet_83(void);

/*=============================================================
============================
 *-----------------------------------------------------------*/
int tans_port_getb()
{
   int rc;

   if ((rc = fgetc(my_file)) == EOF)  my_file_done = YES;

   return rc;
}

/*=============================================================
============================
```

```
 *-------------------------------------------------------------------*/
main(int argc, char **argv)
{
   static char msg[] =
        "\nThese keys active during program:\n"
        "   ESC   stops program\n"
        "\nHit any key to begin";

   /*-----------------------------------------------------------------
    */
   if (argc < 2)
   {
        printf("Usage: %s tans-file1 ... tans-fileN\n", argv[0]);
        return;
   }

   puts(msg);  key_get();

   for (argv++;  argc > 1;  argv++, argc--)
   {
        do_file(*argv);

        if (my_terminate)  break;
   }
}

/*==================================================================
=========================
 *-------------------------------------------------------------------*/
static void do_file(char *name)
{
   if ((my_file = fopen(name, "rb")) == NULL)
   {
        printf("Can't open file <%s>\n", name);
        return;
   }

   printf("File <%s>\n", name);
   my_file_done = NO;

   /*-----------------------------------------------------------------
    */
   for (;;)
   {
        /*-------------------------------------------------------------
         */
        if (key_hit())  main_key_process();

        if (my_terminate  ||  my_file_done)  break;
```

```
        /*-------------------------------------------------------
         */
        if (tans_packet_recv(&my_packet) == NO) continue;

        if (my_packet.state != TANS_state__packet_ok) continue;

        switch (my_packet.raw[1])
        {
           case 0x41: process_packet_41(); break;
           case 0x43: process_packet_43(); break;
           case 0x83: process_packet_83(); break;

           default: break;
        }
    }

    fclose(my_file);
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
static void main_key_process()
{
    int key = key_get();

    switch (key)
    {
        case KEY_esc:  my_terminate = YES;  break;
    }
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
static void process_packet_41()
{
    int size;
    Tans_pkt_21_41 *packet;

    /*-------------------------------------------------------
     */
    size = my_packet.cooked_size - sizeof(Tans_pkt_head) - sizeof(Tans_pkt_tail);

    if (size != sizeof(Tans_pkt_21_41))  return;

    packet = (Tans_pkt_21_41 *)(my_packet.cooked + sizeof(Tans_pkt_head));

    /*-------------------------------------------------------
     */
```

```
    swap_float(&packet->secs_of_week, &packet->secs_of_week);
    swap_float(&packet->gps_utc_offset, &packet->gps_utc_offset);
    swap_short(&packet->week, &packet->week);

    printf("Time: Week %d, secs %f, utc offset %f\n", packet->week,
      packet->secs_of_week, packet->gps_utc_offset);

    my_packet_41 = *packet;

    /*-------------------------------------------------------------------
     */
//NOTE: convert GPS time to UTC
}

/*==========================================================================
==========================
 *-------------------------------------------------------------------------*/
static void process_packet_43()
{
    int size;
    Tans_pkt_43 *packet;

    /*-------------------------------------------------------------------
     */
    size = my_packet.cooked_size - sizeof(Tans_pkt_head) - sizeof(Tans_pkt_tail);

    if (size != sizeof(Tans_pkt_43))
    {
        printf("Velocity pkt size %d\n", size);
        return;
    }

    packet = (Tans_pkt_43 *)(my_packet.cooked + sizeof(Tans_pkt_head));

    /*-------------------------------------------------------------------
     */
    swap_float(&packet->sol_time, &packet->sol_time);
    swap_float(&packet->vel.x, &packet->vel.x);
    swap_float(&packet->vel.y, &packet->vel.y);
    swap_float(&packet->vel.z, &packet->vel.z);
    swap_float(&packet->gigo, &packet->gigo);

    printf("Velocity (m/s): %10.3f %10.3f %10.3f (%10.3f)\n",
      packet->vel.x, packet->vel.y, packet->vel.z, packet->gigo);

    my_packet_43 = *packet;

    /*-------------------------------------------------------------------
     */
    if (my_packet_41.week != 0 && my_packet_43.sol_time == my_packet_83.sol_time)
```

```
    {
        //NOTE: got pos and vel now, use pkt_41 UTC with pkt_43/83 sol_time
    }
}

/*=========================================================================
=========================
typedef struct { Vector pos; double gigo; float sol_time; } Tans_pkt_83;
 *-------------------------------------------------------------------------*/
static void process_packet_83()
{
    int size;
    Tans_pkt_83 *packet;

    /*-----------------------------------------------------------------------
     */
    size = my_packet.cooked_size - sizeof(Tans_pkt_head) - sizeof(Tans_pkt_tail);

    if (size != sizeof(Tans_pkt_83))
    {
        printf("Position pkt size %d\n", size);
        return;
    }

    packet = (Tans_pkt_83 *)(my_packet.cooked + sizeof(Tans_pkt_head));

    /*-----------------------------------------------------------------------
     */
    swap_float(&packet->sol_time, &packet->sol_time);
    swap_double(&packet->pos.x, &packet->pos.x);
    swap_double(&packet->pos.y, &packet->pos.y);
    swap_double(&packet->pos.z, &packet->pos.z);
    swap_double(&packet->gigo, &packet->gigo);

    printf("Position (km):  %10.3f %10.3f %10.3f (%10.3f)\n",
      packet->pos.x / 1000.0, packet->pos.y / 1000.0, packet->pos.z / 1000.0,
      packet->gigo);

    my_packet_83 = *packet;

    /*-----------------------------------------------------------------------
     */
    if (my_packet_41.week != 0 && my_packet_43.sol_time == my_packet_83.sol_time)
    {
        //NOTE: got pos and vel now, use pkt_41 UTC with pkt_43/83 sol_time
    }
}
```

```
/*=================================== .====================
==========================
 */
#include <stdio.h>
#include <stdlib.h>

#include <keys.h>
#include <tans_pkt.h>
#include <pkt_if.h>

/*=================================================================
==========================
 *  Week 697 is 16 May 1993.
 *-----------------------------------------------------------------*/
static int my_terminate = NO;

static Tans_packet my_packet;

static FILE *my_file;

/*=================================================================
==========================
 *-----------------------------------------------------------------*/
static void
   main_key_process(void),
   dump_it(char *title1, char *title, ubyte *data, ushort bytes);

/*=================================================================
==========================
 *-----------------------------------------------------------------*/
int tans_port_getb() { return fgetc(my_file); }

/*=================================================================
==========================
Tans_packet;
   ubyte raw[TANS_PKT_SIZE_RAW], cooked[TANS_PKT_SIZE_COOKED];
   int raw_size, cooked_size, state;
 *-----------------------------------------------------------------*/
main(int argc, char **argv)
{
   static char msg[] =
        "\nThese keys active during program:\n"
        "  ESC   stops program\n"
        "\nHit any key to begin";

   /*-----------------------------------------------------------
    */
   if (argc < 2)
   {
        printf("Usage: %s tans-data-file\n", argv[0]);
```

```
        return;
    }

    if ((my_file = fopen(argv[1], "rb")) == NULL)
    {
        printf("Can't open file <%s>\n", argv[1]);
        return;
    }

    puts(msg);  key_get();

    /*-------------------------------------------------------------
     */
    for (;;)
    {
        /*-------------------------------------------------------------
         */
        if (key_hit())  main_key_process();

        if (my_terminate)  break;

        /*-------------------------------------------------------------
         */
        if (tans_packet_recv(&my_packet))
        {
            if (my_packet.state == TANS_state__packet_ok)
            {
                static ushort pkt_cnt;
                pkt_cnt++;
                printf("%5u recv %02x = %3d bytes:", pkt_cnt, my_packet.raw[1],
                  my_packet.cooked_size - 4);
                dump_it(" ", "\t\t\t   ", my_packet.cooked+2, my_packet.cooked_size-4);
            }
            else
            {
                dump_it("bad pkt ", "\t", my_packet.raw, my_packet.raw_size);
            }
        }
    }

    fclose(my_file);
}

/*=============================================================
 *-----------------------------------------------------------*/
static void main_key_process()
{
    int key = key_get();
```

