Kashif Javed | Elena Troubitsyna

Ensuring Mode Consistency for a Complex Fault-Tolerant Distributed Satellite System

Turku Centre *for* Computer Science

TUCS Technical Report
No 1040, Feb 2012

Ensuring Mode Consistency for a Complex Fault-Tolerant Distributed Satellite System

Kashif Javed
Åbo Akademi University, Department of Computer Science
kashif.javed@abo.fi
Elena Troubitsyna
Åbo Akademi University, Department of Computer Science
elena.troubitsyna@abo.fi

TUCS Technical Report
No 1040, Feb 2012

Abstract

Distributed Attitude and Orbit Control System (DAOCS) of the satellite has a complex architecture. There is an essential requirement to ensure error-free operation of the satellite system. Formal modeling and verification can ascertain correct behavior and functioning of various components. This technical report presents a proposed model of a DAOCS. A synchronization and coordination mechanism has been incorporated to deal with mode transitions. The implementation of modeling and verification procedure has exhibited proper working of mode transitions while maintaining required consistency in the system.

Keywords: Distributed mode-rich satellite systems; fault-tolerant systems; syncronization software; and mode consistency

TUCS Laboratory
Distributed Systems Laboratory

1 Introduction

Various modes are designed to control behavior of satellite systems. Different modes are incorporated to meet a number of requirements and to define specific functional profiles of the distributed fault-tolerant system[4,5]. An important problem associated with designing mode-rich satellite systems is to ensure correctness of mode transitions. In this report, an approach has been proposed for modeling and verification of Distributed Attitude and Orbit Control System (D-AOCS) [1,2]. D-AOCS is a typical example of a mode-rich system. To ensure mode consistency, mode managers responsible for controlling different parts of the spacecraft should synchronize their mode transitions. We demonstrate how to model and verify handshaking protocol ensuring that modes are changed consistently. An important part of our modeling is fault tolerance. We demonstrate how to ensure consistency for both normal and backward mode transitions.

2 Architecture

The behavior of Distributed System for Attitude and Orbit Control System for a Single Spacecraft (DSAOCSS) [1] is related to that of centralized Attitude and Orbit Control System (AOCS). DSAOCSS consists of many components, including AOCS Manager, Unit Manager, Several Mode Managers and FDIR (Failure Detection, Isolation and Recovery) Manager. AOCS Manger deals with two controllers (i.e. Control Pointing Controller (CPC) and Phase Pointing Controller (FPC)). The purpose of CPC and FPC is to direct line of sight and is also responsible for coarse and fine accuracy. Unit level state transitions and mode transitions are managed by Unit Manager and Mode Manager respectively. FDIR Manager ensures handling of branch state transition errors and controller phase transition errors [2]. Two managers named as Mode Manager 1 (MM1) and Mode Manager 2 (MM2) are taken to clarify the working of DSAOCSS. The architecture of Unit Manager and Mode Managers are described below.

2.1 Unit Manager

Unit Manager in DSAOCSS organizes the internal states of the units. The components of Unit Manager are supervised by Mode Manager. Earth Sensor (ES), Sun Sensor (SS), Star Tracker (STR), Global Positioning System (GPS), Reaction Wheel (RW), Thruster (THR) and Payload Instrument (PLI) are seven defined controlled units. All unit components are responsible for mode synchronization, decision making on unit states, performing branch state transitions and unit reconfiguration [4,5]. SS, STR, GPS, RW

and PLI provide data to the AOC Manager. RW and THR execute the commands from AOC Manager. These units are also responsible for detection and reporting the branch state transition errors [1].

Every unit comprises of two identical branches (i.e. the nominal branch and the redundant branch). Only one branch is selected from each unit at one time. A unit branch in the 'on' state is always assigned locked status and the unit branch in 'off' state has unlocked status. There are six states of unit components (i.e. on, off, coarse, fine, standby and science). The internal states of ES, SS, STR, RW and THR are either 'on' or 'off'. Three possible GPS's operational states are 'off', 'coarse' and 'fine'. PLI's state can be in 'off', 'standby' or 'science' [3].

2.2 Mode Manager

MM1 and MM2 have their individual modes and each mode comprises of different units and controllers. Each mode manager is responsible to check the preconditions of mode transitions, manages the controllers and units, and completes the mode transitions. Six different types of controlled modes in MM1 and MM2 are Off, Standby, Safe, Nominal, Preparation, and Science. A concise review of the mode functions is as under:

1. ***Off Mode:*** When the central data management unit completes booting of AOCS software, then the satellite instantly goes into off mode.

2. ***Standby Mode:*** The process of separation of the satellite from the launcher is monitored during the standby mode.

3. ***Safe Mode:*** After successful separation from the launcher, the satellite switches to the safe mode. In this mode, initially the satellite obtains a stable attitude and then the CPC is set to the running phase.

4. ***Nominal Mode:*** When satellite enters in this mode, the FPC goes in the running phase and the CPC turns into idle phase. The PLI is needed for measurements because satellite attempts to operate the FPC in this mode.

5. ***Preparation Mode:*** The FPC is achieved in the preparation mode and the PLI gets ready to perform the necessary tasks. navigation.

6. ***Science Mode:*** The PLI in this mode is designed to carry out the required tasks and it stays in science mode till the desired tasks are finished .

The MM1 and MM2 communicate with each other to process the mode transitions in parallel. When a mode transition to off or standby is done, then

every unit branch goes to off state and both controllers are in idle phase. After that, both mode managers communicate with each other. If there is no error then transition to the next mode is executed. When the mode is switched to safe mode in MM1 and MM2, the selected branches of ES, SS and RW are turned to 'on' state and only FPC remains in idle phase. Both mode managers send messages to inform each other that no error exists in the given modes and it turns the mode transition to the nominal mode. If a mode transition goes to the nominal mode in mode managers, the required branches of RW, STR and THR are set to 'on' state and GPS is turned into 'coarse' state. The messages sent and received by the mode managers notify that no unit or controller error has occured. On reaching to the preparation mode, the concerned branches of RW, STR THR are in 'on' state and GPS PLI are in the 'fine' state and 'standby' state respectively. They ensure the correctness of the modes in MM1 and MM2 and make transition to the science mode. In case of the science mode, the preffered branch of PLI operates with 'science' state and remaining every unit branch is same as that of preparation mode. When a mode transition goes to nominal, preparation or science mode, only CPC remains in idle phase. MM1 and MM2 both inform to each other regarding correctness of the system mode.

3 Fault Tolerance

Fault tolerance means that a system continues to run even if an unexpected error has occurred relating to any hardware or software component. There are two ways to recover from a fault tolerant system (can be either in roll forward or roll back error recovery). In roll forward error recovery, if an error occurs in a system, it moves the system to a new error free state. In case of roll back error recovery, if an error occurs in a system, it goes back to some previous state to handle the error. The roll back error recovery is done by mode-downgrading while dealing with mode rich systems. The mode downgradation depends on branch state transition errors and phase transition errors. There are different aspects relating to the branch state transition errors. When a branch state transition error on the redundant branch of ES, RW or SS occurs and there is no error in the remaining redundant branches, then the mode goes back to off mode. If the redundant branch of GPS, STR or THR gets corrupted, it results a mode transition to safe. A mode transition to nominal takes place when there is a branch state transition error on the redundant branch of PLI. The important error checks are incorporated while dealing with the attitude or phase transitions. When the current mode is safe and a non-ignored phase error is produced, it results a mode transition to off. If the phase error is generated in the nominal, then it goes back to safe. In case the existing mode is preparation and a phase error occurs, a

mode transition to nominal takes place. A mode transition to preparation takes place when a phase error occurs in the science mode [3]. In case of unit reconfiguration, a branch state transition error on the nominal branch of ES, SS, RW, GPS, STR, THR or PLI causes a unit reconfiguration if there is no branch state transition error on the redundant branches of ES, SS, RW, GPS, STR, THR and PLI. If the mode task is not completed within a given time interval or multiple errors occur in the unit branches and controller phases, then timeout signal is produced for safe condition.

4 Handshake Protocol

Handshaking is a process in which connection is established among two processes and information is transferred from one process to another without the need for human involvement to set constraints. MM1 and MM2 do handshake with each other to update the condition of their modes. Different scenarios of handshake protocol are explained covering the following key points: If all conditions of unit states and controller phases within each mode of MM1 and MM2 fulfill their requirements, then mode managers pass the 'no error' message to notify that the mode is in error-free state. It results into the forward mode transition and mode manager switches the current mode to the next mode as described in section II. If an error occurs in the mode of MM1 and there is no error in the mode of MM2, then MM1 sends an 'error' message to MM2 . MM1 starts error recovery according to the section III. Until the error recovery of MM1 is not completed, MM2 keeps on waiting. After the successful error recovery, both mode managers proceed to the next mode. When an error occurs only in the mode of MM2, then MM1 receives an 'error' message from MM2. MM1 waits until error has been recovered in MM2. The error recovery is done as mentioned in section III. The mode managers switch to next mode after receiving the information from MM2 that the error is recoverd. On receiving an 'error' message from MM1 and MM2 simultaneously, error recovery starts in both mode managers. The backward mode transitions are executed in MM1 and MM2. After achieving the successful recovery, mode managers move to the next mode. There are two types of errors (i.e. unit branch state transition errors and controller phase transition errors). Handshaking algorithm for handling such type of errors is very challenging and is shown as under:

```
void handshake(int u_MM1, int u_MM2,int c_MM1,int c_MM2) {
// 'u' denotes unit error flag and 'c' denotes controller error flag
 if (u_MM1==0&&u_MM2==0&&c_MM1==0&&c_MM2==0) {
/* The associated code illustrates that no error occurs in the unit
branchs of ES, SS, RW, GPS, STR, THR or PLI and controller phase of
 CPC or FPC in the given mode of MM1 and MM2. It accounts the forward
```

```
 mode transition according to the section II. */}
elseif(u_MM1==1&&u_MM2==0&&c_MM1==0&&c_MM2==0) {
/* The associated code illustrates that an error occurs in the unit
branch of ES, SS, RW, GPS, STR, THR or PLI in the given mode of MM1.
 It accounts the backward mode transition according to the section III.
 MM2 stays on waiting until an error is recovered. */}
elseif(u_MM1==0&&u_MM2==1 &&c_MM1==0&&c_MM2==0) {
/* The associated code illustrates that an error occurs in the unit
branch of ES, SS, RW, GPS, STR, THR or PLI in the given mode of MM2.
It accounts the backward mode transition according to the section III.
MM1 stays on waiting until an error is recovered. */}
elseif(u_MM1==1&&u_MM2==1&&c_MM1==0&&c_MM2==0) {
/* The associated code illustrates that an error occurs in the unit
 branch of ES, SS, RW, GPS, STR, THR or PLI in the given mode of both
 mode managers. MM1 and MM2 account the backward mode transition according
 to the section III. */}
elseif(u_MM1==0&&u_MM2==0&&c_MM1==1&&c_MM2==0) {
/* The associated code illustrates that an error occurs in the controller
 phase of CPC or FPC in the given mode of MM1. It accounts the backward
 mode transition according to the section III.  MM2 stays on waiting until
 an error is recovered. */}
elseif(u_MM1==0&&u_MM2==0&&c_MM1==0&&c_MM2==1) {
/* The associated code illustrates that an error occurs in the controller
phase of CPC or FPC in the given mode of MM2. It accounts the backward
 mode transition according to the section III.  MM1 stays on waiting until
 an error is recovered. */}
elseif(u_MM1==0&&u_MM2==0&&c_MM1==1&&c_MM2==1) {
/* The associated code illustrates that an error occurs in the controller
phase of CPC or FPC in the given mode of both mode managers. MM1 and MM2
account the backward mode transition according to the section III. */}
else {
/* The associated code describes that it is an invalid condition.
Program is terminated.*/} }
```

