3810
55P

NASA Technical Memorandum 109094

Thermal Enclosure System Functional Simulation User's Manual

A. Terry Morris
Langley Research Center, Hampton, Virginia

(NASA-TM-109094) THERMAL ENCLOSURE SYSTEM FUNCTIONAL SIMULATION USER'S MANUAL (NASA. Langley Research Center) 55 p

N94-29489

Unclas

G3/66 0003810

March 1994

National Aeronautics and
Space Administration
Langley Research Center
Hampton, Virginia 23681-0001

Table of Contents

Thermal Enclosure System

Functional Simulation

USER'S MANUAL

ABSTRACT

A form and function simulation of the thermal enclosure system (TES) for a microgravity protein crystal growth experiment has been developed as part of an investigation of the benefits and limitations of intravehicular telerobotics to aid in microgravity science and production. A user can specify the time, temperature, and sample rate profile for a given experiment, and menu options and status are presented on an LCD display. This report describes the features and operational procedures for the functional simulation.

1. INTRODUCTION

NASA Langley Research Center is conducting research on benefits and limitations of the application of telerobotics to support microgravity science and production. The use of telerobotics can potentially increase scientific operations during unmanned periods and assist astronauts with routine tasks during man-tended periods. Methods for interactive monitoring and control of space experiments will be developed, to increase the experimenter's ability to interact with the flight system from the ground, and enhance onboard system sensing and task capability.

The Intravehicular Automation and Robotics (IVAR) laboratory will be used to develop and demonstrate the technology and telerobotic techniques needed to minimize onboard human resources for experiment operation and monitoring, and to provide the experimenter on the ground a more direct role in the control and error recovery for his experiment. This will be accomplished by application of knowledge-based expert system techniques for monitoring processes, diagnosis of problems and system errors, and support for the experimenter or operator in recovering. Full-size mockups of selected laboratory experiments will be developed for the IVAR testbed, which contains a full-size mockup of a space station laboratory module with an internal telerobotic logistics system. Several full size mockups of selected laboratory experiments will be developed and evaluated in the IVAR testbed.

The first task is a protein crystal growth experiment, where fluids are mixed and crystals are grown inside a thermal enclosure system (TES). The flight TES was designed by Space Industries, Inc. for specific use within the mid-deck of the space shuttle. Space Industries also developed the full size mockup for the IVAR laboratory. The mockup is designed to operate similarly to the flight system with respect to controls and displays. The following sections describe the capabilities of the thermal enclosure system mockup and its associated menu-driven software control.

2. Description of Flight Thermal Enclosure System

The flight thermal enclosure system was designed to provide a stable thermal environment for space related experiments. The TES was designed by Space Industries, Inc. for specific use in the mid-deck of the Space Shuttle. The TES cavity is hermetically sealed and can be used to service experiments such as materials solidification and protein crystal growth. Programmed timelines command the TES to change temperature and to collect temperature data at specified intervals. An experimenter may also command the flight TES with the four-button keypad and view software status through a four line forty character LCD display.

3. Operation of Simulated TES

The TES mockup operates through an 80535 microprocessor. The microprocessor sends control and data information to a two-line forty character LCD. A keypad is also provided. Four momentary switches (buttons) serve as input devices to the microprocessor. The buttons are numbered from one to four from left to right, respectively. Figure 3.1 displays the TES mockup interface.

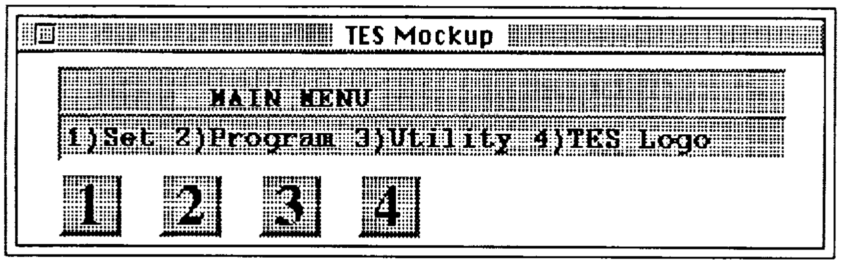


Figure 3.1. TES mockup interface

Software flow of the TES is menu driven. The mockup software is similar to the flight TES software, except that there is no interaction between the physical thermal electric devices (TEDs), the thermistors, and the microprocessor. The following sections describe each menu of the TES software.

3.1 Display Logo

Once the TES is powered, the program initializes the LCD by selecting the 5X7 dot by 8 bit function set and turning the display on. The microprocessor then outputs the TES Display Logo to the LCD. The logo is designed to shift left and wrap-around the display. Figure 3.2 shows the TES Display Logo. A user may exit the display logo by selecting any of the four buttons.

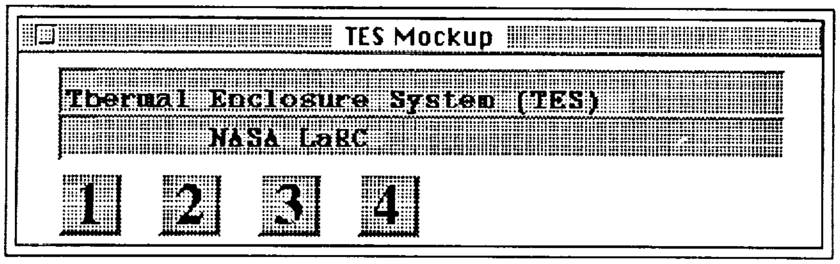


Figure 3.2. TES Display Logo

3.2 Main Menu

The TES Main Menu (Figure 3.3) is the base menu for thermal enclosure system operations. Button #1 selects the Set Menu, #2 selects the Program Menu, #3 selects the Utility Menu, and #4 activates the Display Logo. The Set Menu allows the user to set current TES variables. The Program Menu allows the user to view, edit, and delete timeline programs. The Utility Menu provides system status and an option for locking or unlocking the keypad.

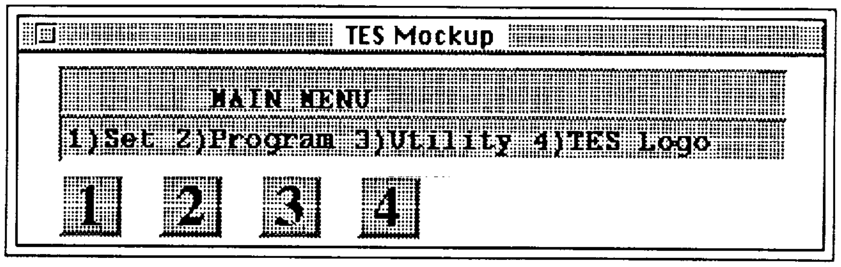


Figure 3.3. TES Main Menu

3.3 Set Menu

The Set Menu (Figure 3.4) allows the user to select three TES parameters to change. Button #1 selects the Set Temperature Menu, #2 selects the Set Thermistors Menu, #3 selects the Set Sample Rate Menu, and #4 returns the operator to the Main Menu.

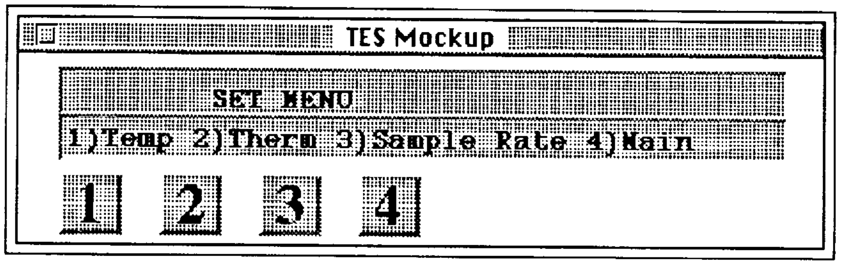


Figure 3.4. Set Menu

3.3.1 Set Temperature Menu

The Set Temperature Menu (Figure 3.5) allows the user to increment or decrement the current temperature. An option (toggle X/.X) is available for the user to select the increment/decrement factor. If 'X' is selected, the microprocessor increments or decrement by one degree (1). If '.X' is selected, the microprocessor increments or decrements by point one degrees (.1). When the TES mockup is reset, the X factor (1 degree) is selected by default. Within the Set Temperature Menu, the current temperature is displayed to the far right of the second line of the LCD. Button #1 increments the temperature, #2 decrements the temperature, #3 toggles the increment/decrement factor, and #4 returns Back to the Set Menu.