```
    switch (key)
    {
        case KEY_esc:  my_terminate = YES;  break;
    }
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
static void dump_it(char *title1, char *title, ubyte *data, ushort bytes)
{
    ushort cnt;

    for (cnt = 0;  cnt < bytes;  cnt++, data++)
    {
        if ((cnt % 16) == 0)
        {
            if (cnt == 0)  printf("%s ", title1);
            else     printf("\n%s ", title);
        }
        printf(" %02x", *data);
    }
    fputc('\n', stdout);
}
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>

#include <keys.h>
#include <edit_rt.h>
#include <times.h>

/*=============================================================
============================
 *-------------------------------------------------------------*/
extern int
    main_key_handler(int),
    main_display_init(void),
    test_key_handler(int),
    test_display_init(void),
    test_report(void);

/*=============================================================
============================
 *-------------------------------------------------------------*/
static int (*my_key_handler)(int) = main_key_handler;

static int terminate = NO;

/*=============================================================
============================
 *-------------------------------------------------------------*/
main()
{
    main_display_init();
    for (;;)
    {
        if (kbhit())
        {
            key_process();
            if (terminate)  break;
        }
    }

    gotoxy(1, 15);
    test_report();
    test_time();
    test_graphics();
}

/*=============================================================
```

```
==========================
 *-------------------------------------------------------------*/
static int key_process(void)
{
   int rc, key = getch();
   if (key == 0)  key = getch() + 0x100;

   rc = (*my_key_handler)(key);

   if (rc == 2)
   {
        my_key_handler = main_key_handler;
        main_display_init();
   }
}

/*=============================================================
==========================
 * Returns:
 * 0    Didn't use the key
 * 1    Used the key
 * 2    Used the key, My subsystem is done
 *-------------------------------------------------------------*/
static int main_key_handler(int key)
{
   switch (key)
   {
        case KEY_esc:
           terminate = YES;
           return YES;

        case KEY_f1:
           if (test_display_init() == YES)
                my_key_handler = test_key_handler;
           return YES;
   }
   return NO;
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
typedef struct { int row, col; char *text; } Label;

main_display_init()
{
   static Label labels[] =
   {
        { 6,16, "Simulator ... yakity yak ... grumble grumble"},
        {10, 6, "F1 - test edit 1"},
```

```
        {12, 6, "ESC  Exit"},
        { 0, 0, NULL }
    };

    Label *label;

    /*-----------------------------------------------------------
     */
    clrscr();

    for (label = labels;  label->text != NULL;  label++)
    {
        gotoxy(label->col, label->row);
        cputs(label->text);
    }
}

/*==========================================================
==========================
 *-----------------------------------------------------------*/
int test_time()
{
    double time = ((((20 * 24) + 2) * 60.0) + 15) * 60.0 + 5.125;
    char work[25];
    int day,hour,min,sec;

    Time_dhms dhms;
    double time2;

    day = (int)(time/86400.0);
    hour= (int)((time-day*86400.0)/3600.0);
    min = (int)((time-day*86400.0-hour*3600.0)/60);
    sec = (int)(time-day*86400.0-hour*3600.0-min*60);

    printf("brute force:  %03d/%02d:%02d:%02d\n", day,hour,min,sec);

    printf("time routine: %s\n", time_dbl_to_string(&time, work));

    time_dbl_to_dhms(&time, &dhms);
    printf("time-to-hms:  %03d/%02d:%02d:%02d.%03d\n",
      dhms.days, dhms.hours, dhms.mins, dhms.secs, dhms.msecs);

    time_dhms_to_dbl(&dhms, &time2);
    printf("time1 %lg\n", time);
    printf("time2 %lg\n", time2);
}
```

```
/*=============================================================
==========================
 *-------------------------------------------------------------*/
#include <stdio.h>
#include <conio.h>

#include <edit.h>
#include <com_port.h>

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static int init(void);
static void test_set(int), test_rx(int), test_tx_keys(int), test_tx_file(int);

static void (*tests[])(int) = { test_set, test_rx, test_tx_keys, test_tx_file };

#define TESTS (sizeof(tests)/sizeof(void *))

static char label[] =
   "\n  0 exit"
   "\n  1 Baud-set test"
   "\n  2 Rx test"
   "\n  3 Tx test - keys"
   "\n  4 Tx test - file"
   "\nwhich test:";

/*=============================================================
==========================
 *-------------------------------------------------------------*/
main(int argc, char **argv)
{
   int port_id, test;

   /*-----------------------------------------------------------
    */
   if ((port_id = init()) == 0)  return;

   for (;;)
   {
        printf(label); scanf("%d", &test);
        if (!test  ||  test > TESTS)  break;
        (*tests[test-1])(port_id);
   }

   com_port_stop(port_id);
   return 0;
}

/*=============================================================
```

```
==========================
 *-------------------------------------------------------------*/
static int init()
{
   int port_id;
   ushort rx_size, tx_size;

   /*-------------------------------------------------------------
    */
   port_id = edit_integer("Enter port id ", 1);
   rx_size = edit_integer("Enter rx size ", 512);
   tx_size = edit_integer("Enter tx size ", 512);

   if (com_port_start(port_id, rx_size, tx_size) == NO)
   {
        printf("Can't start port %d\n", port_id);
        port_id = 0;
        return 0;
   }

   return port_id;
}

/*=========================================================
==========================
 *-------------------------------------------------------------*/
static void test_set(int port_id)
{
   long baud = edit_integer_long("Enter baud    ", 9600);
   int parity = edit_integer("Enter parity (0 none, 1 even, 2 odd) ", 0);
   int stops  = edit_integer("Enter stop bits, 1 or 2 ", 1);

   parity = (parity == 0
     ? COM_PORT_par_none
     : (parity == 1
       ? COM_PORT_par_even
         : COM_PORT_par_odd));

   stops = (stops == 1 ? COM_PORT_stop_1 : COM_PORT_stop_2);

   com_port_set(port_id, baud, parity, stops);
}

/*=========================================================
==========================
 *-------------------------------------------------------------*/
static void test_rx(int port_id)
{
   char *filename = edit_text("Enter dump file (blank for none) ", "rx.dat");
   FILE *dump;
```

```
   /*-------------------------------------------------------------
    */
   dump = (filename[0] ? fopen(filename, "wb") : NULL);

   puts("Hit any key to stop");

   for (;;)
   {
        int value;

        if (kbhit())  { if (getch() == 0) getch();  break; }
        if ((value = com_port_get(port_id)) == -1)  continue;

        printf(" %02x", value);
        fputc(value, dump);
   }

   if (dump != NULL)  { fflush(dump);  fclose(dump); }
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static void test_tx_keys(int port_id)
{
   puts("Hit any function-key to stop");

   for (;;)
   {
        int key;

        if (! kbhit())  continue;
        if ((key = getch()) == 0)   key = getch() + 0x100;

        if (key > 0x100)  break;
        com_port_put(port_id, key);
   }
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static void test_tx_file(int port_id)
{
   char *filename = edit_text("Enter file name ", "rx.dat");
   FILE *file;
   int got_value, value;

   /*-------------------------------------------------------------
    */
```

```
    if (filename[0] == 0)  return;
    if ((file = fopen(filename, "rb")) == NULL)
    {
         printf("  Can't open <%s>\n", filename);
         return;
    }

    /*-------------- ----------------------------------------
     */
    puts("Hit any key to stop");

    got_value = NO;
    for (;;)
    {
         if (kbhit())  { if (getch() == 0) getch();  break; }
         if (got_value == NO)
         {
            if ((value = fgetc(file)) == EOF)  break;
            got_value = YES;
         }
         if (com_port_put(port_id, value))  got_value = NO;
    }
    fclose(file);
}
```

```
/*=========  ==========================================
==========  ==================
 */
#include <stdio.h>
#include <conio.h>
#include <fcntl.h>
#include <io.h>
#include <alloc.h>