5 Proposed System Design Using Handshake

The proposed system design has been implemented using SystemC. SystemC is important at system level for functional verification and support event driven simulation environments [6]. It offers application program interface for transaction based verification, handling exceptions and verification tasks [7]. The system model comprises of six defined modes named as A (Off), B (Standby), C (Safe), D (Nominal), E (Preparation) and F (Science). Three

different operations have been implemented (i.e. forward mode transitions, backward mode transitions, and unit reconfiguration). The flow chart given in Figure 1 shows detailed design structure only for Mode E to Mode F of the system. After necessary declarations of modes, units and controllers, the different verification procedures are described in the following sections.

5.1 Verification of Forward Mode Transition

When all the unit states are in off state, controller phases are in idle phase, and no unit reconfiguration is in progress, then current mode is A in MM1 and MM2. The unit/controller error flag is set to low and mode managers exchange the information of error-free mode status. After this, the mode moves forward to the next mode (i.e. Mode B) in MM1 and MM2. So in this way, when all conditions of unit states and controller phases within each mode of each manager fulfill their requirements through handshake protocol, mode managers update the error-free mode conditions to each other. Then the current mode switches to the next mode within each mode manager until it completes its operation after Mode F. The following part of the implemented code corresponds to the forward mode transition for MM1 and MM2.

```
for (int mode = 1; mode < 7; mode++){
if(mode==A) {// Off Mode
ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;
PLI1=off;CPC1=idle;FPC1=idle; u_MM1=0; c_MM1=0;
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;
PLI2=off;CPC2=idle;FPC2=idle; u_MM2=0; c_MM2=0;
/* The remaining part of the code, by calling the handshake protocol
function on the basis of unit and controller error flag, is mentioned
in section IV.*/}
else if(mode==B) {// Standby Mode
ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;
PLI1=off;CPC1=idle;FPC1=idle; u_MM1=0; c_MM1=0;
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;
PLI2=off; CPC2=idle;FPC2=idle; u_MM2=0; c_MM2=0;
/* The remaining part of the code, by calling the handshake protocol
function on the basis of unit and controller error flag, is mentioned
in section IV.*/}
else if(mode==C) { // Safe Mode
ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;THR1=off;
PLI1=off;CPC1=run;FPC1=idle; u_MM1=0; c_MM1=0;
ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;THR2=off;
PLI2=off; CPC2=run;FPC2=idle; u_MM2=0; c_MM2=0;
/* The remaining part of the code, by calling the handshake protocol
function on the basis of unit and controller error flag, is mentioned
```

```
in section IV.*/}
else if(mode==D) { // Nominal Mode
ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;
PLI1=off;CPC1=idle;FPC1=run; u_MM1=0; c_MM1=0;
ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;
PLI2=off; CPC2=idle;FPC2=run; u_MM2=0; c_MM2=0;
/* The remaining part of the code, by calling the handshake protocol
function on the basis of unit and controller error flag, is mentioned
in section IV.*/}
else if(mode==E) {// Preparation Mode
ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;
PLI1=Standby;CPC1=idle;FPC1=run;u_MM1=0;c_MM1=0;
ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;
PLI2=Standby;CPC2=idle;FPC2=run;u_MM2=0;c_MM2=0;
/* The remaining part of the code, by calling the handshake protocol
function on the basis of unit and controller error flag, is mentioned
in section IV.*/}
else if(mode==F) { // Science Mode
ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;
PLI1=Science;CPC1=idle;FPC1=run;u_MM1=0;c_MM1=0;
ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;
PLI2=Science;CPC2=idle;FPC2=run;u_MM2=0;c_MM2=0;
/* The remaining part of the code, by calling the handshake protocol
function on the basis of unit and controller error flag, is mentioned
in section IV.*/}
else  cout<<" Program is terminated.";}
```

5.2 Verification of the Steps in the Backward Mode Transition

The backward mode transition depends on the two types of errors (i.e. unit branch state transition error and controller phase transition error). Handshaking procedure for handling these errors is given below.

1. ***Verification of the Steps in Unit Branch State Transition Error***

   Initially, all the unit states are in the off state and the controller phases are in idle phase. No unit reconfiguration is in progress and the current mode is A of MM1 and MM2. The unit/controller error flag is set to low. The mode transition to the Mode B of both mode managers is autonomous for constantly monitoring and ensuring of separation of the spacecraft from the launcher. The system automatically goes into the Mode C after ensuring to both mode managers that the task of

Mode B is successfully completed. If there is an error in ES, SS or RW in the given mode of any mode manager, the mode manager with error goes back to Mode A after informing the unit error status to the error-free mode manager and the error-free mode manager waits for error recovery. In case of an error in GPS, THR or STR in the mode of MM1 or MM2, the mode manager with error enters in Mode C after updating to the error-free mode manager. The error-free mode manager waits until error recovery is done. If error occurs in both managers simultaneously then both will execute the backward transition according to the error. At Mode C, the satellite has attained secure attitude and required coarse pointing. Now the system switches to Mode D of MM1 and MM2. If there is an error in PLI in given mode of any mode manager, the mode manager with error enters in Mode D again after notifying the other mode manager that has to wait for error recovery. In case unit error in ES, SS or RW and GPS, THR or STR, the system encounters the same operation that is described in above paragraph for Mode C. Otherwise, it goes to Mode E after fully utilizing FPC in order to prepare the PLI for measurements. If there is an error in ES, SS or RW at Mode E of any mode manager, the effected mode manager again switches to Mode A. If an error occurs in GPS, THR or STR in mode of any manager, the effected mode manager returns to Mode C. However, if PLI gets an error, the effected mode manager goes to Mode D. Before backward transition to the desired mode, the messages exchange between the effected mode manager and the error-free mode manager to acknowledge the error status. In case error arises in both mode managers simultaneously then both will perform the backward transition according to the error. In Mode E, proper FPC is ensured so that the PLI becomes ready to complete the required measurements. After this, the system enters in Mode F of MM1 and MM2. While in Mode F, if any mode manager receives an error in ES, SS or RW, then it moves to Mode A. In case of an error in GPS, THR or STR, it switches to Mode C. If there is an error in PLI, it returns to Mode D. When error takes place in both mode managers at the same time then corresponding to the error type, both will make the backward mode transition. The mode managers exchange the information of the error status then carry out backward transition to the desired mode. When the science mode is fully achieved, it means that the system has successfully completed its assigned tasks. The following scenario is taken to describe the unit branch transition error in case of Mode D.

```
if(mode==D) {// Nominal Mode
if((ES1!=off || SS1!=off || RW1!=on) && STR1==on && GPS1==coarse &&
 THR1==on && PLI1==off && CPC1==idle && FPC1==run && ES2==off &&
```

```
 SS2==off && RW2==on && STR2==on && GPS2==coarse && THR2==on &&
 PLI2==off && CPC2==idle && FPC2==run){
u_MM1=1;c_MM1=0;
u_MM2=0;c_MM2=0;
/* The remaining part of the code, by calling the handshake protocol
function on the basis of unit and controller error flag, is mentioned
in section IV.*/}
else if(ES1==off && SS1==off && RW1==on && STR1==on && GPS1==coarse &&
THR1==on && PLI1==off && CPC1==idle && FPC1==run && ES2==off &&
SS2==off && RW2==on && (STR2!=on || GPS2!=coarse || THR2!=on) &&
PLI2==off && CPC2==idle && FPC2==run){
u_MM1=0;c_MM1=0;
u_MM2=1;c_MM2=0;
/* The remaining part of the code, by calling the handshake protocol
function on the basis of unit and controller error flag, is mentioned
in section IV.*/}
else if(ES1==off && SS1==off && RW1==on && STR1==on && GPS1==coarse &&
THR1==on && PLI1!=off && CPC1==idle && FPC1==run && ES2==off &&
SS2==off&& RW2==on && STR2==on && GPS2==coarse && THR2==on && PLI2!=off
&&CPC2==idle && FPC2==run){
u_MM1=1;c_MM1=0;
u_MM2=1;c_MM2=0;
/* The remaining part of the code, by calling the handshake protocol
function on the basis of unit and controller error flag, is mentioned
in section IV.*/}
else{
/* The associated code describes that no transitions take place. */ }
}
else  cout<<" Program is terminated.";
```

2. ***Verification of the Steps in Controller Phase Transition Errors*** When CPC and FPC do not fulfill the requirement of mode of any mode manager, the error flag is set to high and the affected mode manager is downgraded to previous mode after utilizing the handshake protocol by sending message to error-free mode manager. In case the phase of controllers in the given mode of both mode managers is corrupted, then both managers do the backward mode transition at once after acknowledging each other. Likewise if error occurs in any controller phase in any mode of any mode manager, the mode of effected mode manager is downgraded in accordance with the principles defined in fault-tolerance. Modes A, B and C are processed normally like forward mode transitions. After Mode C, if there is an error in CPC or