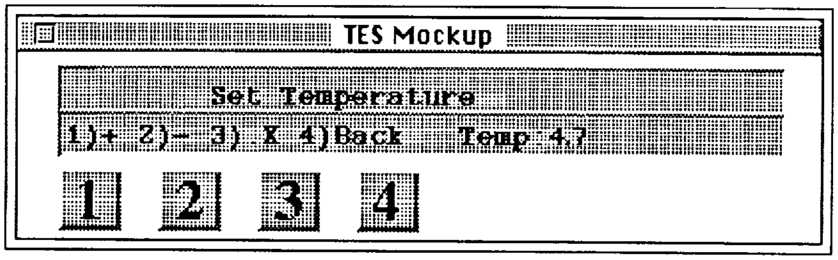
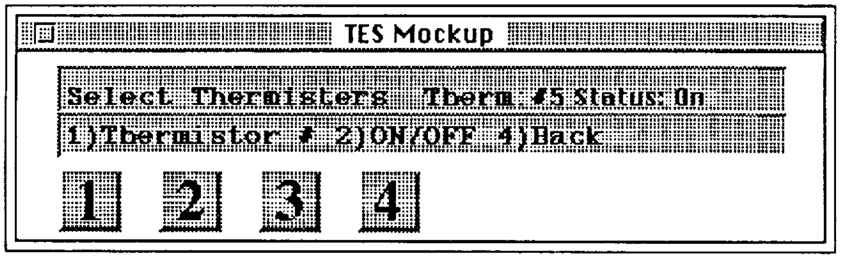


Figure 3.5. Set Temperature Menu

3.3.2 Set Thermistors Menu

The Set Thermistors Menu (Figure 3.6) allows the user to activate several of the thermistors. Button #1 selects the next thermistor to activate. The thermistors are numbered from #1 to #10. Once thermistor #10 is reached, the system will cycle to thermistor #1. Button #2 toggles to activates ('ON') or deactivate ('OFF') the current thermistor. All thermistors are deactivated at reset. The current thermistor number is displayed in the middle of the first line of the LCD. The thermistor status is displayed to the far right of the same line. Button #4 return the user Back to the Set Menu.

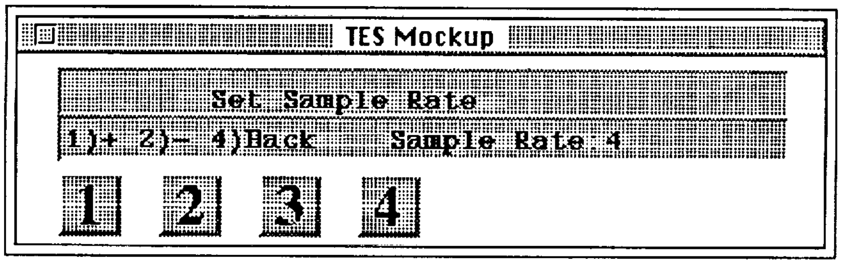


Select Thermisters Therm #5 Status: On
1)Thermistor # 2)ON/OFF 4)Back

Figure 3.6. Set Thermistors Menu

3.3.3 Set Sample Rate Menu

The Set Sample Rate Menu (Figure 3.7) allows the user to increment or decrement how often (in minutes) to sample the thermistors. When the TES mockup is initialized, the default sample time is set to 10 (1 sample every 10 minutes). The maximum sample time is 999 minutes (equivalent to 1 sample every 16 hours and 39 minutes). The LCD displays the current sample time to the far right of the second line. Button #1 increments the sample time by 1, #2 decrements the sample time by 1 minute, and #4 returns the user Back to the Set Menu.

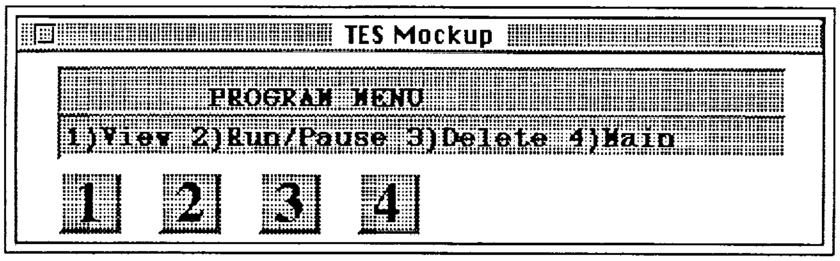


Set Sample Rate
1)+ 2)- 4)Back Sample Rate 4

Figure 3.7. Set Sample Rate Menu

3.4 Program Menu

The Program Menu (Figure 3.8) allows the user to view, edit, and delete timeline programs. Section 2.4.1 explains the purpose and features of a timeline program. The user may also halt execution of a currently running timeline program. Button #1 selects the View Menu, #2 selects the Run/Pause Menu, #3 deletes the current timeline program, and #4 returns the operator to the Main Menu.

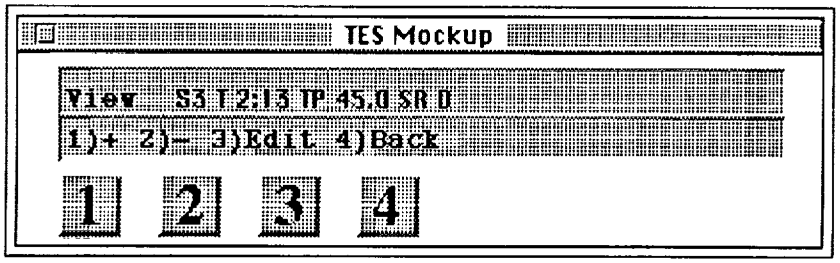


```
PROGRAM MENU
1)View 2)Run/Pause 3)Delete 4)Main
```

Figure 3.8. Program Menu

3.4.1 View Menu

The View Menu (Figure 3.9) allows the user to view or edit a timeline sequence. The top line of the LCD displays four parameters of the timeline sequence. The parameters are sequence number, time value, temperature setting, and sample rate setting, respectively. Button #1 increments the timeline sequence by simply displaying the timeline parameters as the sequence number increases from one to ten. If the user selects Button #1 again when the sequence displays the tenth line of parameter values, the sequence starts over, that is, displaying sequence one parameter values. The user may define a total of ten sequences per timeline.



```
View   S3 T2:13 TP 45.0 SR 0
1)+ 2)- 3)Edit 4)Back
```

Figure 3.9. View Menu

Figure 3.10 displays a timeline sequence. The timeline is a user programmable feature that selects how long the user wants a particular protein crystal solution to be exposed to a desired temperature. The flight TES contained a data logging system which collected thermistor data at the specified sample rate in the sequence timeline.

SEQ	TIME	TEMP	RATE
00	05:30	030.0	10
01	02:15	100.0	30
02	01:00	300.0	05
03	00:00	000.0	00
04	00:00	000.0	00

Figure 3.10. Timeline Sequence Segment

The following is an explanation of the timeline segment (Figure 3.10). Once the user has started a timeline from the Program Menu, the program initializes to the #00 index. For a total of 5 hours and 30 minutes the TES system will maintain a temperature of 30 degrees Celsius. During this time interval, the system captures data from the thermistors (selected in the Set Thermistors Menu) and records 1 sample every 10 minutes. After 5 hours and 30 minutes, the system changes the internal temperature to 100 degrees Celsius and records the temperature data at a rate of 1 sample every 30 minutes. And after 2 hours and 15 minutes, the system changes the temperature to 300 degrees Celsius for one hour, recording 1 sample every 5 minutes.

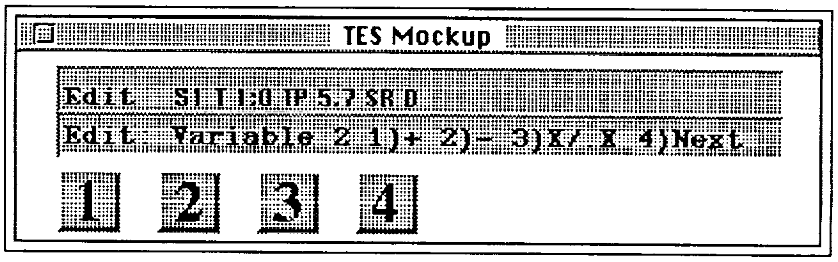
The following is a list of minimum and maximum values for each timeline parameter:

parameter	minimum	maximum
sequence number	1	10
time value	0 minutes	99 hours and 59 minutes (4 days 3 hours and 59 minutes)
temperature	0 degrees Celsius	999.9 degrees Celsius
sample rate	0 samples per minute	1 sample per 999 minutes (equivalent to 1 sample every 16 hours and 39 minutes)

Button #2 of the View Menu decrements the timeline program. Button #3 selects the Edit Menu option, while Button #4 returns the user Back to the Program Menu.

3.4.2 Edit Menu

The Edit Menu (Figure 3.11) allows the user to change the timeline variables. The first line of the LCD displays the current line of the timeline sequence. The second line displays the timeline variable to change. Button #1 increments the current variable, #2 decrements the current variable, #3 toggles the temperature factor (either 1 or .1), and #4 sequences the variables. Once in the Edit Menu, time will be the initial variable displayed. Button #4 pressed once will change the variable to the temperature setting. Pressed once more, Button #4 will change the variable to the sample rate setting. If Button #4 is pressed once more, the user will be returned Back to the View Menu. When a variable is changed, its new value is displayed on the first line of the LCD.

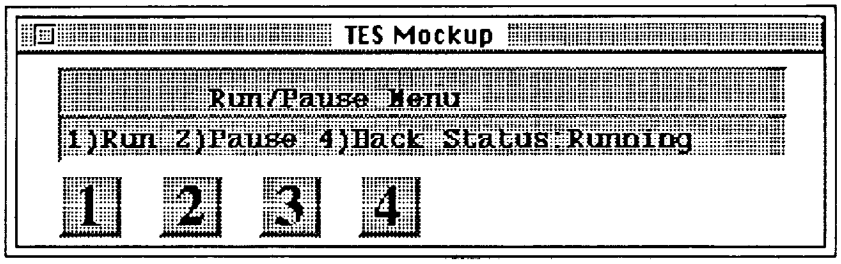


```
Edit  S1 T 1:0 IP 5.7 SR 0
Edit  Variable 2 1)+ 2)- 3)X/.X 4)Next
```

Figure 3.11. Edit Menu

3.4.3 Run/Pause Menu

The Run/Pause Menu (Figure 3.12) allows the user to execute or pause the timeline sequence. On the right side of the second line of the LCD, the system displays the current status of the system, that is, 'RUNNING' or 'PAUSED'. Button #1 starts the system running or continues from where the system was paused. Button #2 pauses the current timeline indefinitely until the Button #1 is pressed. When paused, execution of the timeline is suspended and the current temperature remains constant within the TES experiment cavity. Button #4 returns the user Back to the Program Menu.



