I/O Communication Interface

of APA512S

with VISION_CTL

by

Sunil Fotedar
Sr. Associate Engineer
Lockheed Engineering and Sciences Company
Houston, Texas
August-September, 1990

INTRODUCTION

For any command there is a response. All commands and responses, the protocol defined by System Services, are sent over the communication link as a message header. The message header contains the codes to define the source, destination, and the type of the function needed to be performed. The message header looks like the following:

source,destination,function,modifier,length,message.

The codes are separated by commas, and are all represented as integers (except message can be represented as integers or reals). If no information is to be transmitted in the modifier code, a NUL is sent as the modifier. The length is the length of the message portion in bytes. All integers are 32-bit unless otherwise specified. All reals are 32-bit floating point. A zero message length may be sent in the case of no message portion of the message header.

Symbolic names composed of all capital letters are used to represent the source codes, destination codes, function codes, and many of the parameters. These are listed in the include file *vision.h*. Each of these symbolic names is associated with some integer value that is actually used in the given message.

The source code is in four separate files: *evar.c*, *io_com.c*, *init.c*, and *track.c*. *evar.c* is the main file, all others are subroutines. *io_com.c* has software for I/O communications over RS232 line; *init.c* has software to initialize, enable, and disable the image processing boards; and *track.c* has software for tracking algorithms.

In order to run the program, simply type *run* at the prompt. If any changes are to be made in the subroutines, type *umacs* <filename> at the prompt, and type <Ctrl-Z> to save the changes and to get back to command line.

In its current configuration, the programs can read upto two message headers concatenated together, respond to each one at a time.