#include <vector.h>

/*=============================================================
=========================
 */
extern int test_write(int), test_read(int), test_ioctl(int);
int (*tests[])(int) = { test_write, test_read, test_ioctl };
#define TESTS (sizeof(tests)/sizeof(void *))

char label[] =
   "\n  0 exit\n  1 write test\n  2 read test\n  3 ioctl test\n"
   "which test:";

#define O_RDWR_BIN (O_RDWR | O_BINARY)

#define FILE_IS_DEVICE        0x0080
#define FILE_IS_RAW   0x0020
#define FILE_SET_BITS 0x00ff

/*=============================================================
=========================
 *-------------------------------------------------------------*/
main(int argc, char **argv)
{
   int handle, status, test;

   /*-------------------------------------------------------------
    */
   if (argc < 2)
   {
        printf("Usage: %s device_name\n", argv[0]);
        return;
   }

   /*-------------------------------------------------------------
    */
   printf("Opening device <%s>\n", argv[1]);

   if ((handle = open(argv[1], O_RDWR_BIN)) < 0)
   {
        printf("Can't open device <%s>\n", argv[1]);
```

```
        return;
    }

    /*-----------------------------------------------------------
     *  If file is a device, place that device in "raw" mode.
     */
    status = ioctl(handle, 0, 0,0);

    if (status & FILE_IS_DEVICE)
    {
        status = (status & FILE_SET_BITS) | FILE_IS_RAW;
        ioctl(handle, 1, status,0);
    }

    /*-----------------------------------------------------------
     */
    for (;;)
    {
        printf(label); scanf("%d", &test);
        if (!test || test > TESTS) break;
        (*tests[test-1])(handle);
    }

    /*-----------------------------------------------------------
     *  If file is a device, place that device back in "cooked" mode.
     */
    if (status & FILE_IS_DEVICE)
    {
        status &= ~FILE_IS_RAW;
        ioctl(handle, 1, status,0);
    }

    close(handle);
}

/*============================================================
============================
 *-----------------------------------------------------------*/
test_write(int handle)
{
    struct { char name[80]; int handle; ulong size; } file;
    ubyte *buffer, *bptr;
    ushort bsize, cnt;
    int bytes, rc;

    /*-----------------------------------------------------------
     */
    printf("enter filename: "); scanf("%s", file.name);
    if ((file.handle = open(file.name, O_RDWR_BIN)) < 0) return;
```

```
    file.size = lseek(file.handle, 0L, SEEK_END);
    lseek(file.handle, 0L, SEEK_SET);
    printf("file <%s> is %ld bytes\n", file.name, file.size);

    buffer = malloc(bsize = 0x4000);
    if (buffer == NULL) { close(file.handle);  return; }

    /*-------------------------------------------------------------
     */
    for (cnt = 0, bytes = 0;  ;  )
    {
        if (bytes <= 0)
        {
            bytes = read(file.handle, buffer, bsize);
            if (bytes <= 0)  break;
            bptr = buffer;
        }

        rc = _write(handle, bptr, bytes);
        if (rc > 0)
        {
            cnt++;
            bptr += rc;  bytes -= rc;
            printf("\r%3d  wrote %d bytes\n", cnt, rc);
        }
        if (kbhit())  break;
    }

    close(file.handle);
    free(buffer);
}

/*=============================================================
==========================  ===
 *-------------------------------------------------------------*/
test_read(int handle)
{
    struct { int handle;  ushort size; } file;
    void *local;
    int rc;

    /*-------------------------------------------------------------
     */
    printf("enter size: ");  scanf("%i", &file.size);
    if (file.size == 0)  return;

    if ((local = malloc(file.size)) == NULL)  return;
    if ((file.handle = open("test.dat", O_RDWR_BIN|O_CREAT, 0666)) < 0)  return;

    /*-------------------------------------------------------------
```

```
    */
    for (;;)
    {
        rc = _read(handle, local, file.size);
        if (rc > 0)
        {
            printf("read %d bytes\n", rc);
            write(file.handle, local, rc);
        }
        if (kbhit())  break;
    }

    close(file.handle);
    free(local);
}


/*=============================================================
==========================
 *-------------------------------------------------------------*/
test_ioctl(int handle)
{
    struct { int type;  ubyte data[10]; } msg;
    int rc;

    for (;;)
    {
        msg.type = 4;
        strcpy(msg.data, "123456789");
        rc = ioctl(handle, 2, &msg, sizeof(msg));
        printf("ioctl %d bytes\n", rc);
        break;
    }
}
```

```
/*=============================================================
============================
 *-----------------------------------------------------------*/
#include <stdio.h>
#include <mem.h>

#include <vector.h>
#include <const.h>
#include <orbmech1.h>

/*=============================================================
============================
 */
double m50efe[3][3], iload_t_epoch;

extern void
   vectcnv(double time, Vector *ecef_r, Vector *ecef_v, Vector *m50_r, Vector *m50_v, int flg);

#define ECEF_TO_M50 0
#define M50_TO_ECEF 1

#define M50_TO_WGS84 0
#define WGS84_TO_M50 1

/*=============================================================
============================
 */
static State_vector_transform m50_ecef;

/*=============================================================
============================
 */
static int
   load_matrix(void);

static void
   show_vector(State_vector *v, char *label),
   show_vector_diff(State_vector *v1, char *label, State_vector *v2),
   show_transform(State_vector_transform *svt, char *label),
   read_vector(State_vector *v, double *dtime, int *time_rel);

static void
   test_set_ecef_vector(void),
   test_old_ecef_to_m50(void),
   test_new_ecef_to_m50(void),
   test_nps_ecef_to_m50(void),
   test_set_m50_vector(void),
   test_old_m50_to_ecef(void),
   test_new_m50_to_ecef(void),
   test_nps_m50_to_ecef(void),
```

```
   test_show_transforms(void),
   test_make_transform(void);

/*=============================================================
============================
 */
void (*tests[])(void) =
{
   test_set_ecef_vector, test_old_ecef_to_m50, test_new_ecef_to_m50, test_nps_ecef_to_m50,
   test_set_m50_vector, test_old_m50_to_ecef, test_new_m50_to_ecef, test_nps_m50_to_ecef,
   test_show_transforms, test_make_transform
};

#define TESTS (sizeof(tests)/sizeof(void *))

char label[] =
   "\n  0 exit"
   "\n  1 set ecef vector"  "\t  5 set m50 vector"
   "\n  2 old  ecef-to-m50" "\t  6 old  m50-to-ecef"
   "\n  3 new  ecef-to-m50" "\t  7 new  m50-to-ecef"
   "\n  4 NPS  ecef-to-m50" "\t  8 NPS  m50-to-ecef"
                     "\n\t\t\t  9 Print transforms"
                     "\n\t\t\t 10 Update transform"
   "\nwhich test:";

/*=============================================================
============================
 */
typedef struct
{
   State_vector vector;
   double dtime;
   int time_rel;
} My_vector_info;

static My_vector_info ecef =
{
   {
        { 913061.83 * M_TO_FT, -5527811.25 * M_TO_FT, 3037681.65 * M_TO_FT },
        { 0.0, 0.0, 0.0 },
        0.0
   },
   0.0, YES
};

static My_vector_info m50 =
{
   {
        { 11694900.0, -14218100.0, 9917380.0 },
        { 1044.00562, 843.50684, 4.32033 },
```

```
        0.0
    },
    0.0, YES
};

/*=============================================================
============================
 *-------------------------------------------------------------*/
main(int argc, char **argv)
{
    int test, year;

    /*-------------------------------------------------------------
     */
    if (load_matrix() == NO)  return;

    printf("Enter year:");  scanf("%d", &year);
    bgprop_year(year);
    if (bgprop_init() == NO)  return NO;

    /*-------------------------------------------------------------
     */
    for (;;)
    {
        printf(label); scanf("%d", &test);
        if (!test  ||  test > TESTS)  break;
        (*tests[test-1])();
    }
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
static int load_matrix()
{
    char name[80];
    FILE *iload;