Run/Pause Menu
1)Run 2)Pause 4)Back Status:Running

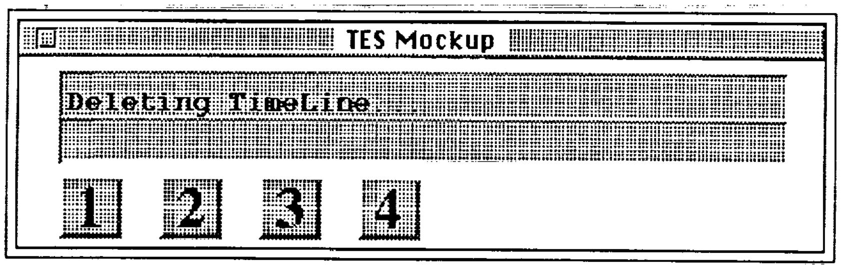
Figure 3.12. Run/Pause Menu

3.4.4 Delete Timeline

Button #3 of the Program Menu (Figure 3.8) allows the user to delete the current programmed timeline. Once pressed, the TES system displays 'Deleting Timeline...' on the first line of the LCD. The system then deletes the timeline leaving an array of zeroed parameters as follows:

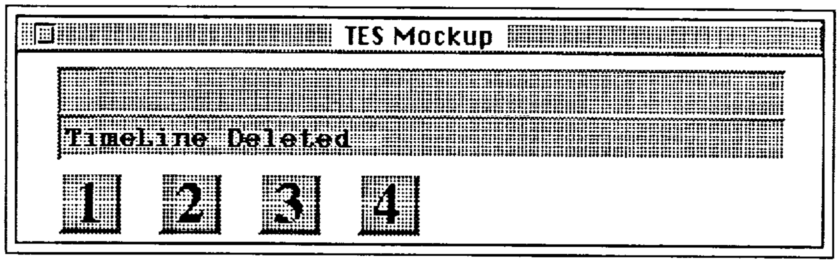
```
01  00:00  000.0  00
02  00:00  000.0  00
03  00:00  000.0  00
04  00:00  000.0  00
05  00:00  000.0  00
06  00:00  000.0  00
07  00:00  000.0  00
08  00:00  000.0  00
09  00:00  000.0  00
10  00:00  000.0  00
```

Once the system deletes the current timeline profile, the second line of the LCD will read 'Timeline is Deleted!' acknowledging that all variables have been initialized. The system then returns to the Program Menu. Figures 3.13 and 3.14 show the delete displays. Once deleted, new timeline programs must be entered via the Edit Menu.



Deleting TimeLine...

Figure 3.13. Delete Display #1

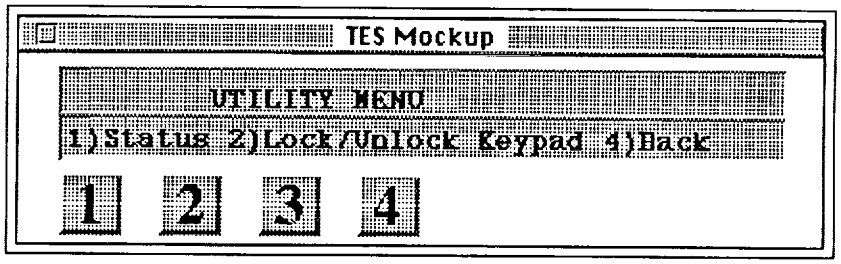


TimeLine Deleted

Figure 3.14. Delete Display #2

3.5 Utility Menu

The Utility Menu (Figure 3.15) allows the user to view system status and lock/unlock the keypad. Button #1 displays the status of the system. Button #2 locks or unlocks the keypad. Button #4 returns the operator to the Main Menu.

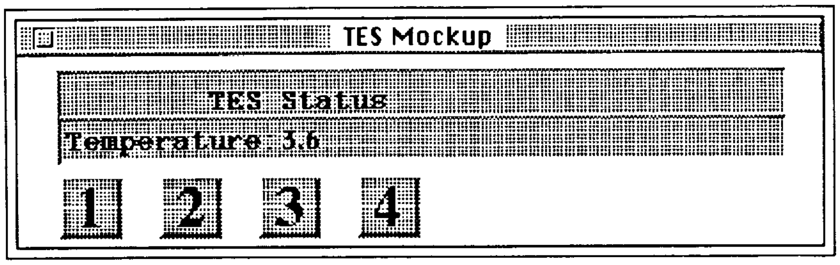


UTILITY MENU
1)Status 2)Lock/Unlock Keypad 4)Back

Figure 3.15. Utility Menu

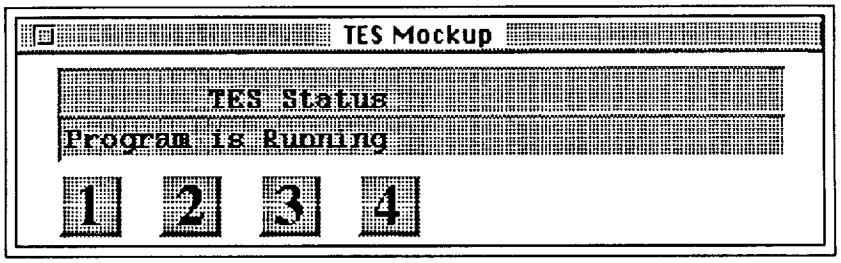
3.5.1 System Status

When Button #1 of the Utility Menu (Figure 3.15) is pressed, the system displays a series of system checks to the LCD. Each check is displayed for a few seconds and then cleared from the display. Four system status checks are displayed which are the current temperature of the TES, the current mode of execution, communication status and a character display check (Figures 3.16 - 3.19). After the final status screen is displayed, the system returns to the Utility Menu.



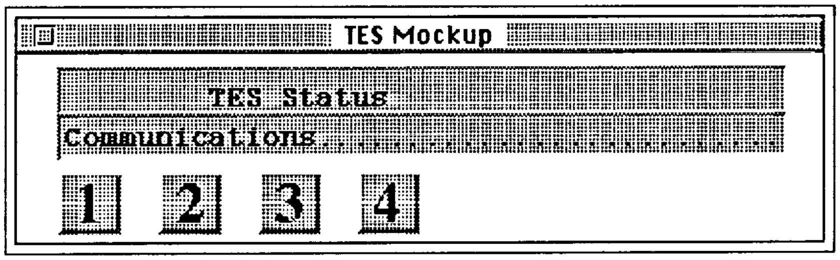
TES Status
Temperature 3.6

Figure 3.16. Current Temperature Status Display



TES Status
Program is Running

Figure 3.17. Current Execution Status



TES Status
Communications...........................

Figure 3.18. Communication Status

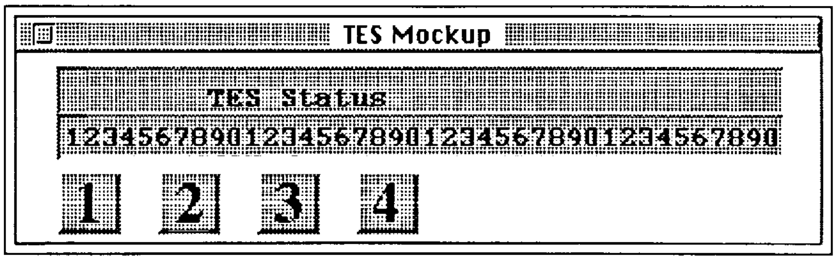


Figure 3.19. Character Display Status

3.5.2 Lock/Unlock Keypad

Button #2 of the Utility Menu (Figure 3.15) allows the user to lock or unlock the keypad. Button #2 depressed once locks the menu and twice unlocks the menu. Once the keypad has been locked, the mockup can be transported without altering the menu contents, preventing unintentional keypad depressions. The LCD will display the Utility Menu indefinitely until Button #2 is pressed again to unlock the keypad.

4. Serial Communication

The TES mockup can be operated remotely through the RS232 serial communications port. The remote functions of the TES operates similar to the functions of the four keypad buttons located on the front of the TES mockup. The flight TES was capable of accepting nine commands or transmitting data via the RS422 communications port. One of the original commands, 'LOAD', allowed a prepared timeline to be downloaded serially. The mockup doesn't contain this feature in its current version. Unlike the flight TES, the mockup's serial protocol was designed to gain full access of all menus. A few additional features were added to the serial protocol to anticipate and correct for synchronization errors between the mockup and the ground station.

4.1 Basic Protocol

All features for serial communication operate similar to the four button switches located on the front of the Thermal Enclosure System. ASCII 1 - 4 correspond to Buttons #1 - #4. Instead of pressing Button #1, the user can send an ASCII '1' through the serial port to get the same response. The menu flow will operate to either input device.