SOURCE CODE

```
/*****************************************************************************
* Author   :   Sunil Fotedar                                                 *
* Place    :   LESC, Houston, TX                                             *
* Abstract :   To develop a communication link between vision*
*              subsystem of EVAR and APA512S, and perform the*
*              blob analysis.                                                *
* Date     :                                                                 *
*****************************************************************************/

/* Header files */

#include <stdio.h>
#include <time.h>
#include <modes.h>
#include <std_def.h>
#include <vfConfig.h>
#include <kernel.h>
#include <apHead.h>
#include <blobHead.h>
#include <discrim.h>
#include <aai.h>
#include <user.h>
#include <vision.h>

int ctrl_C=FALSE;

char    message_buffer[WIDE], tempmsg[WIDE];

struct  blob_structure bst[BLOBS];
struct  blob_structure_short sst[BLOBS];
Message msg;

PARAM_FLAGS     flags;
BLOB_DESC       *bd1, *bd2, *bdtemp;
CONNECT_BIDS    boards;
int             conFlags;

long    from  = (long)(APA512S);   /* source identification */
long    to    = (long)(VISION_CTL);  /* destination identification */
int     display_data = NO;
int     time_diff = 0;
int     threshold = 0x80;
int     target_acq_auto_flag = UNSET, frame_count;
int     ret_en=-1;
int     read_port, write_port;


/* Beginning of main() */

main(argc, argv)

int     argc;
char    **argv;

{
   int        i, j, count, model, camera, ret, ret1=-99, no_blobs, code;
   int        cam=0, time_ret, respond_ret, blob_ret;
   long       cmd, val, len, funct;
```

```
  /* Command-line options */

if(argc > 1) {
       if((strncmp(argv[1],"HELP",4) == SUCCESS) ||
             (strncmp(argv[1],"help",4) == SUCCESS) ||
             (strncmp(argv[1],"?",1) == SUCCESS)) {
                help_message();
                exit(0);
       }
       for(i=1;i<argc;++i) {
          if (strncmp(argv[i],"DATA",4) == SUCCESS) display_data = YES;
          if (strncmp(argv[i],"data",4) == SUCCESS) display_data = YES;
       }
   }

/* Open I/O port */

       if((read_port=initialize_io_read())==FAILURE) {       /* open I/O port to read */
       printf("APA512S Problem in opening the I/O port.\n");
       goto apa512s_exit;
   }

   if((write_port=initialize_io_write())==FAILURE) {       /* open I/O port to write */
       printf("APA512S Problem in opening the I/O port.\n");
       goto apa512s_exit;
   }

/* Initilialize APA512S boards */

   if((ret=initialize_boards())!=SUCCESS) {
       printf("APA512S Problem in initializing APA512S boards.\n");
       goto apa512s_exit;
   }

/* Initialize buffers */

       memset(message_buffer,0x00,WIDE);
       memset(tempmsg,0x00,WIDE);
       memset(bst,0x00,BLOBS);
       memset(sst,0x00,BLOBS);


/* Wait for initialization message */

   initialize:

   target_acq_auto_flag = UNSET;

   printf("\nAPA512S Waiting for RS232 initialization sequence!!!\n");

       if((ret=read_initialization())!=SUCCESS) {
          printf("APA512S Problem in reading the RS232 initialization sequence.\n");
          goto apa512s_exit;
       }
       printf("\nAPA512S Syn'c message read: %s\n",message_buffer);
       printf("\nAPA512S Ready to go to work!!!\n");
       if((ret=send_message())!=SUCCESS) {
          printf("APA512S Problem in sending data out.\n");
          goto apa512s_exit;
       }
       send_return(I_AM_HERE_TODAY,NUL);

while(1)
```

```
{

   memset(message_buffer,0x00,WIDE);
   memset(tempmsg,0x00,WIDE);
   memset(bst,0x00,BLOBS);

   printf("\nAPA512S Waiting for a command from Vision Subsystem!!!\n");

   check_message();

   sscanf(message_buffer,"%ld,%ld,%ld,%ld,%ld,",
                  &msg.source,&msg.destin,&msg.functn,&msg.modifer,
                  &msg.length);

   printf("length=%lx\n",msg.length);
   if(display_data)
      printf("APA512S Message read (in Hex): %lx,%lx,%lx,%lx,%lx.\n",
               msg.source,msg.destin,msg.functn,msg.modifer,msg.length);

            respond_ret = respond();
            if(respond_ret==SUCCESS) goto initialize;