    /*-------------------------------------------------------------
     */
    printf("Enter iload filename:");  name[0] = 0;  scanf("%s", name);
    if (name[0] == 0)  return NO;

    if ((iload = fopen(name, "r")) == NULL)  return NO;

    /*-------------------------------------------------------------
     */
    fscanf(iload, "%le", &iload_t_epoch);
    fscanf(iload, "%le %le %le %le %le %le %le %le %le",
```

```
    &m50efe[0][0], &m50efe[0][1], &m50efe[0][2],
    &m50efe[1][0], &m50efe[1][1], &m50efe[1][2],
    &m50efe[2][0], &m50efe[2][1], &m50efe[2][2]);
  fclose(iload);

  m50_ecef.time = iload_t_epoch;
  memcpy(m50_ecef.matrix, m50efe, sizeof(m50_ecef.matrix));

  return YES;
}

/*==========================================================
==========================
 *-----------------------------------------------------------*/
static void show_vector(State_vector *v, char *label)
{
  static char fmt1[] =
      "%s\n"
      "  %13.3lf x   %11.5lf xdot\n"
      "  %13.3lf y   %11.5lf ydot\n"
      "  %13.3lf z   %11.5lf zdot\n"
      "  %13.3lf mag %11.5lf mag\n"
      "  time  %lf\n";

  printf(fmt1, label,
    v->pos.x, v->vel.x, v->pos.y, v->vel.y, v->pos.z, v->vel.z,
    vector_magnitude(&v->pos), vector_magnitude(&v->vel), v->time);
};

static void show_vector_diff(State_vector *v1, char *label, State_vector *v2)
{
  static char fmt1[] =
      "%s\n"
      "  %13.3lf x   %11.5lf xdot | %13.3lf x   %11.5lf xdot\n"
      "  %13.3lf y   %11.5lf ydot | %13.3lf y   %11.5lf ydot\n"
      "  %13.3lf z   %11.5lf zdot | %13.3lf z   %11.5lf zdot\n"
      "  %13.3lf mag %11.5lf mag  | %13.3lf mag %11.5lf mag\n"
      "  time  %lf\n";

  State_vector delta;

  vector_diff(&v1->pos, &v2->pos, &delta.pos);
  vector_diff(&v1->vel, &v2->vel, &delta.vel);
  printf(fmt1, label,
    v1->pos.x, v1->vel.x, delta.pos.x, delta.vel.x,
    v1->pos.y, v1->vel.y, delta.pos.y, delta.vel.y,
    v1->pos.z, v1->vel.z, delta.pos.z, delta.vel.z,
    vector_magnitude(&v1->pos), vector_magnitude(&v1->vel),
    vector_magnitude(&delta.pos), vector_magnitude(&delta.vel),
    v1->time);
```

```
};

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static void show_transform(State_vector_transform *svt, char *label)
{
   static char fmt[] =
        "%s  time: %20lf\n"
        "\t%20le %20le %20le\n"
        "\t%20le %20le %20le\n"
        "\t%20le %20le %20le\n";

   printf(fmt, label, svt->time,
     svt->matrix[0][0], svt->matrix[0][1], svt->matrix[0][2],
     svt->matrix[1][0], svt->matrix[1][1], svt->matrix[1][2],
     svt->matrix[2][0], svt->matrix[2][1], svt->matrix[2][2]);
}

/*=============================================================
==========================
 *-------------------------------------------------------------*/
static void read_vector(State_vector *v, double *dtime, int *time_rel)
{
   char answer[80];
   int is_meters;

   /*----------------------------------------------------------
    */
   show_vector(v, "Existing vector:");
   printf("  dtime %lf\n", *dtime);

   printf("Change vector ? "); answer[0] = 0; scanf("%s", answer);
   if (answer[0] == 'y')
   {
        printf("Enter meters or ft ? "); answer[0] = 0; scanf("%s", answer);
        is_meters = (answer[0] == 'm' ? YES : NO);

        printf("Enter x y z xdot ydot zdot: ");
        scanf("%lf %lf %lf %lf %lf %lf", &v->pos.x, &v->pos.y, &v->pos.z,
          &v->vel.x, &v->vel.y, &v->vel.z);

        if (is_meters)
        {
           v->pos.x *= M_TO_FT; v->pos.y *= M_TO_FT; v->pos.z *= M_TO_FT;
           v->vel.x *= M_TO_FT; v->vel.y *= M_TO_FT; v->vel.z *= M_TO_FT;
        }
   }

   /*----------------------------------------------------------
```

```
    */
    printf("Change time ? "); answer[0] = 0; scanf("%s", answer);
    if (answer[0] == 'y')
    {
        printf("Enter abs/rel time: ");
        answer[0] = 0; scanf("%s %lf", answer, dtime);
        if (answer[0] == 'r')           *time_rel = YES;
        else if (answer[0] == 'a')      *time_rel = NO;
    }

    show_vector(v, "Vector in ft");
    printf("  dtime %lf\n", *dtime);
}

/*=============================================================
=========================
 *-------------------------------------------------------------*/
static void test_set_ecef_vector(void)
{
    read_vector(&ecef.vector, &ecef.dtime, &ecef.time_rel);
}

static void test_old_ecef_to_m50()
{
    State_vector m50_new;

    if (ecef.time_rel)      ecef.vector.time = iload_t_epoch + ecef.dtime;
    else                    ecef.vector.time = ecef.dtime;

    vectcnv(ecef.vector.time,
      &ecef.vector.pos, &ecef.vector.vel,
      &m50_new.pos, &m50_new.vel, ECEF_TO_M50);
    m50_new.time = ecef.vector.time;

    show_vector(&ecef.vector, "Vector ECEF");
    show_vector_diff(&m50_new, "Vector M50\t\t\t\t\tM50 - Ref M50", &m50.vector);
}

static void test_new_ecef_to_m50()
{
    State_vector m50_new;

    if (ecef.time_rel)      ecef.vector.time = iload_t_epoch + ecef.dtime;
    else                    ecef.vector.time = ecef.dtime;

    state_vector__ecef_to_m50(&ecef.vector, &m50_new, &m50_ecef, YES);
    m50_new.time = ecef.vector.time;

    show_vector(&ecef.vector, "Vector ECEF");
    show_vector_diff(&m50_new, "Vector M50\t\t\t\t\tM50 - Ref M50", &m50.vector);
```

```
}

static void test_nps_ecef_to_m50()
{
   State_vector m50_new;

   if (ecef.time_rel)      ecef.vector.time = iload_t_epoch + ecef.dtime;
   else            ecef.vector.time = ecef.dtime;

   m50_wgs84_m50(&ecef.vector, &m50_new, WGS84_TO_M50, NULL, YES);
   m50_new.time = ecef.vector.time;

   show_vector(&ecef.vector, "Vector ECEF");
   show_vector_diff(&m50_new, "Vector M50\t\t\t\t\tM50 - Ref M50", &m50.vector);
}

/*==========================================================
==========================
 *----------------------------------------------------------*/
static void test_set_m50_vector(void)
{
   read_vector(&m50.vector, &m50.dtime, &m50.time_rel);
}

static void test_old_m50_to_ecef()
{
   State_vector ecef_new;

   if (m50.time_rel)      m50.vector.time = iload_t_epoch + m50.dtime;
   else            m50.vector.time = m50.dtime;

   vectcnv(m50.vector.time,
     &ecef_new.pos, &ecef_new.vel,
     &m50.vector.pos, &m50.vector.vel, M50_TO_ECEF);
   ecef_new.time = m50.vector.time;

   show_vector(&m50.vector, "Vector M50");
   show_vector_diff(&ecef_new, "Vector ECEF\t\t\t\t\tECEF - Ref ECEF", &ecef.vector);
}

static void test_new_m50_to_ecef()
{
   State_vector ecef_new;

   if (m50.time_rel)      m50.vector.time = iload_t_epoch + m50.dtime;
   else            m50.vector.time = m50.dtime;

   state_vector__m50_to_ecef(&m50.vector, &ecef_new, &m50_ecef, YES);
   ecef_new.time = m50.vector.time;
```