4.2 Additional Features

Additional features have been added for remote control considerations. These features cover initialization, synchronization, and audio check feedback. To assist in these additional features, a command value has been assigned to each software function. In the case of initialization, the system repeatedly sends the command value of the TES Display Logo through the serial communications port. The ground station or remote site receives this command value whenever the TES software is in this ground state. The command value stops transmitting whenever an input value is received either by one of the four Buttons on the TES front face or an ASCII equivalent value via the serial port.

A synchronization feature was added to prevent asynchronous program flows between the TES mockup and the remote ground station. Whenever a menu is selected, the value of the current menu command value is sent to the communications port. For example, if the TES mockup has been operating via the remote station and a break appears in the communications line, the remote station can restart without affecting the current TES software state. The remote station can then request the current command value of the mockup. Once received, the ground station can continue operating the mockup remotely from the current TES software state. The remote site must send an ASCII '5' to request the current command value.

To ensure a proper communications link, an audio check feature has been added to the Utility Menu (Figure 3.5). If the user sends an ASCII '3' to the TES system, three beeps will sound. This feature was added to provide feedback to the remote site. The audible sounds indicate that the remote signal was received.

5. Hardware Diagrams and Connections

This section describes the hardware used in the mockup and the proper wiring configurations for the momentary switches.

5.1 DP-31/535 Single Board Computer

The DP-31/535 single board computer was used to implement the TES simulation software. The board measures 4.125 by 6.5 inches and permits up to 120K Bytes of EPROM and/or RAM. The microcontroller used is the Siemens 80535. The 80535 provides 256 Bytes of RAM, 8 channel 8 bit A/D Converter, parallel and buffered serial ports.

5.1.1 Block Diagram

Figure 5.1 provides a block diagram of the DP-31/535.

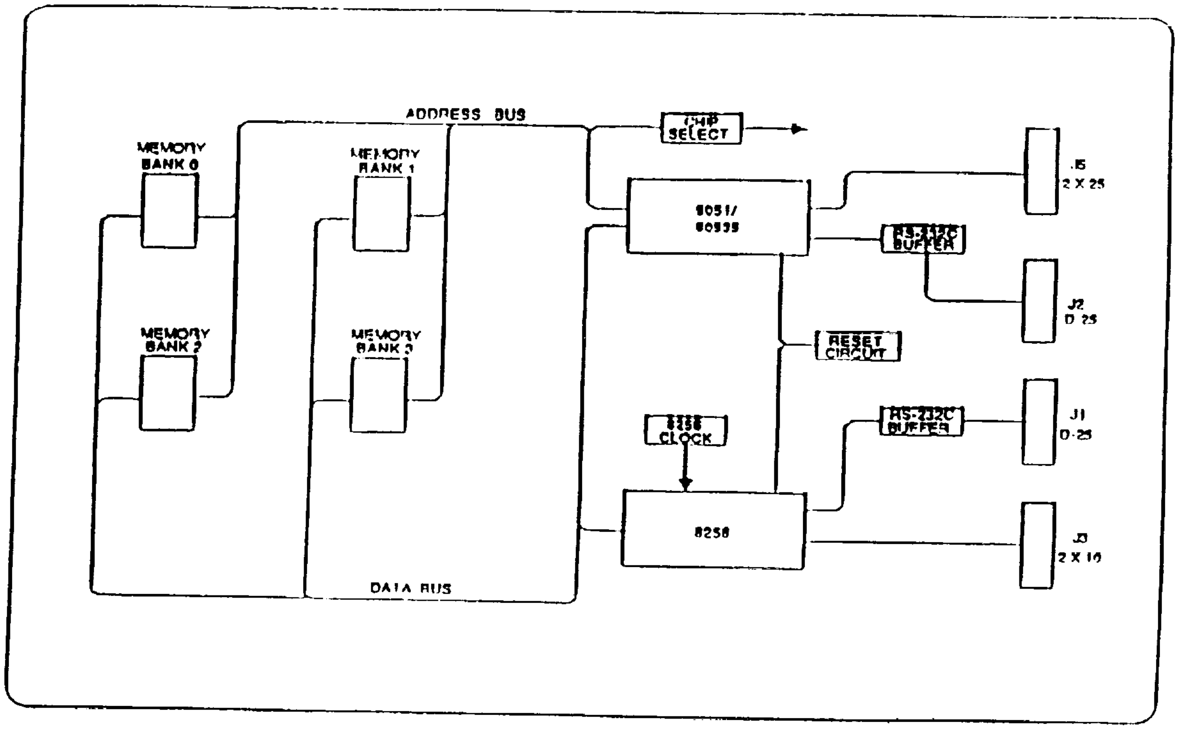


Figure 5.1. DP-31/535 Block Diagram

5.1.2 J5 Connections

Figure 5.2 shows the 50 pin jumper (J5) on the DP-31/535 board and the TES mockup configuration. Pins 15 and 16 are used for the speaker. Pins 25 - 34 and pin 36 are used for the LCD. Pins 37 - 40 are used for the four momentary switches on the front face of the TES mockup.

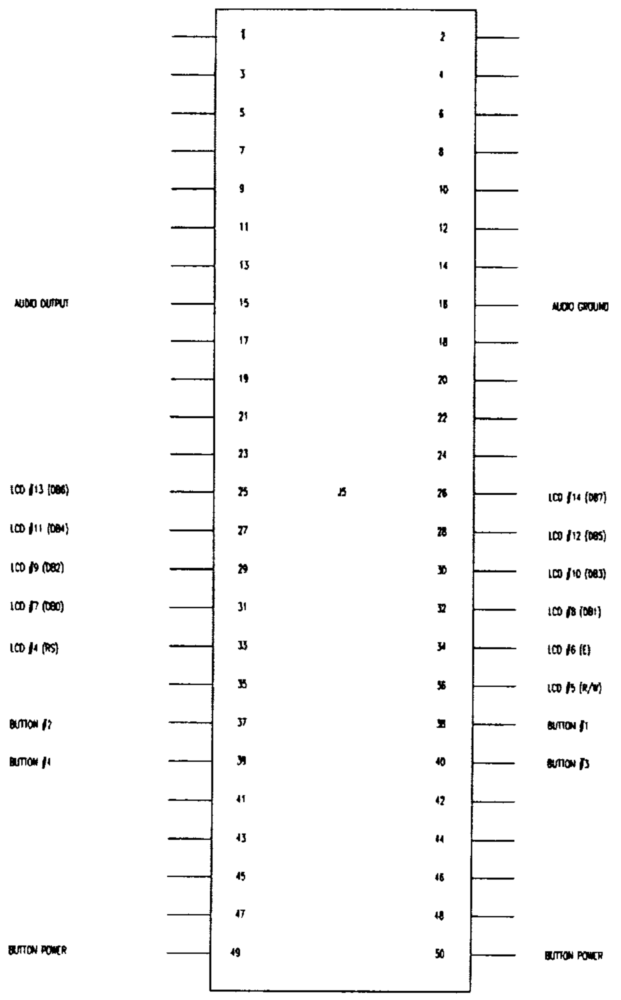


Figure 5.2. J5 Connections

5.1.3 Momentary Switches

For clarity, the momentary switches are the same as the buttons. All four buttons or momentary switches make up the keypad. Each momentary switch is connected normally open (NO). When a button is pressed, a +5 volt signal is sent to the DP-31\535 microcontroller. Figure 5.3 displays the current wiring configuration used in the TES mockup. Button #1 is connected to pin 38 of the J5 connector. Button #2 is connected to pin 37. Button #3 is connected to pin 40. And Button #4 is connected to pin 39 of the J5 connector. The +5 volt signal is attached to each button as default. The one +5 volt signal is connected to either pin 49 or pin 50 of the J5 connector.

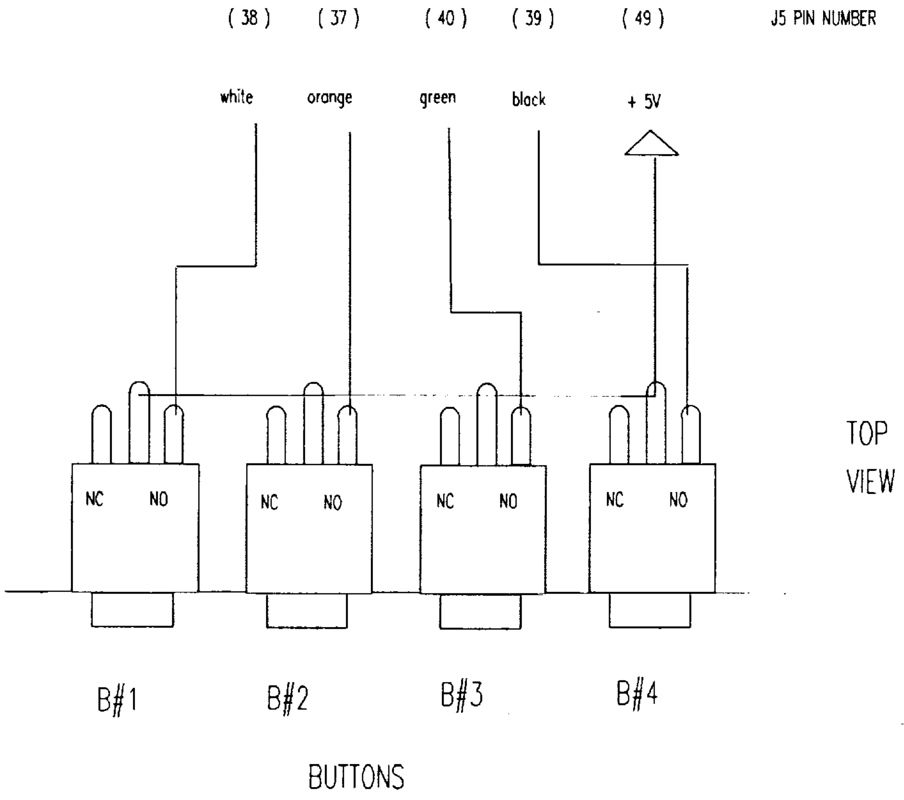


Figure 5.3. J5 Momentary Switches (Buttons)

5.1.4 Power Requirements

The DP-31/535 system board requires four connections through the J4 connector. Pins 1 - 4 should be supplied by -12 volts, +12 volts, GND, and +5 volts, respectively. VAGND, VAREF, and VPD connections are available and are optional. Refer to the DP-31/535 Single Board Computer User's Manual for further information on signal connections.