/* Handle all other commands */

   commands:

   target_acq_auto_flag = UNSET;


switch((int)(msg.functn)) {

      case   STOP_BLOB_TRACK :

            printf("APA512S Blob Track is stopped.\n");
            break;

      case   BLOB_DETECT_AND_TRACK :

               printf("APA512S BLOB_DETECT_AND_TRACK is received. No action is taken.\n");
            break;

      case   TARGET_ACQ_MANUAL :
               /* msg.modifer = camera # = sourceid */

            (int)bst[0].sourceid = (int)msg.modifer;
            if(((int)bst[0].sourceid<CAMERA_LEFT)||
               ((int)bst[0].sourceid>CAMERA_MAPPER_VIDEO))
                     (int)bst[0].sourceid = CAMERA_CENTER;

               ret = video_switch();

            if(ret==VIDEO_SWITCHED) {
                  printf("\nAPA512S Video is from Camera# %d.\n",
                           (int)bst[0].sourceid);

                  ret = image_process(&no_blobs);

                  send_status(ret,no_blobs);

            }
            else if(ret==SUCCESS) goto initialize;
            else if(ret==FAILURE) goto commands;
            break;
```

```
case    TARGET_ACQ_AUTO :

   target_acq_auto_flag = SET;
   model = (int)msg.modifer;   /* model */

   for(cam=CAMERA_LEFT;cam<=CAMERA_LHAND;cam++)
   {
      (int)bst[0].sourceid = cam;
      ret = video_switch();

      if(ret==VIDEO_SWITCHED) {
            printf("\nAPA512S Video is from Camera# %d.\n",cam);

            bst[0].sourceid = cam;
            bst[0].idcode = model;
            ret = image_process(&no_blobs);

            send_status(ret,no_blobs);

      }
      else if(ret==SUCCESS) goto initialize;
      else if(ret==FAILURE) goto commands;

         ret1 = check_for_message();
         if(ret1==SUCCESS) goto initialize;
         else if(ret1==FAILURE) goto commands;
   }
   target_acq_auto_flag = UNSET;
   break;

case    BLOB_DETECT :

      for(cam=CAMERA_LEFT;cam<(CAMERA_LHAND+2);cam++) {

         if(cam==(CAMERA_LHAND+1)) cam=CAMERA_LEFT;

            (int)bst[0].sourceid = cam;

            ret = video_switch();

      if(ret==VIDEO_SWITCHED) {
            printf("\nAPA512S Video is from Camera# %d.\n",cam);

            (int)bst[0].sourceid = cam;

            ret = image_process(&no_blobs);

            send_status(ret,no_blobs);

      }
      else if(ret==SUCCESS) goto initialize;
      else if(ret==FAILURE) goto commands;
         ret1 = check_for_message();
         if(ret1==SUCCESS) goto initialize;
         else if(ret1==FAILURE) goto commands;
      }    /* end of infinite loop */

      break;

case    BLOB_TRACK :

      memset(tempmsg,0x00,WIDE);
   sprintf(tempmsg,"%ld,%ld,%ld,%ld,%ld,",
         msg.source,msg.destin,msg.functn,msg.modifer,msg.length);
```

```
    sscanf(message_buffer+strlen(tempmsg),"%f,%f,%f,%f,%f,%f,%ld,",
      &bst[0].azm_centroid,&bst[0].elv_centroid,
      &bst[0].rng_centroid,&bst[0].height_env,
      &bst[0].width_env,&bst[0].depth_env,&bst[0].sourceid);

  printf("APA512S Track parameters are: %f,%f,%f,%f,%f,%f,%ld,\n",
      bst[0].azm_centroid,bst[0].elv_centroid,
      bst[0].rng_centroid,bst[0].height_env,
      bst[0].width_env,bst[0].depth_env,bst[0].sourceid);

    if(((int)bst[0].sourceid<CAMERA_LEFT)
      ||((int)bst[0].sourceid>CAMERA_MAPPER_VIDEO))
          (int)bst[0].sourceid = CAMERA_CENTER;

    radian_to_pixel();

    ret = video_switch();

    if(ret==VIDEO_SWITCHED) {  /* 0x1116 received */
      printf("\nAPA512S Video is from Camera# %d.\n",(int)bst[0].sourceid);

      frame_count = 0;
                  /* Reset frame counter */
    for(;;) {

      track:

    cam = (int)bst[0].sourceid;
  printf("before blob_track, source:%d\n",(int)bst[0].sourceid);

  blob_ret = blob_track(&no_blobs,&camera);

    (int)bst[0].sourceid = camera;

  printf("no_blobs = %d, camera = %d, source = %d, cam=%d.\n",
          no_blobs, camera, (int)bst[0].sourceid, cam);
      if(camera != cam) {
        ret = video_switch();
      if(ret==VIDEO_SWITCHED) {
        printf("\nAPA512S Video switched to Camera# %d.\n",
              (int)bst[0].sourceid);
        goto track;
      }