FPC, the system goes back to Mode A. Otherwise, it moves to Mode D. At Mode D, in case of an error in CPC or FPC, it shifts back to Mode C, and goes to Mode E if no error is encountered. Similarly, the mode switches from E to D and F to E if there is an error at Modes E and F respectively. Likewise, the system moves from Mode E to Mode F if there is no error. Now it completes the given tasks as long as they are required. The following portion of the code represents the scenario of phase transition for Mode E.

```
if(mode==E) {// Preparation Mode
if(ES1==off && SS1==off && RW1==on && STR1==on && GPS1==fine && THR1==on
&& PLI1==standby && CPC1!=idle && FPC1==run && ES2==off && SS2==off &&
RW2==on && STR2==on && GPS2==fine && THR2==on && PLI2==standby &&
CPC2==idle && FPC2==run){
u_MM1=0;c_MM1=1;
u_MM2=0;c_MM2=0;
/* The remaining part of the code, by calling the handshake protocol
function on the basis of unit and controller error flag, is mentioned
in section IV.*/}
else if(ES1==off && SS1==off && RW1==on && STR1==on && GPS1==fine &&
THR1==on && PLI1==standby && CPC1==idle && FPC1==run && ES2==off &&
SS2==off && RW2==on && STR2==on && GPS2==fine && THR2==on &&
PLI2==standby && CPC2==idle && FPC2!=run){
u_MM1=0;c_MM1=0;
u_MM2=0;c_MM2=1;
/* The remaining part of the code, by calling the handshake protocol
function on the basis of unit and controller error flag, is mentioned
in section IV.*/}
else if(ES1==off && SS1==off && RW1==on && STR1==on && GPS1==fine &&
THR1==on && PLI1==standby && CPC1==idle && FPC1!=run && ES2==off &&
SS2==off && RW2==on && STR2==on && GPS2==fine && THR2==on &&
PLI2==standby && CPC2!=idle && FPC2==run){
u_MM1=0;c_MM1=1;
u_MM2=0;c_MM2=1;
/* The remaining part of the code, by calling the handshake protocol
function on the basis of unit and controller error flag, is mentioned
in section IV.*/}
else{
/* The associated code describes that no transitions take place. */ }
}
else  cout<<" Program is terminated.";
```

3. ***Verification of the Steps in Unit Reconfiguration***

If error exists on nominal unit branch at any mode of MM1 or MM2, then it is replaced by redundant unit branch in the given mode of mode manager. The unit reconfiguration is done to complete the remaining operation of the system according to the principles that is defined in the fault-tolerance. Unit reconfiguration is, however, a burden on the system and takes some time while switching from nominal branch to redundant branch of the unit. In case of the nominal unit branch in the given mode of both mode managers is corrupted, then unit reconfiguration is done in both mode manager after exchanging the information between the mode managers regarding unit reconfiguration. The following piece of the code shows the scenario of unit reconfiguration for Mode F.

```
 if (mode==F) { //  Science Mode
/*N_ES1, N_SS1, N_RW1, N_GPS1, N_STR1, N_THR1, N_PLI1 and N_ES2, N_SS2,
N_RW2, N_GPS2, N_STR2, N_THR2, N_PLI2 are Nominal Unit branches of MM1 and
MM2 respectively. R_ES1, R_SS1, R_RW1, R_GPS1, R_STR1, R_THR1, R_PLI1 and
R_ES2, R_SS2, R_RW2, R_GPS2, R_STR2, R_THR2, R_PLI2 are Redundant Unit
branches of MM1 and MM2 respectively */
if((N_ES1!=off || N_SS1!=off || N_RW1!=on) && R_ES1==off && R_SS1==off &&
R_RW1==on && N_ES2==off && N_SS2==off && N_RW2==on && R_ES2==off &&
R_SS2==off && R_RW2==on ) {
u_MM1=1;c_MM1=0;
  u_MM2=0;c_MM2=0;
/* The remaining part of the code, by calling the handshake protocol
function on the basis of unit and controller error flag, is mentioned in
section IV.*/}
else if(N_GPS1==coarse && N_STR1==on && N_THR1==on && N_PLI1==off &&
R_GPS1==coarse && R_STR1==on && R_THR1==on && R_PLI1==off &&
N_GPS2!=coarse|| N_STR2!=on || N_THR2!=on ||N_PLI2!=off) && R_GPS2==coarse
&& R_STR2==on&& R_THR2==on && R_PLI2==off) {
u_MM1=0;c_MM1=0;
u_MM2=1;c_MM2=0;
/* The remaining part of the code, by calling the handshake protocol
function on the basis of unit and controller error flag, is mentioned in
section IV.*/}
else if((N_ES1!=off || N_SS1!=off || N_RW1!=on) && R_ES1==off && R_SS1==off
&& R_RW1==on && (N_ES2==off || N_SS2!=off || N_RW2!=on) && R_ES2==off &&
R_SS2==off && R_RW2==on ) {
  u_MM1=1;c_MM1=0;
u_MM2=1;c_MM2=0;
/* The remaining part of the code, by calling the handshake protocol
```

```
function on the basis of unit and controller error flag, is mentioned in
section IV.*/}
else{
/* The associated code describes that no transition takes place. */ }
}
else  cout<<" Program is terminated.";}
```

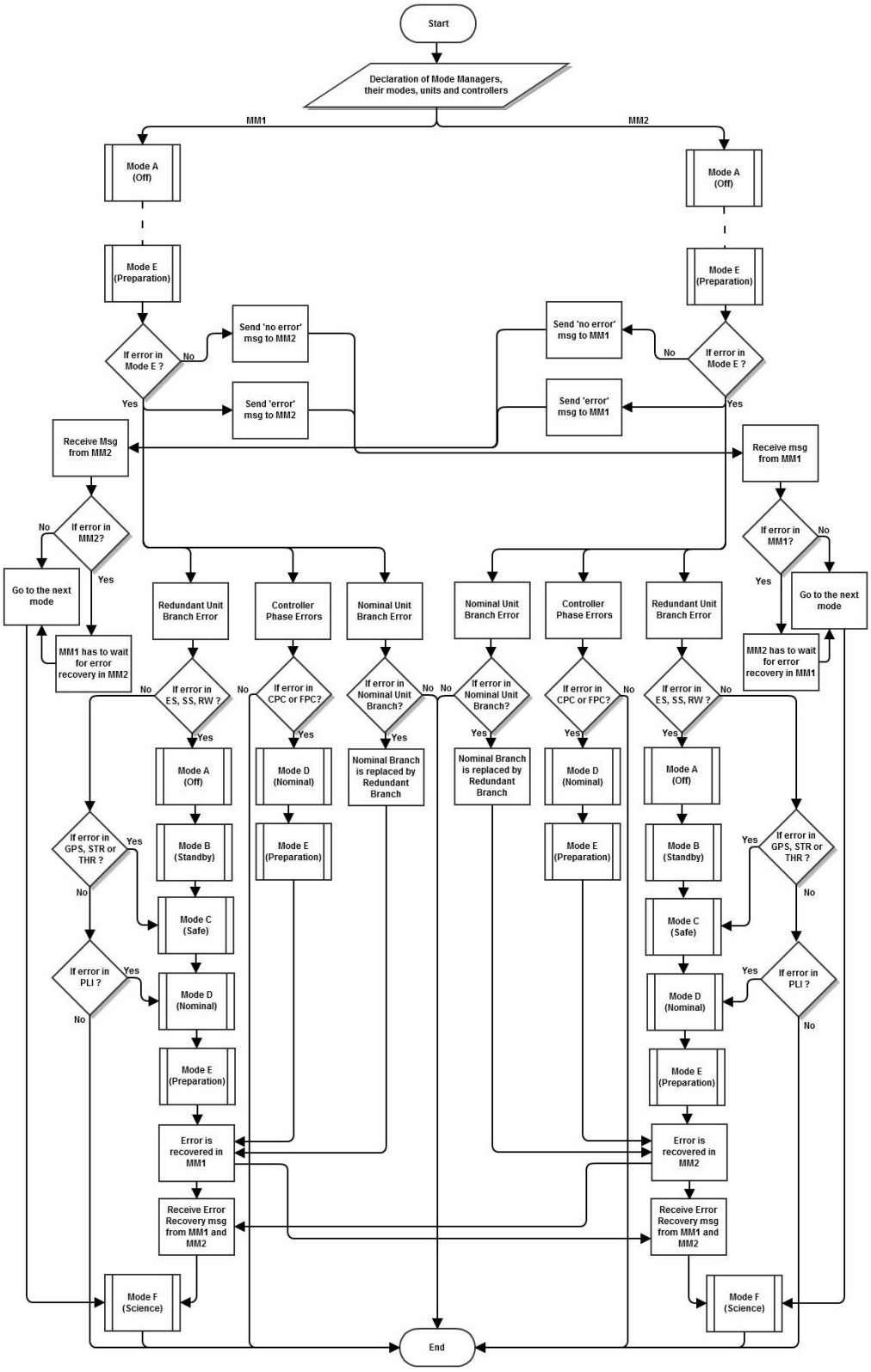

Figure 1: System flow chart for Mode E to Mode F

6 Conclusions and Future Work

An effort has been made in this report to demonstrate how to model and verify distributed satellite systems with complex mode transition logic. The proposed methodology is validated by a case study - design of a distributed Attitude and Orbit Control System. The system has been implemented in SystemC language. To perform formal verification, SystemC specification can be easily interfaced with various model checking techniques. So, it is preferred over other programming languages. Our previously reported work concentrated on modeling centralized mode-rich system but the research work presented in this report is an extension to the architectural design of our earlier system. In this approach, we have put the main focus on mode synchronization aspect and demonstrated how to achieve mode consistency via handshaking protocol. Our work complements research done on formal modeling of mode-rich satellite systems. Formal modeling of the distributed architecture presented in our report is a completely novel aspect and describes verification of the steps in different mode transition errors and unit reconfiguration. As a future research work, the authors are planning to investigate how to interface architectural modeling with the proposed scheme.

References

[1] "DEPLOY Work Package 3 - Software Requirements Document for Distributed System for Attitude and Orbit Control for a Single Spacecraft", Space Systems Finland, Ltd., June 2011

[2] "DEPLOY Work Package 3 - Attitude and Orbit Control System Software Requirements Document", Space Systems Finland, Ltd., December 2010.

[3] K. Javed and E. Troubitsyna, "Designing a Fault-Tolerant Satellite System in SystemC", In Proc. of International Conference of Systems, IEEE Computer Press, March 2012. To appear.

[4] M. Heimdahl and N. Leveson, "Completeness and Consistency in Hierarchical State-Based Requirements", IEEE Transactions on Software Engineering, Vol.22, No. 6, June 1996, pp. 363-377.

[5] N. Leveson, L. D. Pinnel, S. D. Sandys, S. Koga, and J. D. Reese, "Analyzing Software Specifications for Mode Confusion Potential", Proceedings of Workshop on Human Error and System Development, C.W. Johnson, Editor, March 1997, Glasgow, Scotland, pp. 132-146.

[6] Blanc, N., Kroening, D., and Sharygina, N. 2008. Scoot: A tool for the analysis of SystemC models. In TACAS. Lecture Notes in Computer Science. Springer, 467-470..

[7] L. Singh and L. Drucker, "Advanced Verification Techniques: A SystemC Based Approach for Successful Tapeout", Springer, 2004.

[8] A. Iliasov, E. Troubitsyna, L. Laibinis, A. Romanovsky, K. Varpaaniemi, D. Ilic, T. Latvala, Developing Mode-Rich Satellite Software by Refinement in Event B. In: Proc.of FMICS 2010, the 15th International Workshop on Formal Methods for Industrial Critical Systems, Lecture Notes for Computer Science, Springer, 2010.