5.2 LCD Module

The AND591 LCD module was selected for the TES mockup display output. This compact dot matrix LCD provides a 2 line x 40 characters panel that can display 160 different kinds of alphabets, numbers, and symbols. The AND591 has built-in control with display RAM and character generator ROM. It also has the option of being interfaced to either a 4 or 8 bit CPU. The dimensions of the LCD module are as follows:

width	182 mm
height	33.5 mm
depth	13 mm.

The following is a list of LCD pin connections:

LCD pin #		Pin Connection	
1	(GND)	3	(J4)
2	(VDD)	4	(J4)
3	(V0)	pot	(25k ohm)
4	(RS)	33	(J5)
5	(R/W)	36	(J5)
6	(E)	34	(J5)
7	(DB0)	31	(J5)
8	(DB1)	32	(J5)
9	(DB2)	29	(J5)
10	(DB3)	30	(J5)
11	(DB4)	27	(J5)
12	(DB5)	28	(J5)
13	(DB6)	25	(J5)
14	(DB7)	26	(J5)

6. Description of Appendixes

Appendix A contains references related to the flight thermal enclosure system and hardware used in the simulated mockup. Appendix B contains a list of the simulation software.

7. Concluding Remarks

This paper describes the operation of the simulated thermal enclosure system used in the Intravehicular Automation and Robotics laboratory. The mockup has been used in tests with an internal telerobotic servicing system to investigate time required and errors encountered in performing a representative microgravity task.

Appendix A Related Documents

This section describes the reference manuals used to obtain information on the flight Thermal Enclosure System control specifications, DP-31/535 Single Board Computer, the 80535 microprocessor, and the AND591 LCD module.

- o Thermal Enclosure System User's Manual
  Space Industries, Inc.
  Webster, Texas
  (713) 338-2676
  September 1991

- o DP-31/535 Single Board Computer User's Manual
  Allen Systems
  2346 Brandon Road
  Columbus, Ohio 43221
  (614) 488-7122
  November 1, 1986

- o Siemens 8-Bit Single Chip Microcontroller Handbook
  Siemens Components, Inc.
  2191 Laurelwood Road
  Santa Clara, CA 95054
  (408) 980-4500

- o The AND591 LCD
  AND
  770 Airport Boulevard
  Burlingame, California 94010
  (415) 347-9916