      else if(ret==SUCCESS) goto initialize;
      else if(ret==FAILURE) goto commands;
      } /* end of if(camera) */
  send_status(blob_ret,no_blobs);

  ret1 = check_for_message(); /* to get out of infinite loop */
  if(ret1==SUCCESS) goto initialize;
  else if(ret1==FAILURE) goto commands;

    }  /* end of infinite loop */
  } /* end of if(ret==VIDEO_SWITCHED) after initial video switch */
    else if(ret==SUCCESS) goto initialize;
    else if(ret==FAILURE) goto commands;

  break;


case    DETERMINE_GRASP :

    send_return(NO_GRASP_REGION,(int)bst[0].sourceid);
```

```
            printf("APA512S NO_GRASP_REGION is sent.\n");

         break;

      case    MONITOR_GRASP :

            send_return(NO_GRASP_REGION,(int)bst[0].sourceid);
            printf("APA512S NO_GRASP_REGION is sent.\n");

         break;

      default:
         if(respond_ret==FAILURE)
            printf("\nAPA512S Command not understood. Try Again!!!\n");

         break;
   }

}  /* End of while(1) */

apa512s_exit:              /* exit the program */

   printf("\nAPA512S Exiting ... Try again!!!\n\n");
   free(tempmsg);
   free(message_buffer);
   free(bst);
   free(sst);
   close(read_port); /* close the read port */
   close(write_port); /* close the write port */
   exit(0);

} /* End of main() */

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


respond()

{
   int     ret, funct;

   funct = (int)(msg.functn);

   switch(funct) {

      case    0:
            return(IGNORE); /*faulty message- something that doesn't make sense */

      case    PCIF_START_SENDING_VISUAL_RESULTS:
            return(IGNORE);  /* ignore, take no action */

      case    PCIF_STOP_SENDING_VISUAL_RESULTS:
            return(IGNORE);  /* ignore, take no action */

      case    SET_THRESHOLD_VALUE:

            threshold = (int)msg.modifer;
            if(threshold > 255) threshold = 255;
            printf("APA512S Threshold is set to %d.\n",threshold);

            disable_boards();
            if((ret_en=enable_boards())!=SUCCESS)
               printf("APA512S There's a problem in enabling APA512S.\n");
            else
```

```
            printf("APA512S The APA512S has been enabled.\n");
        return(IGNORE);

case    ENABLE1:

    if((ret_en=enable_boards())!=SUCCESS)
        printf("APA512S There's a problem in enabling APA512S.\n");
    else
        printf("APA512S The APA512S has been enabled.\n");

    return(IGNORE);

case    DISABLE1:

        if((ret=disable_boards())!=SUCCESS)
            printf("APA512S There's a problem in disabling APA512S.\n");
        else {
            printf("APA512S The APA512S has been disabled.\n");
            return(SUCCESS);
        }
        break;

case    RESET1:

        if((ret=disable_boards())!=SUCCESS)
            printf("APA512S There's a problem in resetting APA512S, and ABNORMAL is sent.\n");
        else {
            printf("APA512S The APA512S is reset.\n");
            return(SUCCESS);
        }
        break;

case    ARE_YOU_HERE_TODAY :

        if(ret_en!=SUCCESS) {
            send_return(NOT_HERE,NUL);
            printf("APA512S NOT_HERE NUL is sent.\n");
        } else {
            send_return(I_AM_HERE_TODAY,NUL);
            printf("APA512S I_AM_HERE_TODAY NUL is sent.\n");
        }
        return(IGNORE);

case    SEND_HEALTH_STATUS :

        if(ret_en!=SUCCESS) {
            send_return(HEALTH_STATUS,ABNORMAL);
            printf("APA512S HEALTH_STATUS ABNORMAL is sent.\n");
        } else {
            send_return(HEALTH_STATUS,NOMINAL);
            printf("APA512S HEALTH_STATUS NOMINAL is sent.\n");
        }
        return(IGNORE);

case    IDLE :

            printf("APA512S IDLE is received. No action is taken.\n");
        return(IGNORE);

case    RESUME :

            printf("APA512S RESUME is received. No action is taken.\n");
        return(IGNORE);
```

```
        case    START_PROCESSING :

                  printf("APA512S START_PROCESSING is received. No action is taken.\n");
                  return(IGNORE);

        case    SINGLE_FRAME :

                  printf("APA512S SINGLE_FRAME is received. No action is taken.\n");