[9] A. Iliasov, E. Troubitsyna, L. Laibinis, A. Romanovsky, K. Varpaaniemi, P. Visnen, D. Ilic, T. Latvala, Verifying Mode Consistency for On-Board Satellite Software. In Proc. of SAFECOMP 2010, The 29th International Conference on Computer Safety, Reliability and Security, September 14-17, Vienna, Austria, Lecture Notes for Computer Science, Springer, 2010

Appendix

System C

```
#include <iostream>
#include <windows.h>
#include <cstdlib>
#include <string>
using namespace std;
void operation(int,int,int,int,int,int,int,int,int,int,int,int,int,int,int,
   int,int,int,int,int,bool,bool,bool,bool);
void error_recovery(int,int,int,int,int,int,int,int,int,int,int,bool);
void controller_phase(int,int,int,int);
void handshake(bool,bool,bool,bool);
void internal_func1(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func2(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func3(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func4(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func5(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func6(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func7(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func8(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func9(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func10(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func11(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func12(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func13(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func14(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func15(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func16(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func17(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func18(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func19(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func20(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func21(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func22(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func23(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func24(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func25(int,int,int,int,int,int,int,int,int,int,bool);
void internal_func26(int,int,int,int,int,int,int,int,int,int,bool);
int AOCS_cycles=5000; //time = 5sec
int main()
{HANDLE hOut;hOut = GetStdHandle(STD_OUTPUT_HANDLE);
```

```
int in,i,ii,m;
char c;bool b=true;
// unit states
const int off=0;const int on=1;const int coarse=2;const int fine=3;
const int unit=0;const int Standby=4;const int Science=5;
const int idle=0;const int run=1;int FPC1,CPC1;int FPC2,CPC2;
// Each unit has two branches i.e. Nominal and Redundant
int ES1,SS1,GPS1,STR1,RW1,THR1,PLI1; // MM1 Units
int ES2,SS2,GPS2,STR2,RW2,THR2,PLI2; // MM2 Units
//Redundant branch of units
int R_ES1,R_SS1,R_GPS1,R_STR1,R_RW1,R_THR1,R_PLI1;
int R_ES2,R_SS2,R_GPS2,R_STR2,R_RW2,R_THR2,R_PLI2;
//Nominal branch of units
int N_ES1,N_SS1,N_GPS1,N_STR1,N_RW1,N_THR1,N_PLI1;
int N_ES2,N_SS2,N_GPS2,N_STR2,N_RW2,N_THR2,N_PLI2;
bool low= false;bool high= true;
bool c_e_MM1,c_e_MM2;//controller errors
bool u_e_MM1,u_e_MM2;// unit errors
do
{cout<<"\t\t  ";
SetConsoleTextAttribute(hOut, BACKGROUND_RED | BACKGROUND_GREEN |
BACKGROUND_BLUE | BACKGROUND_INTENSITY);
printf(" Welcome to the Distributed System of AOCS \n\n\n");
SetConsoleTextAttribute(hOut,FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
cout<<"Would you like to run this program"<<"\n";
cout<<"Press 'Y' for Yes"<<"\n"<<"Press 'N' for No"<<endl;
cin>>c;
cout<<endl;
if(c == 'y' || c == 'Y')b=true;
else if (c == 'n' || c == 'N')b=false;
else cout<<"Please enter either 'y' or 'n'"<<endl;}
while ((c!='y') && (c!='Y') && (c!='n')&& (c!='N'));
if (c == 'n' || c == 'N') cout<<"Program terminated"<<endl;
while(b)
{if(c == 'y' || c == 'Y')
cout<<"There are six types of Mode.\n\n";
cout<<"Mode A : Off\n\n";cout<<"Mode B : Standby\n\n";
cout<<"Mode C : Safe\n\n";cout<<"Mode D : Nominal\n\n";
cout<<"Mode E : Preparation\n\n";
cout<<"Mode F : Science\n\n";
cout<<"Select one of the following operation\n\n";
cout<<"1.\tError Free System\n"<<endl;
```

```
cout<<"2.\tFaulty System\n"<<endl;
cout<<"Please press '1' or '2'"<<endl;
cin>>in;cout<<endl;
if(in==1) // Error Free System
{ c_e_MM1=low;c_e_MM2=low;cout<<"Note:\n";
cout<<" '0' denotes 'Off state' & 'Idle phase'\n";
cout<<" '1' denotes 'On state' & 'Running phase'\n";
cout<<" '2' denotes 'Coarse state'\n";
cout<<" '3' denotes 'Fine state'\n";
cout<<" '4' denotes 'Standby state'\n";
cout<<" '5' denotes 'Science state'\n\n";
for (int mode = 1; mode < 7; mode++)//
{if(mode==1)
{ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;PLI1=off;
CPC1=idle;FPC1=idle;CPC2=idle;FPC2=idle;u_e_MM1=low;u_e_MM2=low;
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;PLI2=off;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if(mode==2)
{ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;PLI1=off;
CPC1=idle;FPC1=idle;CPC2=idle;FPC2=idle;u_e_MM1=low;u_e_MM2=low;
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;PLI2=off;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if(mode==3)
{ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;THR1=off;PLI1=off;
CPC1=run;FPC1=idle;CPC2=run;FPC2=idle;u_e_MM1=low;PLI2=off;
u_e_MM2=low;ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;THR2=off;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if(mode==4)
{ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;PLI1=off;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;u_e_MM1=low;PLI2=off;
u_e_MM2=low;ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if(mode==5)
{ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;PLI1=Standby;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;u_e_MM1=low;PLI2=Standby;
u_e_MM2=low;ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if(mode==6)
```

```
{ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;PLI1=Science;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;u_e_MM1=low;PLI2=Science;
u_e_MM2=low;ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else cout<<"Program Terminated\n\n\n";}}
else if(in==2)   //  Faulty System
{cout<<"Select Error type\n\n";
cout<<"1.\tUnit State Errors\n"<<endl;
cout<<"2.\tController Phase Errors\n"<<endl;
cout<<"Please press '1' or '2'"<<endl;
cin>>i;cout<<endl;
if(i==1)// Unit State Errors
{cout<<"Select Error type\n\n";
cout<<"1.\tRedundant Branch Errors\n"<<endl;
cout<<"2.\tNominal Branch Errors\n"<<endl;
cout<<"Please press '1' or '2'"<<endl;
cin>>ii;cout<<endl;cout<<"Note:\n";
cout<<" '0' denotes 'Off state' & 'Idle phase'\n";
cout<<" '1' denotes 'On state' & 'Running phase'\n";
cout<<" '2' denotes 'Coarse state'\n";
cout<<" '3' denotes 'Fine state'\n";
cout<<" '4' denotes 'Standby state'\n";
cout<<" '5' denotes 'Science state'\n\n";
if(ii==1)// Redundant Branch Errors
{for (int mode = 1; mode < 7; mode++)
{if(mode==1)//off
{ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;PLI1=off;
CPC1=idle;FPC1=idle;CPC2=idle;FPC2=idle;
u_e_MM1=low;u_e_MM2=low;
c_e_MM1=low;c_e_MM2=low;
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;PLI2=off;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if(mode==2)//standby
{ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;PLI1=off;
CPC1=idle;FPC1=idle;CPC2=idle;FPC2=idle;
u_e_MM1=low;u_e_MM2=low;
c_e_MM1=low;c_e_MM2=low;
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;PLI2=off;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if(mode==3)//safe
```

```
{ES1=on;SS1=off;RW1=on; ES2=off;SS2=on;RW2=on;
GPS1=off;STR1=off;THR1=off; GPS2=off;STR2=off;THR2=off;
PLI1=off; PLI2=off;
CPC1=run;FPC1=idle; CPC2=run;FPC2=idle;
u_e_MM1=low;c_e_MM1=low; c_e_MM2=low;u_e_MM2=low;
if(ES1==on && SS1==on && RW1==on && GPS1==off && STR1==off && THR1==off
   && PLI1==off && ES2==on && SS2==on && RW2==on && GPS2==off &&
   STR2==off && THR2==off && PLI2==off)
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if ((ES1!=on || SS1!=on || RW1!=on) && GPS1==off && STR1==off &&
 THR1==off && PLI1==off &&ES2==on && SS2==on && RW2==on &&
 GPS2==off && STR2==off && THR2==off && PLI2==off)
{u_e_MM1=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;PLI1=off;CPC1=idle;
FPC1=idle;m=1;u_e_MM1=low;
error_recovery(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,CPC1,FPC1,m,u_e_MM1);}
else if ((ES2!=on || SS2!=on || RW2!=on) && GPS1==off && STR1==off &&
 THR1==off && PLI1==off && ES1==on && SS1==on && RW1==on &&
 GPS2==off &&STR2==off && THR2==off && PLI2==off)
{u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;PLI2=off;CPC2=idle;
FPC2=idle;m=2;u_e_MM2=low;
error_recovery(ES2,SS2,GPS2,STR2,RW2,THR2,PLI2,CPC2,FPC2,m,u_e_MM2);}
else if (ES1==on && SS1==on && RW1==on && (GPS1!=off || STR1!=off ||
 THR1!=off) && PLI1==off &&ES2==on && SS2==on && RW2==on &&
 GPS2==off && STR2==off && THR2==off && PLI2==off)
{u_e_MM1=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;THR1=off;PLI1=off;CPC1=run;
FPC1=idle;m=3;u_e_MM1=low;
error_recovery(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,CPC1,FPC1,m,u_e_MM1);}
else if (ES1==on && SS1==on && RW1==on && (GPS2!=off || STR2!=off ||
 THR2!=off) && PLI1==off && ES2==on && SS2==on && RW2==on &&
 GPS1==off && STR1==off && THR1==off && PLI2==off)
{u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
```

```
ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;THR2=off;PLI2=off;CPC2=run;
FPC2=idle;m=4;u_e_MM2=low;
error_recovery(ES2,SS2,GPS2,STR2,RW2,THR2,PLI2,CPC2,FPC2,m,u_e_MM2);}
else if ((ES1!=on || SS1!=on || RW1!=on || ES2!=on || SS2!=on ||
 RW2!=on) && GPS1==off && STR1==off && THR1==off && PLI1==off
 && GPS2==off && STR2==off && THR2==off && PLI2==off)
{u_e_MM1=high;u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;PLI1=off;
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;PLI2=off;
m=8;u_e_MM1=low;u_e_MM2=low;CPC1=idle;FPC1=idle;CPC2=idle;FPC2=idle;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
m=2;operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,
THR2,PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=on;SS1=on;RW1=on;ES2=on;SS2=on;RW2=on;m=3;CPC1=run;CPC2=run;
u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"\nError recovered in Safe Mode";
cout<<"\t\tError recovered in Safe Mode";}
else if (ES1==on && SS1==on && RW1==on && (GPS1!=off || STR1!=off ||
 THR1!=off || GPS2!=off || STR2!=off || THR2!=off) && PLI1==off
 && ES2==on && SS2==on && RW2==on && PLI2==off)
{u_e_MM1=high;u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;THR1=off;PLI1=off;
ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;THR2=off;PLI2=off;
CPC1=run;FPC1=idle;CPC2=run;FPC2=idle;m=9;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"\nError recovered in Safe Mode";
cout<<"\t\tError recovered in Safe Mode";}
else if ((ES1!=on || SS1!=on || RW1!=on || ES2!=on || SS2!=on ||
 RW2!=on) && (GPS1!=off || STR1!=off || THR1!=off || GPS2!=off
 || STR2!=off || THR2!=off) && ( PLI1!=off ||  PLI2!=off))
{u_e_MM1=high;u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"Multiple unit errors occur";
cout<<"\t\tMultiple unit errors occur";
```

```
cout<<"\nTimeout !!!";cout<<"\t\tTimeout !!!"; goto stop;}
else cout<<"\n\n";}
else if (mode==4)//nominal
{ES1=off;SS1=off;RW1=on; ES2=off;SS2=off;RW2=on;
GPS1=coarse;STR1=on;THR1=off; GPS2=coarse;STR2=on;THR2=on;
PLI1=off; PLI2=off;
CPC1=idle;FPC1=run; CPC2=idle;FPC2=run;
u_e_MM1=low;u_e_MM2=low;c_e_MM1=low;c_e_MM2=low;
if(ES1==off && SS1==off && RW1==on && GPS1==coarse && STR1==on &&
   THR1==on && PLI1==off &&ES2==off && SS2==off && RW2==on &&
   GPS2==coarse && STR2==on && THR2==on && PLI2==off)
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if ((ES1!=off || SS1!=off || RW1!=on) && GPS1==coarse && STR1==on
 && THR1==on && PLI1==off && ES2==off && SS2==off && RW2==on &&
 GPS2==coarse && STR2==on && THR2==on && PLI2==off)
{u_e_MM1=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;PLI1=off;CPC1=idle;
FPC1=idle;m=5;u_e_MM1=low;
error_recovery(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,CPC1,FPC1,m,u_e_MM1);}
else if ((ES2!=off || SS2!=off || RW2!=on) && GPS1==coarse && STR1==on
 && THR1==on && PLI1==off && ES1==off && SS1==off && RW1==on &&
 GPS2==coarse && STR2==on && THR2==on && PLI2==off)
{u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;PLI2=off;CPC2=idle;
FPC2=idle;m=6;u_e_MM2=low;
error_recovery(ES2,SS2,GPS2,STR2,RW2,THR2,PLI2,CPC2,FPC2,m,u_e_MM2);}
else if (ES1==off && SS1==off && RW1==on && (GPS1!=coarse || STR1!=on
 || THR1!=on) && PLI1==off && ES2==off && SS2==off && RW2==on
 && GPS2==coarse && STR2==on && THR2==on && PLI2==off)
{u_e_MM1=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;THR1=off;PLI1=off;CPC1=run;
FPC1=idle;m=7;u_e_MM1=low;
error_recovery(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,CPC1,FPC1,m,u_e_MM1);}
else if (ES1==off && SS1==off && RW1==on && (GPS2!=coarse || STR2!=on
 || THR2!=on) && PLI1==off && ES2==off && SS2==off && RW2==on
 && GPS1==coarse && STR1==on && THR1==on && PLI2==off)
```

```
{u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;THR2=off;PLI2=off;CPC2=run;
FPC2=idle;m=8;u_e_MM2=low;
error_recovery(ES2,SS2,GPS2,STR2,RW2,THR2,PLI2,CPC2,FPC2,m,u_e_MM2);}
else if (ES1==off && SS1==off && RW1==on && GPS1==coarse && STR1==on
 && THR1==on && ES2==off && SS2==off && RW2==on && GPS2==coarse
 && STR2==on && THR2==on && PLI1!=off && PLI2==off)
{u_e_MM1=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;PLI1=off;CPC1=idle;
FPC1=run;m=9;u_e_MM1=low;
error_recovery(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,CPC1,FPC1,m,u_e_MM1);}
else if (ES1==off && SS1==off && RW1==on && GPS1==coarse && STR1==on &&
 THR1==on && ES2==off && SS2==off && RW2==on && GPS2==coarse &&
 STR2==on && THR2==on && PLI1==off && PLI2!=off)
{u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;PLI2=off;CPC2=idle;
FPC2=run;m=10;u_e_MM2=low;
error_recovery(ES2,SS2,GPS2,STR2,RW2,THR2,PLI2,CPC2,FPC2,m,u_e_MM2);}
else if (ES1==off && SS1==off && RW1==on && GPS1==coarse && STR1==on &&
 THR1==on && ES2==off && SS2==off && RW2==on && GPS2==coarse &&
 STR2==on && THR2==on && (PLI1!=off || PLI2!=off))
{u_e_MM1=high;u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;PLI1=off;CPC1=idle;
FPC1=run;ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;PLI2=off;
m=7;u_e_MM1=low;u_e_MM2=low;CPC2=idle;FPC2=run;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"\nError recovered in Nominal Mode";
cout<<"\t\tError recovered in Nominal Mode";}
else if ((ES1!=off || SS1!=off || RW1!=on || ES2!=off || SS2!=off ||
 RW2!=on) && GPS1==coarse && STR1==on && THR1==on && PLI1==off
 && GPS2==coarse && STR2==on && THR2==on && PLI2==off)
{u_e_MM1=high;u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
```

```
ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;PLI1=off;
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;PLI2=off;
m=8;u_e_MM1=low;u_e_MM2=low;CPC1=idle;FPC1=idle;CPC2=idle;FPC2=idle;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
m=2;operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,
THR2,PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;THR1=off;PLI1=off;
ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;THR2=off;PLI2=off;
m=3;CPC1=run;FPC1=idle;CPC2=run;FPC2=idle;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;PLI1=off;
ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;PLI2=off;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"\nError recovered in Nominal Mode";
cout<<"\t\tError recovered in Nominal Mode";}
else if (ES1==off && SS1==off && RW1==on && (GPS1!=coarse || STR1!=on
 || THR1!=on || GPS2!=coarse || STR2!=on || THR2!=on) &&
 PLI1==off && ES2==off && SS2==off && RW2==on &&  PLI2==off)
{u_e_MM1=high;u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;THR1=off;PLI1=off;
ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;THR2=off;PLI2=off;
CPC1=run;FPC1=idle;CPC2=run;FPC2=idle;m=9;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;PLI1=off;
ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;PLI2=off;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"\nError recovered in Nominal Mode";
cout<<"\t\tError recovered in Nominal Mode";}
else if ((ES1!=off || SS1!=off || RW1!=on || ES2!=off || SS2!=off ||
 RW2!=on)&&(GPS1!=coarse || STR1!=on || THR1!=on ||GPS2!=coarse
 || STR2!=on || THR2!=on) && ( PLI1!=off || PLI2!=off))
{u_e_MM1=high;u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
```

```
cout<<"Multiple unit errors occur";
cout<<"\t\tMultiple unit errors occur";
cout<<"\nTimeout !!!";cout<<"\t\tTimeout !!!"; goto stop;}
else cout<<"\n\n";}
else if(mode==5)//preparation
{ES1=off;SS1=off;RW1=on; ES2=off;SS2=off;RW2=on;
GPS1=fine;STR1=on;THR1=on; GPS2=fine;STR2=on;THR2=on;
PLI1=Standby; PLI2=off;
CPC1=idle;FPC1=run; CPC2=idle;FPC2=run;
u_e_MM1=low;u_e_MM2=low;c_e_MM1=low;c_e_MM2=low;
if( (ES1==off && SS1==off && RW1==on && GPS1==fine && STR1==on &&
   THR1==on && PLI1==Standby) && (ES2==off && SS2==off && RW2==on &&
   GPS2==fine && STR2==on && THR2==on && PLI2==Standby))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if ((ES1!=off || SS1!=off || RW1!=on) && GPS1==fine && STR1==on &&
 THR1==on && PLI1==Standby && ES2==off && SS2==off && RW2==on
 && GPS2==fine && STR2==on && THR2==on && PLI2==Standby)
{u_e_MM1=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;PLI1=off;CPC1=idle;
FPC1=idle;m=11;u_e_MM1=low;
error_recovery(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,CPC1,FPC1,m,u_e_MM1);}
else if ((ES2!=off || SS2!=off || RW2!=on) && GPS1==fine && STR1==on &&
 THR1==on && PLI1==Standby && ES1==off && SS1==off && RW1==on
 && GPS2==fine && STR2==on && THR2==on && PLI2==Standby)
{u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;PLI2=off;CPC2=idle;
FPC2=idle;m=12;u_e_MM2=low;
error_recovery(ES2,SS2,GPS2,STR2,RW2,THR2,PLI2,CPC2,FPC2,m,u_e_MM2);}
else if (ES1==off && SS1==off && RW1==on && (GPS1!=fine || STR1!=on ||
 THR1!=on) && PLI1==Standby && ES2==off && SS2==off && RW2==on
 && GPS2==fine && STR2==on && THR2==on && PLI2==Standby)
{u_e_MM1=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;THR1=off;PLI1=off;CPC1=run;
FPC1=idle;m=13;u_e_MM1=low;
error_recovery(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,CPC1,FPC1,m,u_e_MM1);}
else if (ES1==off && SS1==off && RW1==on && (GPS2!=fine || STR2!=on ||
```

```
 THR2!=on) && PLI1==Standby && ES2==off && SS2==off && RW2==on
 && GPS1==fine && STR1==on && THR1==on && PLI2==Standby)
{u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;THR2=off;PLI2=off;CPC2=run;
FPC2=idle;m=14;u_e_MM2=low;
error_recovery(ES2,SS2,GPS2,STR2,RW2,THR2,PLI2,CPC2,FPC2,m,u_e_MM2);}
else if (ES1==off && SS1==off && RW1==on && GPS1==fine && STR1==on &&
 THR1==on && ES2==off && SS2==off && RW2==on && GPS2==fine &&