Appendix B Software Listing

The following software listing was written in the C programming language. Microprocessor specifics are described in the Siemens 8-Bit Single Chip Microcontroller Handbook.

```
/* TES1.C - code to operate the Thermal Enclosure System (TES).
 *      This code controls 2 input sources (4 LCD buttons & serial port)
 *      and 2 output devices (an LCD display & serial port).
 *                                  July 1993 - ATM */

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <reg515.h>

void   initialize();
void   display_on();
void   display_off();
void   cursor_on();
void   cursor_off();
void   clear_LCD();
void   position_cursor();
void   position( int line, int pos );
void   line1();
void   line2();
void   send_char( char letter );
void   send_line( char *string );
void   clear_line1();
void   clear_line2();
void   pause();
void   wait();
void   debounce( int num );
int    buttons( int menu );
int    buttons_wait( int menu );
void   send_menu( int menu );
void   shift_display_left();
void   shift_display_right();
void   float_to_string( float num, char *fstring );
void   num_to_string( int num, int digits, char *nstring );
void   time_to_string( int num, char *tstring );
void   speaker_sound();

/* global variable for serial comm */
int    external_input = 0;

main()
{
   int loop,get_key,value,line,pos,cur_address,ii,char_hex,check;
   int button,command,num,thermistor[9],SR[10],SEQ[10],TIME[10];
   int tX,tX_OLD,tnum,a,b,var,mode,lock;
```

```
float temperature[10];
char string[40],fstring[4],nstring[2],status_string[3],BF[6],letter;
char sstring[3],tstring[5],mstring[7];


SCON = 0x52;       /* setup serial port control          */
PCON &= 0x7F;      /* clear SMOD                         */
BD = 1;            /* use internal baud rate generator   */
                   /*  based on: baud_rate = 2^SMOD/2500 *
                                                            clock_frequency
                                                           = 2^0/2500 * 12e6
                                                           = 4800        */
ES = 1;            /* enable serial interrupt           */
EAL = 1;           /* enable all interrupts             */

loop = 1;          /* initialize while ( loop = 1 )      */
strcpy( string,"                              " );


initialize();      /* initialize LCD              */
command = 1;
get_key = 0;

/* TES variables */
for( tnum=1;tnum<=10;tnum=tnum+1 )
     thermistor[tnum] = 0;
tnum = 1;
for( b=0;b<=11;b=b+1 )
{
     SEQ[b] = b;            /* sequence #  */
     TIME[b] = 0;           /* minutes     */
     temperature[b] = 0.0;  /* temperature */
     SR[b] = 0;             /* samples/sec */
}

SEQ[1] = 1;
TIME[1] = 90;
temperature[1] = 123.4;
SR[1] = 10;
b = 1;
tX = 1;
tX_OLD = 1;
var = 1;   /* edit time variable */
lock = 0;
```

```
while( loop )
{

        switch( command )
        {
                case 1:          /*******  TES INTRO   *********/
                  cursor_off();
                  line1();
                  get_key = 0;
                  strcpy( string,"    Thermal Enclosure System (TES)     " );
                  send_line( string );
                  line2();
                  strcpy( string,"             NASA LaRC              " );
                  send_line( string );

                  /*debounce( 5 );*/
                  wait();
                  P5 =0x00;

                  while( get_key == 0 ) /* shift left and look for key input */
                  {
                    shift_display_left();
                    for( ii=0; ii<3; ii++ ) /* approx 2.5 sec wait loop */
                    {
                            get_key = buttons( command );
                            if( get_key == 5 ) get_key = 0;
                            if( get_key != 0 ) ii=100;
                            wait();
                    }
                  }

                  command = 2;

                  /*debounce( 1 );*/

                  break;

                case 2:          /*******  MAIN MENU   *********/
                  clear_LCD();
                  line1();
                  strcpy( string,"             MAIN MENU              " );
                  send_line( string );
                  line2();
```

```
  strcpy( string,"1) Set  2) Program  3) Utility  4) Logo " );
  send_line( string );
  get_key = 0;

  /*debounce( 1 );*/
  wait();
  get_key = buttons_wait( command );

  if( get_key==1 )      command = 3;  /* Set Menu     */
  if( get_key==2 )      command = 7;  /* Program Menu */
  if( get_key==3 )      command = 11; /* Utility Menu */
  if( get_key==4 )      command = 1;  /* Logo         */
  if( get_key==5 )      command = 2;

  /*debounce( 1 );*/
  wait();
  break;

case 3:            /*******  SET MENU  *********/
  clear_LCD();
  line1();
  strcpy( string,"            SET MENU            " );
  send_line( string );
  line2();
  strcpy( string,"1) Temp 2) Therm 3) Sample Rate 4) Main " );
  send_line( string );
  get_key = 0;

  get_key = buttons_wait( command );

  if( get_key==1 )      command = 4;  /* Set Temp   */
  if( get_key==2 )      command = 5;  /* Set Therms */
  if( get_key==3 )      command = 6;  /* Set SR     */
  if( get_key==4 )      command = 2;  /* Main Menu  */
  if( get_key==5 )      command = 3;

  /*debounce( 1 );*/
  wait();
  break;

case 4:            /***  Set Temperature MENU  ***/
  strcpy( fstring,"     " );
  clear_LCD();
  line1();
  strcpy( string,"          Set Temperature          " );
```

```
send_line( string );
line2();
strcpy( string,"1) + 2) - 3) X/.X 4) Back   TEMP:       " );
send_line( string );
position( 2, 34 );
float_to_string( temperature[b], fstring );
send_line( fstring );
strcpy( fstring,"      " );
get_key = 0;

get_key = buttons_wait( command );

if( get_key==1 )    /* Increment Temperature   */
{
  if( tX == 1 )
          temperature[b] = temperature[b] + 1.0;
  if( tX == 0 )
          temperature[b] = temperature[b] + 0.1;
  if( temperature[b] > 999.9 )
          temperature[b] = 999.9;
  command = 4;
}
if( get_key==2 )    /* Decrement Temperature   */
{
  if( tX == 1 )
          temperature[b] = temperature[b] - 1.0;
  if( tX == 0 )
          temperature[b] = temperature[b] - 0.1;
  if( temperature[b] < 0.0 )
          temperature[b] = 0.0;
  command = 4;
}
if( get_key==3 )    /* Select 1's or .1's  */
{
  tX_OLD = tX;
  if( tX_OLD == 1 )
          tX = 0;
  if( tX_OLD != 1 )
          tX = 1;
  command = 4;
  /*for(ii=0;ii<3;ii=ii+1)*/
          wait();
}
if( get_key==4 )
{
        command = 3;  /* Set Menu  */
        debounce( 1 );
      }
      if( get_key==5 )   command = 4;

      /*wait();*/

      break;

    case 5:          /***  Set Thermistors MENU  ***/
      strcpy( nstring,"  " );
      clear_LCD();
      line1();
      strcpy( string," Select Thermistors:  #    Status:    " );
      send_line( string );
      line2();
      strcpy( string,"1) Thermistor #   2) ON/OFF 4) Back    " );
      send_line( string );

      position( 1, 23 );
      num_to_string( tnum, 2, nstring );
      send_line( nstring );
      /*printf("\ntnum = %d   tnum nstring = %s\n",tnum,nstring);*/
      strcpy( nstring,"  " );

      position( 1, 36 );
      if( thermistor[tnum] == 1 )
        strcpy( status_string,"ON " );
       else
        strcpy( status_string,"OFF" );
      /*printf("thermistor[%d] = %d    status =
%s\n\n",tnum,thermistor[tnum],status_string);*/
      send_line( status_string );
      strcpy( status_string,"   " );


      get_key = 0;

      get_key = buttons_wait( command );

      if( get_key==1 )    /* Select Next Thermistor   */
      {
        tnum = tnum + 1;
        if( tnum > 10 )
              tnum = 1;
```

```
                    /*printf("get_key=1:  new tnum = %d\n",tnum);*/
                    command = 5;
                    /*wait();*/
                 }
                 if( get_key==2 )     /* Turn Thermistor ON/OFF    */
                 {
                    a = thermistor[tnum];
                    /*printf("get_key=2:  a = %d   old thermistor[%d] =
%d\n",a,tnum,thermistor[tnum]);*/
                    if( a == 1 )
                            thermistor[tnum] = 0;
                    if( a == 0 )
                            thermistor[tnum] = 1;
                    /*printf("get_key=2:  a = %d   new thermistor[%d] =
%d\n",a,tnum,thermistor[tnum]);*/
                    command = 5;
                    /*for(ii=0;ii<3;ii=ii+1)*/
                            wait();
                 }
                 if( get_key==4 )
                 {
                    command = 3;  /* Set Menu  */
                    debounce( 1 );
                 }
                 if( get_key==5 ) command = 5;

                 /*wait();*/
                 break;

               case 6:              /***   Set Sample Rate MENU   ***/
                 /*strcpy( sstring,"   " );*/
                 clear_LCD();
                 line1();
                 strcpy( string,"            Set Sample Rate            " );
                 send_line( string );
                 line2();
                 strcpy( string,"1) + 2) -  4) Back   SAMPLE RATE:        " );
                 send_line( string );
                 position( 2, 34 );
                 num_to_string( SR[b], 3, nstring );
                 send_line( nstring );
                 /*strcpy( sstring,"   " );*/
                 get_key = 0;

                 get_key = buttons_wait( command );
```

```
  if( get_key==1 )    /* Increment SR   */
  {
    SR[b] = SR[b] + 1;

    if( SR[b] > 999 )
            SR[b] = 999;
    command = 6;
  }
  if( get_key==2 )    /* Decrement SR   */
  {
    SR[b] = SR[b] - 1;

    if( SR[b] < 0 )
            SR[b] = 0;
    command = 6;
  }
  if( get_key==4 )
  {
    command = 3;  /* Set Menu  */
    debounce( 1 );
  }
  if( get_key==5 ) command = 6;

  /*wait();*/

  break;

case 7:           /*******  PROGRAM MENU   *********/
  clear_LCD();
  line1();
  strcpy( string,"          PROGRAM MENU            " );
  send_line( string );
  line2();
  strcpy( string,"1) View 2) Run/Pause 3) Delete  4) Main " );
  send_line( string );
  get_key = 0;

  get_key = buttons_wait( command );

  if( get_key==1 )     command = 8;   /* View Menu  */
  if( get_key==2 )     command = 10;  /* Run/Pause  */
  if( get_key==3 )     command = 13;  /* Delete     */
  if( get_key==4 )     command = 2;   /* Main Menu  */
  if( get_key==5 )     command = 7;
  /*debounce( 1 );*/
  wait();
  break;

case 8:          /***   VIEW MENU   ***/
  strcpy( nstring,"  " );
  strcpy( fstring,"     " );
  strcpy( tstring,"     " );
  clear_LCD();
  line1();
  strcpy( string,"VIEW: S   T      TEMP       SR     " );
  send_line( string );
  line2();
  strcpy( string,"1) + 2) - 3) Edit 4) Back              " );
  send_line( string );

  position( 1, 7 );
  num_to_string( SEQ[b], 2, nstring );
  send_line( nstring );
  strcpy( nstring,"  " );

  position( 1, 13 );
  time_to_string( TIME[b], tstring );
  send_line( tstring );
  strcpy( tstring,"     " );

  position( 1, 25 );
  float_to_string( temperature[b], fstring );
  send_line( fstring );
  strcpy( fstring,"      " );

  position( 1, 35 );
  num_to_string( SR[b], 2, nstring );
  send_line( nstring );
  strcpy( nstring,"  " );


  get_key = 0;

  get_key = buttons_wait( command );

  if( get_key==1 )    /* Select Next Timeline Sequence */
  {
    b = b + 1;
    if( b > 10 )
            b = 10;
    command = 8;
  }
  if( get_key==2 )     /* Select Previous Timeline Sequence */
  {
    b = b - 1;
    if( b < 1 )
            b = 1;
    command = 8;
  }
  if( get_key==3 )
  {
    command = 9;  /* Edit Menu  */
  }
  if( get_key==4 )
  {
    command = 7;  /* Program Menu  */
  }
  if( get_key==5 )    command = 8;

  /*wait();*/
  break;

case 9:            /***  EDIT MENU  ***/
  strcpy( nstring,"  " );
  strcpy( fstring,"     " );
  strcpy( tstring,"    " );
  clear_LCD();
  line1();
  strcpy( string,"EDIT: S   T      TEMP      SR     " );
  send_line( string );
  line2();
  strcpy( string,"EDIT:       1) + 2) - 3) X/.X 4) Next" );
  send_line( string );

  position( 1, 7 );
  num_to_string( SEQ[b], 2, nstring );
  send_line( nstring );
  strcpy( nstring,"  " );

  position( 1, 13 );
  time_to_string( TIME[b], tstring );
  send_line( tstring );
  strcpy( tstring,"    " );
```

```
position( 1, 25 );
float_to_string( temperature[b], fstring );
send_line( fstring );
strcpy( fstring,"     " );

position( 1, 35 );
num_to_string( SR[b], 2, nstring );
send_line( nstring );
strcpy( nstring,"  " );

if( var == 1 )  /* edit time */
{
  position( 2, 8 );
  strcpy( tstring,"TIME" );
  send_line( tstring );
  strcpy( tstring,"    " );
}

if( var == 2 )  /* edit temp */
{
  position( 2, 8 );
  strcpy( tstring,"TEMP" );
  send_line( tstring );
  strcpy( tstring,"    " );
}

if( var == 3 )  /* edit sample rate */
{
  position( 2, 8 );
  strcpy( tstring,"RATE" );
  send_line( tstring );
  strcpy( tstring,"    " );
}

get_key = 0;

get_key = buttons_wait( command );
/*printf("\nEDIT: var = %d  get_key = %d\n",var,get_key);*/

if( get_key==1 )    /* Increment variable */
{
  if( var == 1 )  /* incr time */
  {
        TIME[b] = TIME[b] + 1;
```

```
            if( TIME[b] > 5999 )
                    TIME[b] = 5999;
        }

        if( var == 2 )  /* incr temp */
        {
            if( tX == 1 )
                    temperature[b] = temperature[b] + 1.0;
            if( tX == 0 )
                    temperature[b] = temperature[b] + 0.1;
            if( temperature[b] > 999.9 )
                    temperature[b] = 999.9;
        }

        if( var == 3 )  /* incr sr */
        {
            SR[b] = SR[b] + 1;
            if( SR[b] > 999 )
                    SR[b] = 999;
        }

        command = 9;
    }
    if( get_key==2 )      /* Decrement variable */
    {
        if( var == 1 )  /* decr time */
        {
            TIME[b] = TIME[b] - 1;
            if( TIME[b] < 0 )
                    TIME[b] = 0;
        }

        if( var == 2 )  /* decr temp */
        {
            if( tX == 1 )
                    temperature[b] = temperature[b] - 1.0;
            if( tX == 0 )
                    temperature[b] = temperature[b] - 0.1;
            if( temperature[b] < 0.0 )
                    temperature[b] = 0.0;
        }

        if( var == 3 )  /* decr sr */
        {
            SR[b] = SR[b] - 1;
            if( SR[b] < 0 )
                    SR[b] = 0;
        }

        command = 9;
    }
    if( get_key==3 )    /* Select 1's or .1's  */
    {
        tX_OLD = tX;
        if( tX_OLD == 1 )
                tX = 0;
        if( tX_OLD != 1 )
                tX = 1;
        command = 9;
        /*for(ii=0;ii<3;ii=ii+1)*/
                wait();
    }
    if( get_key==4 )
    {
        /*printf("EDIT: get_key = 4 (before incr) var = %d\n",var);*/
        var = var + 1;
        /*printf("EDIT: get_key = 4 (after  incr) var = %d\n",var);    */
        command = 9;
        if( var == 4 )
        {
                var = 1;
                command = 8;  /* View Menu  */
        }
        debounce( 1 );
    }
    if( get_key==5 )  command = 9;

    /*wait();*/

    break;

  case 10:            /*******   Run/Pause MENU   *********/
    strcpy( mstring,"      " );
    clear_LCD();
    line1();
    strcpy( string,"          Run/Pause Menu            " );
    send_line( string );
    line2();
    strcpy( string,"1) Run 2) Pause 4) Back  STATUS:       " );
    send_line( string );
  get_key = 0;
  position( 2, 33 );
  if( mode == 1 )
    strcpy( mstring,"RUNNING" );
   else
    strcpy( mstring,"PAUSED " );
  send_line( mstring );

  get_key = buttons_wait( command );

  if( get_key==1 )     /* Select Run Mode  */
  {
    mode = 1;
    command = 10;
    /*for(ii=0;ii<3;ii=ii+1)*/
            wait();
  }
  if( get_key==2 )     /* Select Pause Mode  */
  {
    mode = 0;
    command = 10;
    /*for(ii=0;ii<3;ii=ii+1)*/
            wait();
  }
  if( get_key==4 )
  {
    command = 7;
    debounce( 1 );
  }
  if( get_key==5 )  command = 10;

  /*debounce( 1 );*/
  /*wait();*/
  break;

case 11:          /*******   UTILITY MENU   *********/
  clear_LCD();
  line1();
  strcpy( string,"        UTILITY MENU            " );
  send_line( string );
  line2();
  strcpy( string,"1) Status 2) Lock/Unlock Keypad 4) Main " );
  send_line( string );
  get_key = 0;
```

```
   get_key = buttons_wait( command );

   if( get_key==1 )      command = 12;  /* Status       */
   if( get_key==2 )       /* Lock/Unlock  */
   {
     a = lock;
     if(a==1)
            lock=0;
     if(a==0)
            lock=1;

     while( lock == 1 )
     {
            debounce( 1 );
            get_key = buttons_wait( command );
            if( get_key==2 )
                   lock=0;
     }

     command = 11;
   }
   if( get_key==3 )
   {
     speaker_sound();
     command = 11;
   }
   if( get_key==4 )      command = 2;   /* Main Menu     */
   if( get_key==5 )      command = 11;

   /*debounce( 1 );*/
   wait();
   break;

 case 12:          /*******  Status Displays  *********/
   strcpy( fstring,"      " );
   strcpy( mstring,"       " );
   clear_LCD();
   line1();
   strcpy( string,"          TES Status              " );
   send_line( string );
   line2();
   strcpy( string,"     Temperature        TEMP:       " );
   send_line( string );
   position( 2, 34 );
   float_to_string( temperature[b], fstring );
```

```
  send_line( fstring );
  strcpy( fstring,"     " );
  for(ii=0;ii<3;ii=ii+1)
    wait();

  line2();
  strcpy( string," ............Program Timeline is       " );
  send_line( string );
  get_key = 0;
  position( 2, 33 );
  if( mode == 1 )
    strcpy( mstring,"RUNNING" );
   else
    strcpy( mstring,"PAUSED " );
  send_line( mstring );
  for(ii=0;ii<3;ii=ii+1)
    wait();

  line2();
  strcpy( string," Communications........................ " );
  send_line( string );
  for(ii=0;ii<3;ii=ii+1)
    wait();

  line2();
  strcpy( string,"1234567890123456789012345678901234567890" );
  send_line( string );
  for(ii=0;ii<3;ii=ii+1)
    wait();

  command = 11;       /* Utility Menu */
  break;

case 13:            /*******  Delete Timeline  *********/
  clear_LCD();
  line1();
  strcpy( string,"          Deleting Timeline.....     " );
  send_line( string );
  for(ii=0;ii<3;ii=ii+1)
    wait();

  for( b=0;b<=11;b=b+1 )
  {
    SEQ[b] = b;          /* sequence #  */
    TIME[b] = 0;          /* minutes     */
```

```
            temperature[b] = 0.0;  /* temperature */
            SR[b] = 0;             /* samples/sec */
          }
          b=1;
          SEQ[1] = 1;
          temperature[1] = 0.0;
          tX = 1;
          tX_OLD = 1;
          var = 1;   /* edit time variable */

          line2();
          strcpy( string,"         Timeline is Deleted!         " );
          send_line( string );
          for(ii=0;ii<3;ii=ii+1)
            wait();

          command = 7;        /* Program Menu */
          break;

        default:
          clear_LCD();
          line1();
          strcpy( string,"  This functions not yet implemented!  " );
          send_line( string );
          line2();
          strcpy( string," Press a key to return to the Main Menu " );
          send_line( string );
          get_key = 0;

          get_key = buttons_wait( command );

          command = 2;  /* Main Menu    */

          /*debounce( 2 );*/
          wait();
          break;


      }  /* end of switch statement */

  }     /* end of while loop */

  }     /* end of main */
```