```
   show_vector(&m50.vector, "Vector M50");
   show_vector_diff(&ecef_new, "Vector ECEF\t\t\t\t\tECEF - Ref ECEF", &ecef.vector);
}

static void test_nps_m50_to_ecef()
{
   State_vector ecef_new;

   if (m50.time_rel)      m50.vector.time = iload_t_epoch + m50.dtime;
   else           m50.vector.time = m50.dtime;

   m50_wgs84_m50(&m50.vector, &ecef_new, M50_TO_WGS84, NULL, YES);
   ecef_new.time = m50.vector.time;

   show_vector(&m50.vector, "Vector M50");
   show_vector_diff(&ecef_new, "Vector ECEF\t\t\t\t\tECEF - Ref ECEF", &ecef.vector);
}

/*=============================================================
============================
 *-------------------------------------------------------------------------*/
static void test_show_transforms()
{
   State_vector ecef_gigo;
   State_vector_transform ecef_m50_new, m50_ecef_new;

   if (m50.time_rel)      m50.vector.time = iload_t_epoch + m50.dtime;
   else           m50.vector.time = m50.dtime;

   m50_wgs84_m50(&m50.vector, &ecef_gigo, M50_TO_WGS84, &m50_ecef_new, YES);
   show_transform(&m50_ecef_new, "Computed M50 to ECEF");

   state_vector__rotate_m50_ecef(&m50_ecef,  m50.vector.time, &ecef_m50_new);
   matrix_transpose(ecef_m50_new.matrix,  m50_ecef_new.matrix);
   m50_ecef_new.time = ecef_m50_new.time;
   show_transform(&m50_ecef_new, "I-Load   M50 to ECEF");
}

static void test_make_transform()
{
   State_vector m50_gigo, ecef_gigo;
   State_vector_transform m50_ecef_new;

   m50_gigo = m50.vector;
   printf("Enter abs time: ");
   scanf("%lf", &m50_gigo.time);

   m50_wgs84_m50(&m50_gigo, &ecef_gigo, M50_TO_WGS84, &m50_ecef_new, YES);
   show_transform(&m50_ecef_new, "Computed M50 to ECEF");
```

```
    m50_ecef = m50_ecef_new;
    iload_t_epoch = m50_ecef.time;
    memcpy(m50efe, m50_ecef.matrix, sizeof(m50_ecef.matrix));
    show_transform(&m50_ecef, "I-Load   M50 to ECEF");
}
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>

#include <keys.h>
#include <edit_rt.h>

/*=============================================================
============================
 *-------------------------------------------------------------*/
static int int1 = 10, int2 = 33;
static double dbl1 = 3.14159265359, dbl2 = 1.414e20;

/*=============================================================
============================
 *--123456789 123456789 123456789 123456789 123456789 123456789
 * 1
 * 2  Hello There
 * 3
 * 4  Int1: X
 * 5  Int1: X
 * 6
 * 7
 * 8  dbl1: X                  dbl2: X
 * 9
 *-------------------------------------------------------------*/
static Edit_label my_labels[] =
{
   { 2, 4, "Hello There"},
   { 4, 4, "Int1:"},
   { 5, 4, "Int2:"},
   { 8, 4, "dbl1:"},
   { 8,34, "dbl2:"}
};
#define LABEL_COUNT (sizeof(my_labels)/sizeof(Edit_label))

static Edit_field my_fields[] =
{
   { 4,10, edit_field_get_int, edit_field_put_int, edit_field_key_int, &int1 },
   { 5,10, edit_field_get_int, edit_field_put_int, edit_field_key_int, &int2 },
   { 8,10, edit_field_get_dbl, edit_field_put_dbl, edit_field_key_dbl, &dbl1 },
   { 8,40, edit_field_get_dbl, edit_field_put_dbl, edit_field_key_dbl, &dbl2 }
};
#define FIELD_COUNT (sizeof(my_fields)/sizeof(Edit_field))

static Edit_screen my_screen =
{
```

```
    LABEL_COUNT, my_labels,
    FIELD_COUNT, my_fields
};

/*=============================================================
============================
 * Returns:
 * 0     Didn't use the key
 * 1     Used the key
 * 2     Used the key, My subsystem is done
 *-------------------------------------------------------------*/
int test_key_handler(int key)
{
    return edit_screen_key_handler(&my_screen,  key);
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
test_display_init()
{
    edit_screen_init(&my_screen);
    return YES;
}

/*=============================================================
============================
 *-------------------------------------------------------------*/
int test_report()
{
    printf("\nint1 : %d\n", int1);
    printf("int2 : %d\n", int2);
    printf("dbl1 : %lg\n", dbl1);
    printf("dbl2 : %lg\n\n", dbl2);
}
```

```
/*=========================================================================
==========================
 *-------------------------------------------------------------------------*/
#include <conio.h>
#include <graphics.h>

#define FONT_SIZE 4
#define HORIZ_DIR 0

/*=========================================================================
==========================
 *-------------------------------------------------------------------------*/
int test_graphics()
{
   int gdriver = DETECT, gmode, font;
   int base, cnt;
   char work[40], *wptr;

   printf("\n\nEnter font # "); scanf("%d", &font);
   initgraph(&gdriver, &gmode, "");
   settextstyle(font, HORIZ_DIR, FONT_SIZE);

   for (base = 0; base < 256; base += 16)
   {
        for (wptr = work, cnt = 0; cnt < 16; cnt++, wptr += 2)
        {
           if ((wptr[0] = base + cnt) == 0) wptr[0] = ' ';
           wptr[1] = ' ';
        }
        clearviewport();
        outtextxy(2,2, work);
        getch();
   }
   closegraph();
}
```

```
/*=============================================================
============================
 *-------------------------------------------------------------*/
#include <math.h>

extern double m50efe[3][3], iload_t_epoch;

extern void
   dmvmul(double m[3][3], double v1[3], double v2[3]),
   eftom50(double time, double mefm50[3][3]),
   dxpose(double m[3][3], double mt[3][3]);

/*=============================================================
============================
 *-------------------------------------------------------------*/
#ifdef GIGO
static show_vector(char *title, double v[3])
{
   printf("%s\t%20lf %20lf %20lf\n", title, v[0], v[1], v[2]);
}

static show_matrix(char *title, double m[3][3])
{
   static char fmt[] =
        "%s\t%20lf %20lf %20lf\n"
        "\t\t%20lf %20lf %20lf\n"
        "\t\t%20lf %20lf %20lf\n";

   printf(fmt, title,
     m[0][0], m[0][1], m[0][2],
     m[1][0], m[1][1], m[1][2],
     m[2][0], m[2][1], m[2][2]);
}
#endif GIGO

/*=============================================================
============================
 * flg = 0   ECEF -> M50
 * flg = 1   M50 -> ECEF
 *-------------------------------------------------------------*/
void vectcnv(double time, double r_ecef[3], double v_ecef[3], double r_m50[3], double v_m50[3], int flg)
{
   double Earth_Rate = 7.29211514646e-5;
   double omega[3], omegaxr[3], mefm50[3][3];
   double vtemp[3], mm50ef[3][3];

   eftom50(time,mefm50);

   omega[0] = Earth_Rate * m50efe[2][0];
   omega[1] = Earth_Rate * m50efe[2][1];
```

```
    omega[2] = Earth_Rate * m50efe[2][2];

    if (flg == 0)
    {
        dmvmul(mefm50,r_ecef,r_m50);
        omegaxr[0] = omega[1]*r_m50[2] - omega[2]*r_m50[1];
        omegaxr[1] = omega[2]*r_m50[0] - omega[0]*r_m50[2];
        omegaxr[2] = omega[0]*r_m50[1] - omega[1]*r_m50[0];
        dmvmul(mefm50,v_ecef,v_m50);
        v_m50[0] += omegaxr[0];  v_m50[1] += omegaxr[1];  v_m50[2] += omegaxr[2];
    }
    else
    {
        omegaxr[0] = omega[1]*r_m50[2] - omega[2]*r_m50[1];
        omegaxr[1] = omega[2]*r_m50[0] - omega[0]*r_m50[2];
        omegaxr[2] = omega[0]*r_m50[1] - omega[1]*r_m50[0];
        dxpose(mefm50,mm50ef);
        dmvmul(mm50ef,r_m50,r_ecef);
        vtemp[0] = v_m50[0] - omegaxr[0];
        vtemp[1] = v_m50[1] - omegaxr[1];
        vtemp[2] = v_m50[2] - omegaxr[2];
        dmvmul(mm50ef,vtemp,v_ecef);
    }
}