               return(IGNORE);

        case    SET_TIME :

               time_diff = (int)msg.modifer - (int)time(NULL)*TICKS;
               printf("APA512S Clock has been set to %ld ticks.\n",msg.modifer);

               return(IGNORE);

        case    B14_VIDEO_SELECT_COMPLETE : /* We receive 0x1116 here */
               return(VIDEO_SWITCHED);

        default:

           return(FAILURE);  /* to handle target_manual, target_auto, etc. -
                        all other commands in main */
     }

}
/*****************************************************************************/
check_for_message()

{
     int     count, ret;

                  memset(message_buffer,0x00,WIDE);
     if((count=read_status())>5) {
        read_message(count);
        sscanf(message_buffer,"%ld,%ld,%ld,%ld,%ld,",
               &msg.source,&msg.destin,&msg.functn,&msg.modifer,&msg.length);

        if(display_data)
        printf("APA512S Message read inside the loop (in Hex): \n%lx,%lx,%lx,%lx,%lx.\n",
                  msg.source,msg.destin,msg.functn,msg.modifer,msg.length);
     }
                  ret = respond();
                  return(ret);
}
/*****************************************************************************/
```

```
/*****************************************************************************
* Author   :   Sunil Fotedar                                                 *
* Place    :   LESC, Houston, TX                                             *
* Abstract :   To develop a communication link between vision*
*             subsystem of EVAR and APA512S.*
* Date     :                                                                 *
*****************************************************************************/

/* Header files */

#include <stdio.h>
#include <time.h>
#include <modes.h>
#include <vision.h>

extern char message_buffer[], tempmsg[];
extern long from, to;
extern int display_data, read_port, write_port;
extern struct blob_structure bst[];
extern struct blob_structure_short sst[];
extern Message msg;

/*************************************************************************/
/* To open an I/O Port                                                   */
/*************************************************************************/

initialize_io_read()

{
   int     path;
   int     open();

   if((path=open(IO_COM,S_IREAD)) < 0) {
      return(FAILURE);
   }
   return(path);
}

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

initialize_io_write()

{
   int     path;
   int     open();

   if((path=open(IO_COM,S_IWRITE)) < 0) {
      return(FAILURE);
   }
   return(path);
}

/*************************************************************************/
/* To check if there's anything at the port ready to be read.       */
/*************************************************************************/

read_status()

{
   int     _gs_rdy();
   int     count;

   if((count=_gs_rdy(read_port)) <= 0) {
```

```
      return(FAILURE);
   }
   else    return(count);
}

/*************************************************************************/
/* To read from an opened port.                                          */
/*************************************************************************/

read_message(count)

int     count;

{
      int len, read();

memset(message_buffer,0x00,WIDE);

if((len=read(read_port,message_buffer,count)) < 0)   return(FAILURE);

return(SUCCESS);
}

/*************************************************************************/
/* Read in the initialization message, and verify.                       */
/*************************************************************************/

read_initialization()

{
   int     ret;

   memset(message_buffer,0x00,WIDE);

   for(;;) {

   if((ret=check_message())<0)
      ret = FAILURE;
   else ret = SUCCESS;
   if((message_buffer[strlen(message_buffer)-2]=='?')&&
            (message_buffer[strlen(message_buffer)-1]=='!'))
         ret = SUCCESS;
   else if((message_buffer[strlen(message_buffer)-2]=='!')&&
            (message_buffer[strlen(message_buffer)-1]=='?'))
         ret = SUCCESS;
   else
         ret = FAILURE;