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

void    pause( void )
{
        char    ret;

        printf("Hit Return to Step!\n");
        scanf("%c",&ret);
}

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

void    wait( void )
{
        int     n,nmax;

        n = 0;
        nmax = 1000;

        while( n<nmax ){ n++; }
}

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

void    debounce( int num )
{
        int     ii,iimax;

        /* debounce wait function */

        /*iimax = num * 20;*/
        iimax = 0;

        for( ii=0; ii<iimax; ii++ )
        {
                wait();
        }

}

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

void    initialize( void )
{
        P4 = 0x38;      /* function set = 2 lines,5X7 dots, 8 bit */
        P5 = 0x01;     /* enable = 1          */
        P5 = 0x00;     /* enable = 0 (clocked) */
        wait();
        P4 = 0x0E;     /* turn display on, cursor on   */
        P5 = 0x01;     /* enable = 1          */
        P5 = 0x00;     /* enable = 0 (clocked) */
        /*printf("\n<<< LCD initialized! >>>\n");*/
}

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

void   display_on( void )
{
        P4 = 0x0E;     /* turn display on, cursor on   */
        P5 = 0x01;     /* enable = 1          */
        P5 = 0x00;     /* enable = 0 (clocked) */
}

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

void   display_off( void )
{
        P4 = 0x08;     /* turn display off, cursor off */
        P5 = 0x01;     /* enable = 1          */
        P5 = 0x00;     /* enable = 0 (clocked) */
}

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

void   cursor_on( void )
{
        P4 = 0x0E;     /* turn display on, cursor on   */
        P5 = 0x01;     /* enable = 1          */
        P5 = 0x00;     /* enable = 0 (clocked) */
}