/*==========================================================
==========================
 *-------------------------------------------------------------*/
void eftom50(double time, double mefm50[3][3])
{
    double delt, clam, slam, lam;
    double earth_rate = 7.29211514646e-5;

    delt = time - iload_t_epoch;
    lam = earth_rate*delt;
    clam = cos(lam);
    slam = sin(lam);

    mefm50[0][0] = m50efe[0][0]*clam + m50efe[1][0]*slam;
    mefm50[0][1] = -m50efe[0][0]*slam + m50efe[1][0]*clam;
    mefm50[0][2] = m50efe[2][0];
    mefm50[1][0] = m50efe[0][1]*clam + m50efe[1][1]*slam;
    mefm50[1][1] = -m50efe[0][1]*slam + m50efe[1][1]*clam;
    mefm50[1][2] = m50efe[2][1];
    mefm50[2][0] = m50efe[0][2]*clam + m50efe[1][2]*slam;
    mefm50[2][1] = -m50efe[0][2]*slam + m50efe[1][2]*clam;
    mefm50[2][2] = m50efe[2][2];
}

void dmvmul(double m[3][3], double v1[3], double v2[3])
```

```
{
   int i,j;

   for(i=0; i<3; i++)
   {
        v2[i] = 0.0;
        for(j=0; j<3; j++)
           v2[i] = m[i][j]*v1[j] + v2[i];
   }
}

void dxpose(double m[3][3], double mt[3][3])
{
   int i,j;

   for(i=0; i< 3; i++)
        for(j=0; j<3; j++)
           mt[i][j] = m[j][i];
}
```

```
/*=============================================================
=========================
 *-------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

/*=============================================================
=========================
 *-------------------------------------------------------------*/
int main(int argc, char **argv)
{
   static char msg[] =
        "Usage: %s mode\n"
        "  3  Normal 80x25 color\n"
        " 64  EGA 43, or VGA 50 line color\n";

   /*-------------------------------------------------------------
    */
   if (argc == 2)
   {
        int mode = atoi(argv[1]);

        if (mode == 3  ||  mode == 64)
        {
           textmode(mode);
           return;
        }
   }

   printf(msg, argv[0]);
}
```

```
#=============================================================
============================
BCC_PATH = c:\util\bcc

LIBPATH = $(BCC_PATH)\lib
INCPATH = inc;$(BCC_PATH)\inc

.path.obj = objs

#=============================================================
============================
C0 = c0l.obj
CLIB = cl.lib
MATHLIB = fp87.lib mathl.lib
GRAPHLIB = graphics.lib

#=============================================================
============================
RM = rm
TJS_DBUG = p_map

#RM = erase
#TJS_DBUG = echo

# For Rendezvous:
#       make clean
#       make -DRNDV  pgm.exe
#
!if $d(RNDV)
CEXTRA = $(CEXTRA) -DNPS_RNDV
!endif

# For GPS:
#       make clean
#       make -DGPS  pgm.exe
#
!if $d(GPS)
CEXTRA = $(CEXTRA) -DNPS_GPS
!endif

!if $d(SIM)
CEXTRA = $(CEXTRA) -DNPS_SIM
!endif

#=============================================================
============================
CC = bcc
AS = tasm
LD = tlink
AR = tlib
```

```
!if $d(DEBUG)
CDEBUG = -N -v -y -DDEBUG
!else
CDEBUG = -N -v -y
!endif

ADEBUG = -zi
AFLAGS = -ml -iinc $(ADEBUG) -j.286P

# For Debuging:
#       make -DDEBUG  pgm.exe
#
!if $d(DEBUG)
LFLAGS_DEBUG = /m/l/v
!endif
LFLAGS = /c/d$(LFLAGS_DEBUG)

.AUTODEPEND

#=============================================================
============================
# Implicit rules
#
{nps}.c.obj:
  $(CC) -c -Inps {$< }
{tans}.c.obj:
  $(CC) -c -Itans {$< }
{orbmech1}.c.obj:
  $(CC) -c -Iorbmech1 {$< }
{util}.c.obj:
  $(CC) -c {$< }
{util}.asm.obj:
    $(AS) $(AFLAGS) $<,$@
{com}.c.obj:
  $(CC) -c {$< }
{com}.asm.obj:
    $(AS) $(AFLAGS) $<,$@
{nps_sim}.c.obj:
  $(CC) -c -Inps_sim {$< }
{tans_sim}.c.obj:
  $(CC) -c -Itans_sim {$< }
{test}.c.obj:
  $(CC) -c {$< }

#=============================================================
============================
#
nothing:
  @echo "Type 'make TARGET'  where target is one of"
  @echo " tans.exe..... TANS, with NPS ***** STS-51 Flight Version *****"
```

```
  @echo " tans_spd.exe. TANS, no NPS, with SPDRIVE"
  @echo " tans51sm.exe. TANS, with NPS, Sim from file"
  @echo " tans51fn.exe. TANS, no NPS, testing"
  @echo " tsim.exe..... TANS device simulator"
  @echo " test_com.exe. Com-port tester (via built-in interrupts)"
  @echo " test_dev.exe. Com-port tester (via DOS device driver)"
  @echo " com_txrx.exe. Com-port two-way tester (via built-in interrupts)"
  @echo " vmode.exe.... Video-mode switcher"
  @echo " scom.exe..... NPS Simulator, input com-port (built-in), no output"
  @echo " sdev.exe..... NPS Simulator, input com-port (DOS driver), no output"
  @echo " sdev_dly.exe. NPS Simulator, input device/disk (TLM packets), no output"
  @echo " sprp.exe..... NPS Simulator, input propagator, no output"
  @echo " sdsk.exe..... NPS Simulator, input disk, no output"
  @echo " smem.exe..... NPS Simulator, input memory (from disk), no output"
  @echo " prp_dsk.exe.. NPS Generator, input propagator, output disk"
  @echo " prp_dev.exe.. NPS Generator, input propagator, output com-port (DOS driver)"
  @echo " prp_com.exe.. NPS Generator, input propagator, output com-port (built-in)"
  @echo " nps.exe..... NPS Stand-alone, input com-port (built-in)"
  @echo " test_vec.exe. Vector functions tester"
  @echo " map_raw.exe.. Map-file converter, text to raw"
  @echo " emm_test.exe. EMM functions tester"
  @echo " tansdump.exe. TANS data-file dumper"
  @echo " tans_vec.exe. TANS data-file vector dumper"
  @echo " tans_v2.exe.. TANS data-file state-vector dumper"
  @echo " pkt_cln.exe.. Packet file filter/cleaner"
  @echo " pkt_show.exe. Packet file dumper"

#=============================================================
============================
#
clean:
    -$(RM) turboc.cfg
    -$(RM) objs\nps.obj
    -$(RM) objs\nps_dap.obj
    -$(RM) objs\nps_pred.obj
    -$(RM) objs\nps_edit.obj
    -$(RM) objs\sprop.obj
    -$(RM) objs\scom.obj
    -$(RM) objs\scom_dly.obj

#=============================================================
============================
# Object file groups
#
ORBMECH_OBJS = amatrix.obj bgprop.obj cowell.obj der.obj gem9.obj getiers.obj \
  gotpot.obj itowgs84.obj jacatm.obj jacdat.obj m50rnp.obj rk4.obj util.obj

EDIT_OBJS = edit.obj key_get.obj
EDIT_RT_OBJS = $(EDIT_OBJS) edit_rt.obj edit_fld.obj
```