   if(ret==SUCCESS) goto read_exit;
      sleep(1);
   }
   read_exit:
      return(SUCCESS);
}

/*************************************************************************/
/* Check (and wait) for incoming message.                                */
/*************************************************************************/

check_message()

{
   int     count, ret;
```

```
        memset(message_buffer,0x00,WIDE);
        memset(tempmsg,0x00, WIDE);

for(;;) {
        if((count=read_status()) < 1)  ret = FAILURE;
        else if((ret=read_message(count)) < 0)  ret = FAILURE;
        else ret = SUCCESS;
        if (ret==SUCCESS) goto read_exit;
        sleep(1);
}
    read_exit:
        return(SUCCESS);
}

/*************************************************************************/
/* To write to an opened port.                                           */
/*************************************************************************/

send_message()

{
    int len;
    int write();

    if((len=write(write_port,message_buffer,strlen(message_buffer))) < 0) {

        return(FAILURE);
    }
        return(SUCCESS);
}

/*************************************************************************/
/* Make and send header message.                                         */
/*************************************************************************/

make_header(cmd,val,len)

long    cmd, val, len;

{
    int     ret;

    memset(message_buffer,0x00,WIDE);
    sprintf(message_buffer,"%ld,%ld,%ld,%ld,%ld,%s",from,to,cmd,val,len,tempmsg);
    if((ret=send_message()) != SUCCESS)
        printf("APA512S There's a problem in sending data out.\n");

    memset(message_buffer,0x00,WIDE);

}

/*************************************************************************/
/* Video switch request.                                                 */
/*************************************************************************/

video_switch()

{
    int     i, ret, ret1, count, comma_count, send_message();
    long    cmd, val, len;
    long    dummy1,dummy2,dummy3,dummy4,dummy5;

    cmd  = (long)(B14_VIDEO_SELECT_CHANNEL);
    val  = (long)bst[0].sourceid;
```

```
   memset(message_buffer,0x00,WIDE);
   memset(tempmsg,0x00,WIDE);

   strcpy(tempmsg,""); /*No message*/
   len  = 0l;

   make_header(cmd,val,len);

   check_message(); /* expect to get 0x1116 back */
/* count number of commas */
   comma_count = 0;
   for(i=0;i<strlen(message_buffer);i++)
         if(message_buffer[i]==',') comma_count++;
   printf("\nNo. of commas=%d.\n",comma_count);

   if(comma_count>8) {
   sscanf(message_buffer,"%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,",
            &msg.source,&msg.destin,&msg.functn,&msg.modifer,&msg.length,
         &dummy1,&dummy2,&dummy3,&dummy4,&dummy5);
   printf("APA512S Message read inside video switch:\n");
   printf("%s\n",message_buffer);
      ret = respond();
      if(ret==SUCCESS) return(SUCCESS); /* to handle disable,reset */
      else if(ret==FAILURE) return(FAILURE); /* to handle target_man,_auto,etc.*/
      else if((ret==VIDEO_SWITCHED)||(ret==IGNORE))
      {
         msg.functn = dummy3;
            ret1 = respond();
            if((ret==IGNORE)&&(ret1==IGNORE)) return(IGNORE);
             else if(ret1==SUCCESS) return(SUCCESS);
            else if(ret1==FAILURE) return(FAILURE);
            else return(VIDEO_SWITCHED);
      } /* end of else if */
} /* end of if(comma_count) */
else {
      sscanf(message_buffer,"%ld,%ld,%ld,%ld,%ld,",
            &msg.source,&msg.destin,&msg.functn,&msg.modifer,&msg.length);
            ret = respond();
            return(ret);
} /* end of else */
}
/*************************************************************************/
/* Send Returned Values.                                                 */
/*************************************************************************/

send_return(retu, mod)

int     retu, mod;


{
   int     ret, send_message();
   long    len;

   memset(tempmsg,0x00,WIDE);

   strcpy(tempmsg,"");

   make_header((long)retu,(long)mod,0l);

}
/*************************************************************************/