 STR2==on && THR2==on && PLI1!=Standby && PLI2==Standby)
{u_e_MM1=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;PLI1=off;CPC1=idle;
FPC1=run;m=15;u_e_MM1=low;
error_recovery(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,CPC1,FPC1,m,u_e_MM1);}
else if (ES1==off && SS1==off && RW1==on && GPS1==fine && STR1==on &&
 THR1==on && ES2==off && SS2==off && RW2==on && GPS2==fine &&
 STR2==on && THR2==on && PLI1==Standby && PLI2!=Standby)
{u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;PLI2=off;CPC2=idle;
FPC2=run;m=16;u_e_MM2=low;
error_recovery(ES2,SS2,GPS2,STR2,RW2,THR2,PLI2,CPC2,FPC2,m,u_e_MM2);}
else if (ES1==off && SS1==off && RW1==on && GPS1==fine && STR1==on &&
 THR1==on && ES2==off && SS2==off && RW2==on && GPS2==fine &&
 STR2==on && THR2==on && (PLI1!=Standby || PLI2!=Standby))
{u_e_MM1=high;u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;PLI1=off;
ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;PLI2=off;
m=7;u_e_MM1=low;u_e_MM2=low;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;PLI1=Standby;
ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;PLI2=Standby;
u_e_MM1=low;u_e_MM2=low;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
```

```
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"\nError recovered in Preparation Mode";
cout<<"\tError recovered in Preparation Mode";}
else if ((ES1!=off || SS1!=off || RW1!=on || ES2!=off || SS2!=off ||
 RW2!=on) && GPS1==fine && STR1==on && THR1==on && GPS2==fine
 && PLI1==Standby  && STR2==on && THR2==on && PLI2==Standby)
{u_e_MM1=high;u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;PLI1=off;
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;PLI2=off;
m=8;u_e_MM1=low;u_e_MM2=low;CPC1=idle;FPC1=idle;CPC2=idle;FPC2=idle;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
m=2;operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,
THR2,PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;THR1=off;PLI1=off;
ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;THR2=off;PLI2=off;
m=3;CPC1=run;FPC1=idle;CPC2=run;FPC2=idle;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;PLI1=off;
ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;PLI2=off;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;m=4;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;PLI1=Standby;
ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;PLI2=Standby;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"\nError recovered in Preparation Mode";
cout<<"\tError recovered in Preparation Mode";}
else if (ES1==off && SS1==off && RW1==on && (GPS1!=fine || STR1!=on ||
 THR1!=on || GPS2!=fine || STR2!=on || THR2!=on) && RW2==on
 && PLI1==Standby && ES2==off && SS2==off  && PLI2==Standby)
{u_e_MM1=high;u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;THR1=off;PLI1=off;
ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;THR2=off;PLI2=off;
CPC1=run;FPC1=idle;CPC2=run;FPC2=idle;m=9;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
```

```
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;PLI1=off;
ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;PLI2=off;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;m=4;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;PLI1=Standby;
ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;PLI2=Standby;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"\nError recovered in Preparation Mode";
cout<<"\tError recovered in Preparation Mode";}
else if ((ES1!=off || SS1!=off || RW1!=on || ES2!=off || SS2!=off ||
 RW2!=on) && (GPS1!=fine || STR1!=on || THR1!=on || GPS2!=fine
 || STR2!=on || THR2!=on) && ( PLI1!=Standby ||  PLI2!=Standby))
{u_e_MM1=high;u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"Multiple unit errors occur";
cout<<"\t\tMultiple unit errors occur";
cout<<"\nTimeout !!!";cout<<"\t\tTimeout !!!"; goto stop;}
else cout<<"\n\n"; }
else if (mode==6) //science
{ES1=off;SS1=off;RW1=on; ES2=off;SS2=on;RW2=on;
GPS1=fine;STR1=off;THR1=on; GPS2=fine;STR2=on;THR2=off;
PLI1=off; PLI2=on;
CPC1=idle;FPC1=run; CPC2=idle;FPC2=run;
u_e_MM1=low;u_e_MM2=low;c_e_MM1=low;c_e_MM2=low;
if(ES1==off && SS1==off && RW1==on && GPS1==fine && STR1==on &&
   THR1==on && PLI1==Science && ES2==off && SS2==off && RW2==on &&
   GPS2==fine && STR2==on && THR2==on && PLI2==Science)
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if ((ES1!=off || SS1!=off || RW1!=on) && GPS1==fine && STR1==on &&
 THR1==on && PLI1==Science && ES2==off && SS2==off && RW2==on
 && GPS2==fine && STR2==on && THR2==on && PLI2==Science)
{u_e_MM1=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;PLI1=off;CPC1=idle;
FPC1=idle;m=17;u_e_MM1=low;
error_recovery(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,CPC1,FPC1,m,u_e_MM1);}
```

```
else if ((ES2!=off || SS2!=off || RW2!=on) && GPS1==fine && STR1==on &&
 THR1==on && PLI1==Science && ES1==off && SS1==off && RW1==on
 && GPS2==fine && STR2==on && THR2==on && PLI2==Science)
{u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;PLI2=off;CPC2=idle;
FPC2=idle;m=18;u_e_MM2=low;
error_recovery(ES2,SS2,GPS2,STR2,RW2,THR2,PLI2,CPC2,FPC2,m,u_e_MM2);}
else if (ES1==off && SS1==off && RW1==on && (GPS1!=fine || STR1!=on ||
 THR1!=on) && PLI1==Science && ES2==off && SS2==off && RW2==on
 && GPS2==fine && STR2==on && THR2==on && PLI2==Science)
{u_e_MM1=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;THR1=off;PLI1=off;CPC1=run;
FPC1=idle;m=19;u_e_MM1=low;
error_recovery(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,CPC1,FPC1,m,u_e_MM1);}
else if (ES1==off && SS1==off && RW1==on && (GPS2!=fine || STR2!=on ||
 THR2!=on) && PLI1==Science && ES2==off && SS2==off && RW2==on
 && GPS1==fine && STR1==on && THR1==on && PLI2==Science)
{u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;THR2=off;PLI2=off;CPC2=run;
FPC2=idle;m=20;u_e_MM2=low;
error_recovery(ES2,SS2,GPS2,STR2,RW2,THR2,PLI2,CPC2,FPC2,m,u_e_MM2);}
else if (ES1==off && SS1==off && RW1==on && GPS1==fine && STR1==on &&
 THR1==on && ES2==off && SS2==off && RW2==on && GPS2==fine &&
 STR2==on && THR2==on && PLI1!=Science && PLI2==Science)
{u_e_MM1=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;PLI1=off;CPC1=idle;
FPC1=run;m=21;u_e_MM1=low;
error_recovery(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,CPC1,FPC1,m,u_e_MM1);}
else if (ES1==off && SS1==off && RW1==on && GPS1==fine && STR1==on &&
 THR1==on && ES2==off && SS2==off && RW2==on && GPS2==fine &&
 STR2==on && THR2==on && PLI1==Science && PLI2!=Science)
{u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;PLI2=off;CPC2=idle;
```

```
FPC2=run;m=22;u_e_MM2=low;
error_recovery(ES2,SS2,GPS2,STR2,RW2,THR2,PLI2,CPC2,FPC2,m,u_e_MM2);}
else if ((ES1!=off || SS1!=off || RW1!=on || ES2!=off || SS2!=off ||
 RW2!=on) && GPS1==fine && STR1==on && THR1==on && THR2==on &&
 PLI1==Science && GPS2==fine && STR2==on && PLI2==Science)
{u_e_MM1=high;u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;PLI1=off;
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;PLI2=off;
m=8;u_e_MM1=low;u_e_MM2=low;CPC1=idle;FPC1=idle;CPC2=idle;FPC2=idle;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
m=2;operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,
THR2,PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;THR1=off;PLI1=off;
ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;THR2=off;PLI2=off;
CPC1=run;FPC1=idle;CPC2=run;FPC2=idle;m=3;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;PLI1=off;
ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;PLI2=off;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;m=4;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;PLI1=Standby;
ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;PLI2=Standby;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;m=5;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;PLI1=Science;
ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;PLI2=Science;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"\nError recovered in Science Mode";
cout<<"\t\tError recovered in Science Mode";}
else if (ES1==off && SS1==off && RW1==on && (GPS1!=fine || STR1!=on ||
 THR1!=on || GPS2!=fine || STR2!=on || THR2!=on) && RW2==on &&
 PLI1==Science && ES2==off && SS2==off && PLI2==Science)
{u_e_MM1=high;u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
```

```
ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;THR1=off;PLI1=off;
ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;THR2=off;PLI2=off;
CPC1=run;FPC1=idle;CPC2=run;FPC2=idle;m=9;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;PLI1=off;
ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;PLI2=off;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;m=4;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;PLI1=Standby;
ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;PLI2=Standby;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;m=5;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;PLI1=Science;
ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;PLI2=Science;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;u_e_MM1=low;u_e_MM2=low;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"\nError recovered in Science Mode";
cout<<"\t\tError recovered in Science Mode";}
else if (ES1==off && SS1==off && RW1==on && GPS1==fine && STR1==on &&
 THR1==on && ES2==off && SS2==off && RW2==on && GPS2==fine &&