```
TIME_OBJS = thms_dbl.obj tdbl_str.obj tdbl_hms.obj thms_str.obj tgps_utc.obj

VECTOR_OBJS = v_vector.obj v_matrix.obj v_quat.obj v_rvbar.obj \
  v_m50.obj v_j2000.obj v_check.obj

SWAP_OBJS = swap_d.obj swap_f.obj swap_s.obj

EMM_OBJS = emm_lowa.obj emm_cons.obj emm_map.obj emm_mapm.obj

!if $d(GPS)
NPS_OBJS_GPS = nps_m50.obj dif_data.obj dif_plot.obj \
  nps_filt.obj lsfilter.obj nps_dap.obj nps_pred.obj
!endif

NPS_OBJS = nps.obj nps_edit.obj nps_plot.obj menuplot.obj \
  rv_data.obj rv_plot.obj py_data.obj py_plot.obj \
  f_and_g.obj plane.obj nps_dap.obj nps_pred.obj $(NPS_OBJS_GPS)

ORB_PKT_OBJ  = orb_pkt.obj
ORB_PORT_OBJ = orb_port.obj
ORB_FILE_OBJ = orb_file.obj

TANS_PKT2_OBJ = tanspkt2.obj
TANS_PKT_OBJ  = tans_pkt.obj
TANS_PORT_OBJ = tansport.obj
TANS_FILE_OBJ = tansfile.obj

TANS_FILE_SIM_OBJ = tnsprtsm.obj
TANS_PORT_SPDRV_OBJ = tnsprtfn.obj

COM_OBJS = com_intc.obj com_inta.obj

TANS_OBJS = tans51.obj tns_io_f.obj prc_tpkt.obj tans_sys.obj tanstime.obj \
  almanac.obj m50_eph.obj display.obj main_dis.obj para_dis.obj misc_dis.obj \
  map_dis.obj spc_edit.obj misc_fun.obj log.obj exit.obj
TANS_NPS_FAKE = nps_fake.obj

#==========================================================================
==========================
# TANS, with NPS        ***** STS-51 Flight Version *****
#
TANS51_OBJS = $(TANS_OBJS) nps_if.obj $(NPS_OBJS) \
  $(ORB_PKT_OBJ) $(ORB_PORT_OBJ) $(TANS_PORT_OBJ) $(COM_OBJS) \
  $(VECTOR_OBJS) $(TIME_OBJS) $(EDIT_RT_OBJS) $(ORBMECH_OBJS) $(EMM_OBJS)

TANS51_LIBS = $(GRAPHLIB) $(MATHLIB) $(CLIB)

tans.exe:: turboc.cfg
tans.exe:: $(TANS51_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
```

```
$(C0) $**,$<,,$(TANS51_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
# TANS51, with NPS, Sim from file
#
TANS51SM_OBJS = $(TANS_OBJS) nps_if.obj $(NPS_OBJS) \
  $(ORB_PKT_OBJ) $(ORB_FILE_OBJ) $(TANS_FILE_OBJ) $(COM_OBJS) \
  $(VECTOR_OBJS) $(TIME_OBJS) $(EDIT_RT_OBJS) $(ORBMECH_OBJS) $(EMM_OBJS)

TANS51SM_LIBS = $(GRAPHLIB) $(MATHLIB) $(CLIB)

tans51sm.exe:: turboc.cfg
tans51sm.exe:: $(TANS51SM_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(TANS51SM_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
# TANS51, no NPS, with SPDRIVE
#
TANS_SPDRV_OBJS = $(TANS_OBJS) $(TANS_NPS_FAKE) $(TANS_PORT_SPDRV_OBJ)\
  $(EDIT_OBJS)

TANS_SPDRV_LIBS = $(GRAPHLIB) $(MATHLIB) $(CLIB)

tans_spd.exe:: turboc.cfg
tans_spd.exe:: $(TANS_SPDRV_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(TANS_SPDRV_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
# TANS51, no NPS
#
TANS51FN_OBJS = $(TANS_OBJS) $(TANS_NPS_FAKE) \
  $(TANS_PORT_OBJ) $(EDIT_OBJS) $(COM_OBJS)
#  $(TANS_PORT_SPDRV_OBJ)

TANS51FN_LIBS = $(GRAPHLIB) $(MATHLIB) $(CLIB)
```

```
tans51fn.exe:: turboc.cfg
tans51fn.exe:: $(TANS51FN_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(TANS51FN_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
# TANS device simulator
#
TSIM_OBJS = tsim.obj tsim_pkt.obj \
  tsim_2x.obj tsim_3x.obj tsim_5x.obj tsim_ex.obj \
  $(TANS_PORT_OBJ) $(COM_OBJS) $(EDIT_OBJS) $(SWAP_OBJS)

TSIM_LIBS = $(MATHLIB) $(CLIB)

tsim.exe:: turboc.cfg
tsim.exe:: $(TSIM_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(TSIM_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
# TANS data-file dumper
#
TANSDUMP_OBJS = tansdump.obj \
  $(TANS_PKT_OBJ) $(SWAP_OBJS) $(EDIT_OBJS)

TANSDUMP_LIBS = $(MATHLIB) $(CLIB)

tansdump.exe:: turboc.cfg
tansdump.exe:: $(TANSDUMP_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(TANSDUMP_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
# TANS data-file vector dumper
#
TANS_VEC_OBJS = tans_vec.obj \
  $(TANS_PKT_OBJ) $(SWAP_OBJS) $(EDIT_OBJS)
```

```
TANS_VEC_LIBS = $(MATHLIB) $(CLIB)

tans_vec.exe:: turboc.cfg
tans_vec.exe:: $(TANS_VEC_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(TANS_VEC_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
# TANS data-file state-vector dumper
#
TANS_V2_OBJS = tans_v2.obj \
  $(TANS_PKT2_OBJ) $(TANS_PKT_OBJ) $(SWAP_OBJS) $(EDIT_OBJS) $(TIME_OBJS)

TANS_V2_LIBS = $(MATHLIB) $(CLIB)

tans_v2.exe:: turboc.cfg
tans_v2.exe:: $(TANS_V2_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(TANS_V2_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
# Com-port two-way tester (via built-in interrupt buffers)
#
# Run this program on two machines simultaneously
#
COM_TXRX_OBJS = com_txrx.obj $(EDIT_OBJS) com_intc.obj com_inta.obj

COM_TXRX_LIBS = $(CLIB)

com_txrx.exe:: turboc.cfg
com_txrx.exe:: $(COM_TXRX_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(COM_TXRX_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
# Com-port tester (via built-in interrupt buffers)
#
TEST_COM_OBJS = test_com.obj $(EDIT_OBJS) com_intc.obj com_inta.obj
```

```
TEST_COM_LIBS = $(CLIB)

test_com.exe:: turboc.cfg
test_com.exe:: $(TEST_COM_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(TEST_COM_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#============================================================
============================
# Com-port tester (via DOS device driver)
#
TEST_DEV_OBJS = test_dev.obj

TEST_DEV_LIBS = $(CLIB)

test_dev.exe:: turboc.cfg
test_dev.exe:: $(TEST_DEV_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(TEST_DEV_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#============================================================
============================
# Packet file filter/cleaner
#
PKT_CLN_OBJS = pkt_cln.obj $(ORB_PKT_OBJ) $(TIME_OBJS) $(VECTOR_OBJS)

PKT_CLN_LIBS = $(MATHLIB) $(CLIB)

pkt_cln.exe:: turboc.cfg
pkt_cln.exe:: $(PKT_CLN_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(PKT_CLN_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#============================================================
============================
# Packet file dumper
#
PKT_SHOW_OBJS = pkt_show.obj $(ORB_PKT_OBJ) $(TIME_OBJS) $(VECTOR_OBJS)

PKT_SHOW_LIBS = $(MATHLIB) $(CLIB)
```

```
pkt_show.exe:: turboc.cfg
pkt_show.exe:: $(PKT_SHOW_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(PKT_SHOW_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#============================================================
============================
# Vector functions tester
#
TEST_VEC_OBJS = test_vec.obj vect_old.obj $(VECTOR_OBJS) $(ORBMECH_OBJS)