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

void   cursor_off( void )
{
        P4 = 0x0C;     /* turn display on, cursor off  */
        P5 = 0x01;     /* enable = 1          */
        P5 = 0x00;     /* enable = 0 (clocked) */
}
```

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

void   clear_LCD( void )
{
        P4 = 0x01;
        P5 = 0x01;
        P5 = 0x00;
        /*printf("\n<<< LCD cleared! >>>\n");*/
}

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

void   position_cursor( void )
{
        int line,pos,cur_address;

        printf("\nLine < 1 > or Line < 2 >?  >> ");
        scanf("%d",&line);
        printf("\nCharacter position < 0 - 39 >?  >> ");
        scanf("%d",&pos);
        printf("\nLine = %d  Position = %d\n",line,pos);
        position( line, pos );
        /*cur_address = 0x80;
        if( line==1 )
        cur_address = cur_address + 0x00;
        else if( line==2 )
        cur_address = cur_address + 0x40;
                cur_address = cur_address + pos;
        /*printf("\nCursor address = %#x\n",cur_address);*/
        P4 = cur_address;
        P5 = 0x01;
        P5 = 0x00;
        /*printf("\n<<< Cursor positioned! >>>\n");*/
}

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

void   position( int line, int pos )
{
        int cur_address;

        cur_address = 0x80;
        if( line==1 )
                cur_address = cur_address + 0x00;
        else if( line==2 )
```

```
                cur_address = cur_address + 0x40;
        cur_address = cur_address + pos;
        P4 = cur_address;
        P5 = 0x01;
        P5 = 0x00;
}


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

void    line1( void )
{
        int cur_address;

        cur_address = 0x80;
        P4 = cur_address;
        P5 = 0x01;
        P5 = 0x00;
        /*printf("\n<<< Cursor positioned at line1! >>>\n");*/
}

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

void    line2( void )
{
        int cur_address;

        cur_address = 0x80 + 0x40;
        P4 = cur_address;
        P5 = 0x01;
        P5 = 0x00;
        /*printf("\n<<< Cursor positioned at line2! >>>\n");*/
}


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

void    send_char( char letter )
{
        /*wait();*/
        P4 = (int) letter;
        P5 = 0x03;
        P5 = 0x02;
        /*printf("P4 = %#x\n",P4);*/
        /*printf("\n<<< Character Sent! >>>\n");*/
```

```
}

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

void    send_line( char *string )
{
        int check,ii,iimax;

        iimax=39;
        for( ii=0;ii<=iimax;ii++ )
        {
                check = iscntrl( string[ii] );
                if( check == 0 )
                     send_char( string[ii] );
                else
                     ii = iimax+1;
                /*printf("\n\nCharacter = %c DEC value = %i  HEX value = %#x\n"
                     ,string[ii],string[ii],string[ii]);*/
        }
        strcpy( string,"                                        " );
}

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

void    clear_line1( void )
{
        int check,ii,iimax;
        char clrstring[40];

        line1();
        strcpy( clrstring,"                                        " );
        for( ii=0;ii<=39;ii++ )
        {
                send_char( clrstring[ii] );
        }
        strcpy( clrstring,"                                        " );
        line1();
}

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

void    clear_line2( void )
{
        int check,ii,iimax;
        char clrstring[40];
        line2();
        strcpy( clrstring,"                              " );
        for( ii=0;ii<=39;ii++ )
        {
                send_char( clrstring[ii] );
        }
        strcpy( clrstring,"                              " );
        line2();
}

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

int     buttons( int menu )
{
        extern int external_input;
        int button;
        char nstring[2];

        /* output of command menu # for external source */
        strcpy( nstring,"  " );
        num_to_string( menu, 2, nstring );
        putchar( nstring[0] );
        putchar( nstring[1] );

        button = 0;


        P5 = 0x00;

        if( (P5&0x10) > 0 )         button = 1;

        if( (P5&0x20) > 0 )         button = 2;

        if( (P5&0x40) > 0 )         button = 3;

        if( (P5&0x80) > 0 )         button = 4;

        if( (button<1)||(button>4) )   button = 0;


        if( external_input == 1 )      button = 1;

        if( external_input == 2 )      button = 2;
        if( external_input == 3 )       button = 3;

        if( external_input == 4 )       button = 4;

        if( external_input == 5 )       button = 5;

        if( (button<1)||(button>5) )    button = 0;

        external_input = 0;

        return( button );
}

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

int     buttons_wait( int menu )
{
        extern int external_input;
        int hit,button;
        char nstring[2];

        /* output of command menu # for external source */
        strcpy( nstring,"  " );
        num_to_string( menu, 2, nstring );
        putchar( nstring[0] );
        putchar( nstring[1] );

        hit = 1;
        while( hit )
        {
                /* get external input, i.e. macintosh, pc, etc. */

                if( external_input == 1 )
                {
                    button = 1;
                    hit = 0;
                }
                if( external_input == 2 )
                {
                    button = 2;
                    hit = 0;
                }
                if( external_input == 3 )
                {
                    button = 3;
                hit = 0;
            }
            if( external_input == 4 )
            {
                button = 4;
                hit = 0;
            }
            if( external_input == 5 )
            {
                button = 5;
                hit = 0;
            }


            /* get input from LCD momentary switches */
            P5 = 0x00;
            if( (P5&0x10) > 0 )
            {
                button = 1;
                hit = 0;
                putchar( 0x30 + button );
            }
            if( (P5&0x20) > 0 )
            {
                button = 2;
                hit = 0;
                putchar( 0x30 + button );
            }
            if( (P5&0x40) > 0 )
            {
                button = 3;
                hit = 0;
                putchar( 0x30 + button );
            }
            if( (P5&0x80) > 0 )
            {
                button = 4;
                hit = 0;
                putchar( 0x30 + button );
            }
        }
        /*printf("\n<<< LCD button #%d detected! >>>\n",button);*/
        external_input = 0;
        return( button );
}
```

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

void    send_menu( int menu )
{
        extern int external_input;
        char nstring[2];

        /* output of command menu # for external source */

        strcpy( nstring,"  " );
        num_to_string( menu, 2, nstring );
        putchar( nstring[0] );
        putchar( nstring[1] );
}


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

void    shift_display_left( void )
{
        P4 = 0x18;      /* shift display 1 position to the left */
        P5 = 0x01;      /* enable = 1           */
        P5 = 0x00;      /* enable = 0 (clocked) */
}

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

void    shift_display_right( void )
{
        P4 = 0x1C;      /* shift display 1 position to the right */
        P5 = 0x01;      /* enable = 1           */
        P5 = 0x00;      /* enable = 0 (clocked) */
}

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

void    float_to_string( float num, char *fstring )
{
        int value,hvalue,tvalue,ovalue,dvalue;
        float temp,temp1,temp2,temp3;

        /* to get the hundred digit value */
        temp = num/100.0;
        hvalue = temp;
        fstring[0] = (char) (0x30 + hvalue);

        /* to get the ten digit value */
        temp1 = num - hvalue*100.0;
        tvalue = temp1/10.0;
        fstring[1] = (char) (0x30 + tvalue);

        /* to get the one digit value */
        temp2 = temp1 - tvalue*10.0;
        ovalue = temp2;
        fstring[2] = (char) (0x30 + ovalue);

        /* to get the decimal point */
        fstring[3] = 0x2E;

        /* to get the tenth digit value */
        temp3 = temp2 - ovalue;
        dvalue = temp3*10.0;
        fstring[4] = (char) (0x30 + dvalue);

        /* to set the null character to denote end of string */
        fstring[5] = (char) 0x00;
}


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

void    num_to_string( int num, int digits, char *nstring )
{
        int hun,ten,one;


        /* to get the hundred digit value */
        hun = num/100.0;
        nstring[digits - 3] = (char) (0x30 + hun);

        /* to get the ten digit value */
        ten = (num - hun*100)/10.0;
        nstring[digits - 2] = (char) (0x30 + ten);

        /* to get the one digit value */
        one = num - hun*100 - ten*10.0;
        nstring[digits -1] = (char) (0x30 + one);

        /* to set the null character to denote end of string */
        nstring[digits] = (char) 0x00;
}

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

void    time_to_string( int num, char *tstring )
{
        int hr,min,hr1digit,hr2digit,min1digit,min2digit;

        /* to get the hour and minute values */
        hr = num/60.0;
        min = num - hr*60;

        /* to get the first hour digit */
        hr1digit = hr/10.0;
        tstring[0] = (char) (0x30 + hr1digit);

        /* to get the second hour digit */
        hr2digit = hr - hr1digit*10;
        tstring[1] = (char) (0x30 + hr2digit);

        /* to get the colon (:) */
        tstring[2] = 0x3A;

        /* to get the first minute digit */
        min1digit = min/10.0;
        tstring[3] = (char) (0x30 + min1digit);

        /* to get the second minute digit */
        min2digit = min - min1digit*10;
        tstring[4] = (char) (0x30 + min2digit);

        /* to set the null character to denote end of string */
        tstring[5] = (char) 0x00;
}

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

void    speaker_sound( void )
{
        int ii;

        /* turn speaker on and off three times approx. 1 sec in length */
        P1 = 0x01;     /* speaker on  */
        wait();
        P1 = 0x00;     /* speaker off */
        wait();
        P1 = 0x01;     /* speaker on  */
        wait();
        P1 = 0x00;     /* speaker off */
        wait();
        P1 = 0x01;     /* speaker on  */
        wait();
        P1 = 0x00;     /* speaker off */
}

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

void    serial_communication( void ) interrupt 4 using 1
{
        extern int external_input;
        char buffer[10];

        if( RI ) /* tests receive interrupt flag (page 7-42) */
        {
                buffer[0] = getchar();
                /*printf("\n****** serial_comm fn ******\n");
                printf("%c",buffer[0]);*/
                if( buffer[0] == '1' )
                     external_input = 1;
                if( buffer[0] == '2' )
                     external_input = 2;
                if( buffer[0] == '3' )
                     external_input = 3;
                if( buffer[0] == '4' )
                     external_input = 4;
                if( buffer[0] == '5' )
                     external_input = 5;
                /*printf("    external_input = %d\n",external_input);
                printf("******************************\n");*/
        }
}
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1994	3. REPORT TYPE AND DATES COVERED Technical Memorandum
4. TITLE AND SUBTITLE Thermal Enclosure System Functional Simulation User's Manual		5. FUNDING NUMBERS 233-03-03-03
6. AUTHOR(S) A. Terry Morris		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA Langley Research Center Hampton, Virginia 23681-0001		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, D.C. 20546-0001		10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA TM-109094
11. SUPPLEMENTARY NOTES		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category - 66		12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A form and function simulation of the thermal enclosure system (TES) for a microgravity protein crystal growth experiment has been developed as part of an investigation of the benefits and limitations of intravehicular telerobotics to aid in microgravity science and production. A user can specify the time, temperature, and sample rate profile for a given experiment, and menu options and status are presented on an LCD display. This report describes the features and operational procedures for the functional simulation.

14. SUBJECT TERMS Functional Simulation, Thermal Enclosure System (TES), Mockup, Thermal Environment, Timeline Programming, and Space Station Laboratory Module.			15. NUMBER OF PAGES 54 16. PRICE CODE A04
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Unlimited