 STR2==on && THR2==on && (PLI1!=Science || PLI2!=Science))
{u_e_MM1=high;u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;PLI1=off;
ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;PLI2=off;
m=7;u_e_MM1=low;u_e_MM2=low;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;PLI1=Standby;
ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;PLI2=Standby;
m=5;u_e_MM1=low;u_e_MM2=low;
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,m,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;PLI1=Science;
ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;PLI2=Science;
u_e_MM1=low;u_e_MM2=low;
```

```
CPC1=idle;FPC1=run;CPC2=idle;FPC2=run;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"\nError recovered in Science Mode";
cout<<"\t\tError recovered in Science Mode";}
else if ((ES1!=off || SS1!=off || RW1!=on || ES2!=off || SS2!=off ||
 RW2!=on) && (GPS1!=fine || STR1!=on || THR1!=on || GPS2!=fine
 || STR2!=on || THR2!=on) && ( PLI1!=Science ||  PLI2!=Science))
{u_e_MM1=high;u_e_MM2=high;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"Multiple unit errors occur";
cout<<"\t\tMultiple unit errors occur";cout<<"\nTimeout !!!";
cout<<"\t\t\t\tTimeout !!!";cout<<"\nProgram Terminated....";goto stop;
}else cout<<"\n";}
else {cout<<"Program Terminated\n"; goto stop;}}}
else if (ii==2)// Nominal Branch Errors
{cout<<"Note:\n";
cout<<" '0' denotes 'Off state' & 'Idle phase'\n";
cout<<" '1' denotes 'On state' & 'Running phase'\n";
cout<<" '2' denotes 'Coarse state'\n";
cout<<" '3' denotes 'Fine state'\n";
cout<<" '4' denotes 'Standby state'\n";
cout<<" '5' denotes 'Science state'\n\n";}
else if (ii<=0 || ii>=3)
{cout<<"Program Terminated\n\n\n";goto stop;}}
else if(i==2)// Controller Phase Errors
{cout<<"Note:\n";
cout<<" '0' denotes 'Off state' & 'Idle phase'\n";
cout<<" '1' denotes 'On state' & 'Running phase'\n";
cout<<" '2' denotes 'Coarse state'\n";
cout<<" '3' denotes 'Fine state'\n";
cout<<" '4' denotes 'Standby state'\n";
cout<<" '5' denotes 'Science state'\n\n";
for (int mode = 1; mode < 7; mode++)
{u_e_MM1=low;u_e_MM2=low;
if(mode==1)
{CPC1=idle;FPC1=idle; CPC2=idle;FPC2=idle;
c_e_MM1=low;c_e_MM2=low;
ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;PLI1=off;
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;PLI2=off;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
```

```
else if(mode==2)
{CPC1=idle;FPC1=idle; CPC2=idle;FPC2=idle;
c_e_MM1=low;c_e_MM2=low;
ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;PLI1=off;
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;PLI2=off;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if(mode==3)
{CPC1=idle;FPC1=run; CPC2=run;FPC2=idle;
ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;THR1=off;PLI1=off;
ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;THR2=off;PLI2=off;
{if(CPC1==run && FPC1==idle)c_e_MM1=low;
else c_e_MM1=high;
if(CPC2==run && FPC2==idle)c_e_MM2=low;
else c_e_MM2=high;}
if(CPC1==run && FPC1==idle && CPC2==run && FPC2==idle)
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if (CPC1==run && FPC1==run && CPC2==run && FPC2==run)
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"Both controllers run in MM1";
cout<<"\t\tBoth controllers run in MM2";
cout<<"\nTimeout !!!";cout<<"\t\t\t\tTimeout !!!";
cout<<"\nProgram terminated"<<endl;goto stop;}
else if (CPC1==run && FPC1==run && ((CPC2==run && FPC2==idle)||
 (CPC2==idle && FPC2==run) || (CPC2==idle && FPC2==idle)))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"Both controllers run in MM1";cout<<"\nTimeout !!!";
cout<<"\nProgram terminated"<<endl; goto stop;}
else if (((CPC1==run && FPC1==idle)||(CPC1==idle && FPC1==run)||
 (CPC1==idle && FPC1==idle)) && CPC2==run && FPC2==run)
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"\t\t\t\t\tBoth controllers run in MM2";
cout<<"\n\t\t\t\t\t\tTimeout !!!"; cout<<"\nProgram terminated"<<endl;}
else if ((CPC1==idle && FPC1==run && CPC2==idle && FPC2==run)||
 (CPC1==idle && FPC1==run && CPC2==idle && FPC2==idle)||
 (CPC1==idle && FPC1==idle && CPC2==idle && FPC2==run)||
 (CPC1==idle && FPC1==idle && CPC2==idle && FPC2==idle))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
```

```
ES1=off;SS1=off;GPS1=off;STR1=off;RW1=off;THR1=off;PLI1=off;CPC1=idle;
FPC1=idle;c_e_MM1=low; FPC2=idle;c_e_MM2=low;
ES2=off;SS2=off;GPS2=off;STR2=off;RW2=off;THR2=off;PLI2=off;CPC2=idle;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,8,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,2,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;THR1=off;PLI1=off;CPC1=run;
FPC1=idle;c_e_MM1=low; FPC2=idle;c_e_MM2=low;
ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;THR2=off;PLI2=off;CPC2=run;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if ((CPC1==idle && FPC1==idle && CPC2==run && FPC2==idle) ||
 (CPC1==idle && FPC1==run && CPC2==run && FPC2==idle))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
CPC1=idle;FPC1=idle;c_e_MM1=low;ES1=off;SS1=off;GPS1=off;STR1=off;
RW1=off;THR1=off;PLI1=off;
error_recovery(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,CPC1,FPC1,1,c_e_MM1);}
else if ((CPC1==run && FPC1==idle && CPC2==idle && FPC2==idle) ||
 (CPC1==run && FPC1==idle && CPC2==idle && FPC2==run))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
CPC2=idle;FPC2=idle;c_e_MM2=low;ES2=off;SS2=off;GPS2=off;STR2=off;
RW2=off;THR2=off;PLI2=off;
error_recovery(ES2,SS2,GPS2,STR2,RW2,THR2,PLI2,CPC2,FPC2,2,c_e_MM2);}}
else if(mode==4)
{CPC1=run;FPC1=idle; CPC2=run;FPC2=idle;
ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;PLI1=off;
ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;PLI2=off;
{if(CPC1==idle && FPC1==run)c_e_MM1=low;
else c_e_MM1=high;
if(CPC2==idle && FPC2==run)c_e_MM2=low;
else c_e_MM2=high;}
if(CPC1==idle && FPC1==run && CPC2==idle && FPC2==run)
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if (CPC1==run && FPC1==run && CPC2==run && FPC2==run)
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"Both controllers run in MM1";
cout<<"\t\tBoth controllers run in MM2";
cout<<"\nTimeout !!!";cout<<"\t\t\t\tTimeout !!!";
```

```
cout<<"\nProgram terminated"<<endl;goto stop;}
else if (CPC1==run && FPC1==run && ((CPC2==run && FPC2==idle)||
 (CPC2==idle && FPC2==run) || (CPC2==idle && FPC2==idle)))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"Both controllers run in MM1";cout<<"\nTimeout !!!";
cout<<"\nProgram terminated"<<endl; goto stop;}
else if (((CPC1==run && FPC1==idle)||(CPC1==idle && FPC1==run)||
 (CPC1==idle && FPC1==idle)) && CPC2==run && FPC2==run)
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"\t\t\t\t\tBoth controllers run in MM2";
cout<<"\n\t\t\t\t\tTimeout !!!"; cout<<"\nProgram terminated"<<endl;}
else if ((CPC1==run && FPC1==idle && CPC2==idle && FPC2==idle)||
 (CPC1==run && FPC1==idle && CPC2==run && FPC2==idle)||
 (CPC1==idle && FPC1==idle && CPC2==run && FPC2==idle) ||
 (CPC1==idle && FPC1==idle && CPC2==idle && FPC2==idle))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;THR1=off;PLI1=off;CPC1=run;
FPC1=idle;c_e_MM1=low; FPC2=idle;c_e_MM2=low;
ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;THR2=off;PLI2=off;CPC2=run;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,9,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;PLI1=off;CPC1=idle;
FPC1=run;c_e_MM1=low; FPC2=run;c_e_MM2=low;
ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;PLI2=off;CPC2=idle;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if ((CPC1==run && FPC1==idle && CPC2==idle && FPC2==run)||
 (CPC1==idle && FPC1==idle && CPC2==idle && FPC2==run))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
CPC1=run;FPC1=idle;c_e_MM1=low;ES1=on;SS1=on;GPS1=off;STR1=off;RW1=on;
THR1=off;PLI1=off;
error_recovery(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,CPC1,FPC1,7,c_e_MM1);}
else if((CPC1==idle && FPC1==run && CPC2==run && FPC2==idle) ||
(CPC1==idle && FPC1==run && CPC2==idle && FPC2==idle))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
CPC2=run;FPC2=idle;c_e_MM2=low;ES2=on;SS2=on;GPS2=off;STR2=off;RW2=on;
THR2=off;PLI2=off;
error_recovery(ES2,SS2,GPS2,STR2,RW2,THR2,PLI2,CPC2,FPC2,8,c_e_MM2);}}
```

```
else if(mode==5)
{CPC1=idle;FPC1=run; CPC2=run;FPC2=idle;
c_e_MM1=low;c_e_MM2=low;
ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;PLI1=Standby;
ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;PLI2=Standby;
{if(CPC1==idle && FPC1==run)c_e_MM1=low;
else c_e_MM1=high;
if(CPC2==idle && FPC2==run)c_e_MM2=low;
else c_e_MM2=high;}
if(CPC1==idle && FPC1==run && CPC2==idle && FPC2==run)
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if (CPC1==run && FPC1==run && CPC2==run && FPC2==run)
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"Both controllers run in MM1";
cout<<"\t\tBoth controllers run in MM2";
cout<<"\nTimeout !!!";cout<<"\t\t\t\tTimeout !!!";
cout<<"\nProgram terminated"<<endl;goto stop;}
else if (CPC1==run && FPC1==run && ((CPC2==run && FPC2==idle)||
 (CPC2==idle && FPC2==run) || (CPC2==idle && FPC2==idle)))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"Both controllers run in MM1";cout<<"\nTimeout !!!";
cout<<"\nProgram terminated"<<endl; goto stop;}
else if (((CPC1==run && FPC1==idle)||(CPC1==idle && FPC1==run)||
 (CPC1==idle && FPC1==idle)) && CPC2==run && FPC2==run)
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"\t\t\t\tBoth controllers run in MM2";
cout<<"\n\t\t\t\t\tTimeout !!!"; cout<<"\nProgram terminated"<<endl;}