TEST_VEC_LIBS = $(MATHLIB) $(CLIB)

test_vec.exe:: turboc.cfg
test_vec.exe:: $(TEST_VEC_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(TEST_VEC_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#============================================================
============================
# EMM functions tester
#
EMM_TEST_OBJS = emm_test.obj $(EMM_OBJS)

EMM_TEST_LIBS = $(MATHLIB) $(CLIB)

emm_test.exe:: turboc.cfg
emm_test.exe:: $(EMM_TEST_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(EMM_TEST_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#============================================================
============================
# Map-file converter, text to raw
#
MAP_RAW_OBJS = map_raw.obj

MAP_RAW_LIBS = $(MATHLIB) $(CLIB)

map_raw.exe:: turboc.cfg
map_raw.exe:: $(MAP_RAW_OBJS)
```

```
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(MAP_RAW_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
# Video-mode switcher
#
VMODE_OBJS = vmode.obj

VMODE_LIBS = $(CLIB)

vmode.exe:: turboc.cfg
vmode.exe:: $(VMODE_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(VMODE_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
# NPS Stand-alone, input from com-port (built-in)
#
NPSX_OBJS = nps_main.obj scom.obj $(NPS_OBJS) \
  $(ORB_PKT_OBJ) $(ORB_PORT_OBJ) $(COM_OBJS) \
  $(VECTOR_OBJS) $(TIME_OBJS) $(EDIT_RT_OBJS) $(ORBMECH_OBJS) $(EMM_OBJS)

NPSX_LIBS = $(GRAPHLIB) $(MATHLIB) $(CLIB)

nps.exe:: turboc.cfg
nps.exe:: $(NPSX_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(NPSX_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
# NPS Simulator, input from com-port (built-in), no data output
#
SCOM_OBJS = sim.obj scom.obj gnone.obj $(NPS_OBJS) \
  $(ORB_PKT_OBJ) $(ORB_PORT_OBJ) $(COM_OBJS) \
  $(VECTOR_OBJS) $(TIME_OBJS) $(EDIT_RT_OBJS) $(ORBMECH_OBJS) $(EMM_OBJS)

SCOM_LIBS = $(GRAPHLIB) $(MATHLIB) $(CLIB)
```

```
scom.exe:: turboc.cfg
scom.exe:: $(SCOM_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(SCOM_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
# NPS Simulator, input from com-port (DOS device driver), no data output
#
SDEV_OBJS = sim.obj scom.obj gnone.obj $(NPS_OBJS) \
  $(ORB_PKT_OBJ) $(ORB_FILE_OBJ) \
  $(VECTOR_OBJS) $(TIME_OBJS) $(EDIT_RT_OBJS) $(ORBMECH_OBJS) $(EMM_OBJS)

SDEV_LIBS = $(GRAPHLIB) $(MATHLIB) $(CLIB)

sdev.exe:: turboc.cfg
sdev.exe:: $(SDEV_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(SDEV_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
# NPS Simulator, input from com-port (DOS device driver), no data output
#
!if $d(GPS)
SDEV_DLY_OBJS_GPS = $(TANS_PKT2_OBJ) $(TANS_PKT_OBJ) $(TANS_FILE_OBJ) $(SWAP_OBJS)
!endif

SDEV_DLY_OBJS = sim.obj scom_dly.obj timer.obj gnone.obj $(NPS_OBJS) \
  $(ORB_PKT_OBJ) $(ORB_FILE_OBJ) $(SDEV_DLY_OBJS_GPS) \
  $(VECTOR_OBJS) $(TIME_OBJS) $(EDIT_RT_OBJS) $(ORBMECH_OBJS) $(EMM_OBJS)

SDEV_DLY_LIBS = $(GRAPHLIB) $(MATHLIB) $(CLIB)

sdev_dly.exe:: turboc.cfg
sdev_dly.exe:: $(SDEV_DLY_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(SDEV_DLY_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
```

```
# NPS Simulator, input from propagator, no data output
#
SPRP_OBJS = sim.obj sprop.obj timer.obj gnone.obj $(NPS_OBJS) \
  $(VECTOR_OBJS) $(TIME_OBJS) $(EDIT_RT_OBJS) $(ORBMECH_OBJS) $(EMM_OBJS)

SPRP_LIBS = $(GRAPHLIB) $(MATHLIB) $(CLIB)

sprp.exe:: turboc.cfg
sprp.exe:: $(SPRP_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(SPRP_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
# NPS Simulator, input from disk, no data output
#
SDSK_OBJS = sim.obj sdisk.obj timer.obj gnone.obj $(NPS_OBJS) \
  $(VECTOR_OBJS) $(TIME_OBJS) $(EDIT_RT_OBJS) $(ORBMECH_OBJS) $(EMM_OBJS)

SDSK_LIBS = $(GRAPHLIB) $(MATHLIB) $(CLIB)

sdsk.exe:: turboc.cfg
sdsk.exe:: $(SDSK_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(SDSK_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
============================
# NPS Simulator, input from memory (from disk), no data output
#
SMEM_OBJS = sim.obj smem.obj gnone.obj $(NPS_OBJS) \
  $(VECTOR_OBJS) $(TIME_OBJS) $(EDIT_RT_OBJS) $(ORBMECH_OBJS) $(EMM_OBJS)

SMEM_LIBS = $(GRAPHLIB) $(MATHLIB) $(CLIB)

smem.exe:: turboc.cfg
smem.exe:: $(SMEM_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(SMEM_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#=============================================================
```

```
============================
# NPS Generator, input from propagator, output to disk
#
PRP_DSK_OBJS = sim.obj sprop.obj timer.obj gdisk.obj nps_none.obj nps_m50.obj \
  $(VECTOR_OBJS) $(TIME_OBJS) $(EDIT_RT_OBJS) $(ORBMECH_OBJS)

PRP_DSK_LIBS = $(GRAPHLIB) $(MATHLIB) $(CLIB)

prp_dsk.exe:: turboc.cfg
prp_dsk.exe:: $(PRP_DSK_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(PRP_DSK_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#============================================================
============================
# NPS Generator, input from propagator, output to com-port (DOS device driver)
#
PRP_DEV_OBJS = sim.obj sprop.obj timer.obj gcom_dev.obj nps_none.obj nps_m50.obj \
  $(VECTOR_OBJS) $(TIME_OBJS) $(EDIT_RT_OBJS) $(ORBMECH_OBJS)

PRP_DEV_LIBS = $(GRAPHLIB) $(MATHLIB) $(CLIB)

prp_dev.exe:: turboc.cfg
prp_dev.exe:: $(PRP_DEV_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(PRP_DEV_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map

#============================================================
============================
# NPS Generator, input from propagator, output to com-port (built-in)
#
PRP_COM_OBJS = sim.obj sprop.obj timer.obj gcom_com.obj nps_none.obj nps_m50.obj \
  $(COM_OBJS) $(VECTOR_OBJS) $(TIME_OBJS) $(EDIT_RT_OBJS) $(ORBMECH_OBJS)

PRP_COM_LIBS = $(GRAPHLIB) $(MATHLIB) $(CLIB)

prp_com.exe:: turboc.cfg
prp_com.exe:: $(PRP_COM_OBJS)
  $(LD) $(LFLAGS) /L$(LIBPATH) @&&|
$(C0) $**,$<,,$(PRP_COM_LIBS)
|
  @$(TJS_DBUG) /la $<
  $(RM) $*.map
```

```
#============================================================
============================
# Compiler configuration file
#
### -ml -1 $(CDEBUG) -O2 -Ff -f287 -wdef -wnod -wpro -wuse
turboc.cfg: makefile
  copy &&|
-ml -1 $(CDEBUG) $(CEXTRA) -Ff -wdef -wnod -wpro -wuse
-I$(INCPATH) -L$(LIBPATH) -n$(.path.obj)
| turboc.cfg

#============================================================
============================
# Gotpot.c needs special compile: no optimization
#
gotpot.obj: orbmech1\gotpot.c
  $(CC) -c -O2-v $?
```