/* Send blob dump.                                                       */
```

```
/*********************************************************************/

send_blob_dump(val)

int     val;

{
   int     i, j, k, delta, ret, cam, send_message();
   long    cmd, len, funct;

   cmd  = (long)(SENDING_BLOBS);

   pixel_to_radian(val);

   memset(message_buffer,0x00,WIDE);
   memset(tempmsg,0x00,WIDE);

   switch((int)(cmd)) {

      case 0l :

            printf("APA512S Exiting... Try Again.\n");
            exit(0);

      case SENDING_BLOBS :

         memset(tempmsg,0x00,WIDE);

      if(val>MAXBLOBS) val = MAXBLOBS;

for(i=0;i<val;i++) {
   sprintf(&tempmsg[strlen(tempmsg)],

"%ld,%ld,%ld,%ld,%ld,%ld,%ld,%f,%f,%.1f,%.1f,%.1f,%.1f,%f,%f,%.1f,%f,%f,%.1f,%f,%f,%.1f,%.1f,%.1f,%.1f
,",
      bst[i].blobnum,bst[i].lblobnum,bst[i].procid,bst[i].timestamp,
      bst[i].sourceid,bst[i].idclass,bst[i].idcode,bst[i].quality,
      bst[i].azm_centroid,bst[i].elv_centroid,bst[i].rng_centroid,
      bst[i].roll_centroid,bst[i].pitch_centroid,bst[i].yaw_centroid,
      bst[i].height_env,bst[i].width_env,bst[i].depth_env,
      bst[i].azm_env_center,bst[i].elv_env_center,bst[i].rng_env_center,
      bst[i].azm_velocity,bst[i].elv_velocity,bst[i].rng_velocity,
      bst[i].roll_velocity,bst[i].pitch_velocity,bst[i].yaw_velocity);

   }
      len=(long)(strlen(tempmsg));

      make_header((long)cmd,(long)val,len);

      if(display_data)
         printf("APA512S Message sent:\n%s\n",tempmsg);

      break;

      case SENDING_BLOBS_SHORT :

            memset(tempmsg,0x00,WIDE);

   delta = MAXBLOBS;
for(i=0;i<=(int)(val/MAXBLOBS);i++) {
   if(i==(int)(val/MAXBLOBS))  delta = val%MAXBLOBS;
   for(j=0;j<delta;j++) {
   sprintf(&tempmsg[strlen(tempmsg)],
      "%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%f,%f,%f,%f,%f,%f,%f,%f,%f,",
```

```
      sst[i*MAXBLOBS+j].blobnum,sst[i*MAXBLOBS+j].lblobnum,sst[i*MAXBLOBS+j].procid,sst[i*MAX-
BLOBS+j].timestamp,
      sst[i*MAXBLOBS+j].sourceid,sst[i*MAXBLOBS+j].idclass,sst[i*MAXBLOBS+j].idcode,sst[i*MAX-
BLOBS+j].quality,
      sst[i*MAXBLOBS+j].azm_centroid,sst[i*MAXBLOBS+j].elv_centroid,sst[i*MAXBLOBS+j].rng_centroid,
      sst[i*MAXBLOBS+j].roll_centroid,sst[i*MAXBLOBS+j].pitch_centroid,sst[i*MAXBLOBS+j].yaw_centroid,
      sst[i*MAXBLOBS+j].height_env,sst[i*MAXBLOBS+j].width_env,sst[i*MAXBLOBS+j].depth_env);
}

      len = (long)(strlen(tempmsg));

      make_header((long)cmd,(long)delta,len);
      memset(tempmsg,0x00,WIDE);
      if(display_data)
         printf("APA512S Message sent:\n%s\n",tempmsg);
      }
         break;

      case    SENDING_GRASP_REGIONS :

         break;

      case    SENDING_GRASP_MONITORS :

         break;

      default :

         printf("\nAPA512S Command not understood. Try Again!!!\n");
         break;
}

}

/*************************************************************************/
/* Send variety of messages based on return values from image       */
/* processing routines.                                             */
/*************************************************************************/

send_status(ret,no_blobs)

int ret,no_blobs;

{
   int     cam;

         cam = (int)bst[0].sourceid;

         if(ret<0) {
            send_return(HEALTH_STATUS,ABNORMAL);
            printf("APA512S Error snapping data into APA512 boards, and ABNORMAL is sent.\n");
         } else if((no_blobs==0)||(ret==NO_BLOBS_FOUND)) {
            send_return(NO_BLOBS_FOUND,cam);
            printf("APA512S NO_BLOBS_FOUND is sent.\n");
         } else if (ret==TARGET_NOT_IN_SCENE) {
            send_return(TARGET_NOT_IN_SCENE,cam);
            printf("APA512S TARGET_NOT_IN_SCENE is sent.\n");
            if(no_blobs!=0) {
               send_blob_dump(no_blobs);
               if(display_data)
                  printf("APA512S %d blobs sent.\n",no_blobs);
            }
         } else {
            send_blob_dump(no_blobs);
```

```
            if(display_data)
                printf("APA512S %d blobs recognized.\n",no_blobs);
        }
}
/*****************************************************************************/
```