else if ((CPC1==run && FPC1==idle && CPC2==idle && FPC2==idle) ||
 (CPC1==run && FPC1==idle && CPC2==run && FPC2==idle) ||
 (CPC1==idle && FPC1==idle && CPC2==run && FPC2==idle) ||
 (CPC1==idle && FPC1==idle && CPC2==idle && FPC2==idle))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=coarse;STR1=on;RW1=on;THR1=on;PLI1=off;CPC1=idle;
FPC1=run;c_e_MM1=low; FPC2=run;c_e_MM2=low;
ES2=off;SS2=off;GPS2=coarse;STR2=on;RW2=on;THR2=on;PLI2=off;CPC2=idle;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,7,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;PLI1=Standby;
```

```
CPC1=idle;FPC1=run;c_e_MM1=low;CPC2=idle;FPC2=run;c_e_MM2=low;
ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;PLI2=Standby;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if ((CPC1==run && FPC1==idle && CPC2==idle && FPC2==run)||
 (CPC1==idle && FPC1==idle && CPC2==idle && FPC2==run))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
CPC1=idle;FPC1=run;c_e_MM1=low;ES1=off;SS1=off;GPS1=coarse;STR1=on;
RW1=on;THR1=on;PLI1=off;
error_recovery(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,CPC1,FPC1,15,c_e_MM1);}
else if((CPC1==idle && FPC1==run && CPC2==run && FPC2==idle) ||
(CPC1==idle && FPC1==run && CPC2==idle && FPC2==idle))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
CPC2=idle;FPC2=run;c_e_MM2=low;ES2=off;SS2=off;GPS2=coarse;STR2=on;
RW2=on;THR2=on;PLI2=off;
error_recovery(ES2,SS2,GPS2,STR2,RW2,THR2,PLI2,CPC2,FPC2,16,c_e_MM2);}}
else if(mode==6)
{CPC1=run;FPC1=run; CPC2=idle;FPC2=run;
c_e_MM1=low;c_e_MM2=low;
ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;PLI1=Science;
ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;PLI2=Science;
{if(CPC1==idle && FPC1==run)c_e_MM1=low;
else c_e_MM1=high;
if(CPC2==idle && FPC2==run)c_e_MM2=low;
else c_e_MM2=high;}
if(CPC1==idle && FPC1==run && CPC2==idle && FPC2==run)
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if (CPC1==run && FPC1==run && CPC2==run && FPC2==run)
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"Both controllers run in MM1";
cout<<"\t\tBoth controllers run in MM2";
cout<<"\nTimeout !!!";cout<<"\t\t\t\tTimeout !!!";
cout<<"\nProgram terminated"<<endl;goto stop;}
else if (CPC1==run && FPC1==run && ((CPC2==run && FPC2==idle)||
 (CPC2==idle && FPC2==run) || (CPC2==idle && FPC2==idle)))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"Both controllers run in MM1";cout<<"\nTimeout !!!";
cout<<"\nProgram terminated"<<endl; goto stop;}
```

```
else if (((CPC1==run && FPC1==idle)||(CPC1==idle && FPC1==run)||
 (CPC1==idle && FPC1==idle)) && CPC2==run && FPC2==run)
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
cout<<"\t\t\t\t\tBoth controllers run in MM2";
cout<<"\n\t\t\t\t\tTimeout !!!"; cout<<"\nProgram terminated"<<endl;}
else if ((CPC1==run && FPC1==idle && CPC2==idle && FPC2==idle)||
 (CPC1==run && FPC1==idle && CPC2==run && FPC2==idle) ||
 (CPC1==idle && FPC1==idle && CPC2==run && FPC2==idle) ||
 (CPC1==idle && FPC1==idle && CPC2==idle && FPC2==idle))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;PLI1=Standby;
CPC1=idle;FPC1=run;c_e_MM1=low;CPC2=idle;FPC2=run;c_e_MM2=low;
ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;PLI2=Standby;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,10,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
ES1=off;SS1=off;GPS1=fine;STR1=on;RW1=on;THR1=on;PLI1=Science;
CPC1=idle;FPC1=run;c_e_MM1=low;CPC2=idle;FPC2=run;c_e_MM2=low;
ES2=off;SS2=off;GPS2=fine;STR2=on;RW2=on;THR2=on;PLI2=Science;
operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if ((CPC1==run && FPC1==idle && CPC2==idle && FPC2==run)||
 (CPC1==idle && FPC1==idle && CPC2==idle && FPC2==run))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
CPC1=idle;FPC1=run;c_e_MM1=low;ES1=off;SS1=off;GPS1=fine;STR1=on;
RW1=on;THR1=on;PLI1=Standby;
error_recovery(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,CPC1,FPC1,25,c_e_MM1);}
else if((CPC1==idle && FPC1==run && CPC2==run && FPC2==idle) ||
(CPC1==idle && FPC1==run && CPC2==idle && FPC2==idle))
{operation(ES1,SS1,GPS1,STR1,RW1,THR1,PLI1,ES2,SS2,GPS2,STR2,RW2,THR2,
PLI2,CPC1,FPC1,CPC2,FPC2,mode,u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);
CPC2=idle;FPC2=run;c_e_MM2=low;ES2=off;SS2=off;GPS2=fine;STR2=on;
RW2=on;THR2=on;PLI2=Standby;
error_recovery(ES2,SS2,GPS2,STR2,RW2,THR2,PLI2,CPC2,FPC2,26,c_e_MM2);}}
else cout<<"Program Terminated\n\n\n"; } }
else if (i<=0 || i>=3){cout<<"Program Terminated\n\n\n";goto stop;}
else {cout<<"Program Terminated\n\n\n";goto stop;} }
else if (in<=0 || in>=3){cout<<"Program Terminated\n\n\n";goto stop;}
else {cout<<"Program Terminated\n\n\n";goto stop;}
stop:cout<<"\n\n\t\t  ";
SetConsoleTextAttribute(hOut, BACKGROUND_RED | BACKGROUND_GREEN |
```

```
BACKGROUND_BLUE | BACKGROUND_INTENSITY);
printf(" Welcome to the Distributed System of AOCS");
SetConsoleTextAttribute(hOut,FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
cout<<"\n\n\nDo you want to continue this program"<<"\n";
cout<<"Press 'Y' for Yes"<<"\n"<<"Press 'N' for No"<<endl;
char bb;cin >> bb;cout<<endl;
if (bb == 'y' || bb == 'Y')b = true;
else{cout<<"Program terminated"<<endl;b = false;}
if ( c == 'n') cout<<"Program terminated"<<endl;}
system("PAUSE");return 0;}
void controller_phase(int CPC1,int FPC1,int CPC2,int FPC2)
{int idle=0;int run=1;
if(CPC1==idle && FPC1==idle && CPC2==idle && FPC2==idle)
{cout<<"\n CPC = "<<CPC1;cout<<"\t\t\t\t CPC = "<<CPC2;
cout<<"\n FPC = "<<FPC1;cout<<"\t\t\t\t FPC = "<<FPC2;}
else if (CPC1==run && FPC1==idle && CPC2==idle && FPC2==idle)
{cout<<"\n FPC = "<<FPC1;cout<<"\t\t\t\t CPC = "<<CPC2;
cout<<"\nCPC is preparing now";cout<<"\t\t\t FPC = "<<FPC2<<endl;
Sleep(AOCS_cycles);cout<<"CPC is set to Ready";cout<<"\n CPC = "<<CPC1;
}else if (CPC1==run && FPC1==idle && CPC2==run && FPC2==run)
{cout<<"\n FPC = "<<FPC1;cout<<"\t\t\t\tCPC is preparing now";
cout<<"\nCPC is preparing now"; Sleep(AOCS_cycles);
cout<<"\t\t\tCPC is set to Ready";cout<<"\nCPC is set to Ready";
cout<<"\t\t\t CPC = "<<CPC2;cout<<"\n CPC = "<<CPC1;
cout<<"\t\t\t\tFPC is preparing now";
cout<<"\n\t\t\t\t\t\tFPC is set to Ready";
cout<<"\n\t\t\t\t\t\t FPC = "<<FPC2;}
else if (CPC1==run && FPC1==idle && CPC2==run && FPC2==idle)
{cout<<"\n FPC = "<<FPC1;cout<<"\t\t\t\t FPC = "<<FPC2;
cout<<"\nCPC is preparing now";cout<<"\t\t\tCPC is preparing now";
Sleep(AOCS_cycles);cout<<"\nCPC is set to Ready";
cout<<"\t\t\tCPC is set to Ready";
cout<<"\n CPC = "<<CPC1;cout<<"\t\t\t\t CPC = "<<CPC2;}
else if (CPC1==run && FPC1==idle && CPC2==idle && FPC2==run)
{cout<<"\n FPC = "<<FPC1;cout<<"\t\t\t\t CPC = "<<CPC2;
cout<<"\nCPC is preparing now";cout<<"\t\t\tFPC is preparing now";
Sleep(AOCS_cycles);cout<<"\nCPC is set to Ready";
cout<<"\t\t\tFPC is set to Ready";
cout<<"\n CPC = "<<CPC1;cout<<"\t\t\t\t FPC = "<<FPC2;}
else if (CPC1==idle && FPC1==run && CPC2==idle && FPC2==idle)
{cout<<"\n CPC = "<<CPC1;cout<<"\t\t\t\t CPC = "<<CPC2;
cout<<"\nFPC is preparing now";cout<<"\t\t\t FPC = "<<FPC2<<endl;
```

```
Sleep(AOCS_cycles);
cout<<"FPC is set to Ready";cout<<"\n FPC = "<<FPC1;}
else if (CPC1==idle && FPC1==run && CPC2==run && FPC2==run)
{cout<<"\n CPC = "<<CPC1; cout<<"\t\t\t\tCPC is preparing now";
cout<<"\nFPC is preparing now"; Sleep(AOCS_cycles);
cout<<"\t\t\tCPC is set to Ready";
cout<<"\nFPC is set to Ready";cout<<"\t\t\t CPC = "<<CPC2;
cout<<"\n FPC = "<<FPC1;cout<<"\t\t\t\tFPC is preparing now";
cout<<"\n\t\t\t\t\tFPC is set to Ready";
cout<<"\n\t\t\t\t\t FPC = "<<FPC2;}
else if (CPC1==idle && FPC1==run && CPC2==run && FPC2==idle)
{cout<<"\n CPC = "<<CPC1;cout<<"\t\t\t\t FPC = "<<FPC2;
cout<<"\nFPC is preparing now";cout<<"\t\t\tCPC is preparing now";
Sleep(AOCS_cycles);cout<<"\nFPC is set to Ready";
cout<<"\t\t\tCPC is set to Ready";
cout<<"\n FPC = "<<FPC1;cout<<"\t\t\t\t CPC = "<<CPC2;}
else if (CPC1==idle && FPC1==run && CPC2==idle && FPC2==run)
{cout<<"\n CPC = "<<CPC1;cout<<"\t\t\t\t CPC = "<<CPC2;
cout<<"\nFPC is preparing now";cout<<"\t\t\tFPC is preparing now";
Sleep(AOCS_cycles);cout<<"\nFPC is set to Ready";
cout<<"\t\t\tFPC is set to Ready";
cout<<"\n FPC = "<<FPC1;cout<<"\t\t\t\t FPC = "<<FPC2;}
else if (CPC1==run && FPC1==run && CPC2==run && FPC2==run)
{cout<<"\nCPC is preparing now";cout<<"\t\t\tCPC is preparing now";
Sleep(AOCS_cycles);cout<<"\nCPC is set to Ready";
cout<<"\t\t\tCPC is set to Ready";
cout<<"\n CPC = "<<CPC1;cout<<"\t\t\t\t CPC = "<<CPC2;
cout<<"\nFPC is preparing now";cout<<"\t\t\tFPC is preparing now";
Sleep(AOCS_cycles);cout<<"\nFPC is set to Ready";
cout<<"\t\t\tFPC is set to Ready";
cout<<"\n FPC = "<<FPC1;cout<<"\t\t\t\t FPC = "<<FPC2;}
else if (CPC1==run && FPC1==run && CPC2==idle && FPC2==idle)
{cout<<"\nCPC is preparing now";cout<<"\t\t\t CPC = "<<CPC2;
cout<<"\n\t\t\t\t\t FPC = "<<FPC2;Sleep(AOCS_cycles);
cout<<"\nCPC is set to Ready";cout<<"\n CPC = "<<CPC1;
cout<<"\nFPC is preparing now";Sleep(AOCS_cycles);
cout<<"\nFPC is set to Ready";cout<<"\n FPC = "<<FPC1;}
else if (CPC1==run && FPC1==run && CPC2==run && FPC2==idle)
{cout<<"\nCPC is preparing now";cout<<"\t\t\tCPC is preparing now";
Sleep(AOCS_cycles);cout<<"\nCPC is set to Ready";
cout<<"\t\t\tCPC is set to Ready";cout<<"\n CPC = "<<CPC1;
cout<<"\t\t\t\t CPC = "<<CPC2;cout<<"\nFPC is preparing now";
cout<<"\t\t\t FPC = "<<FPC2;Sleep(AOCS_cycles);
```

```
cout<<"\nFPC is set to Ready";cout<<"\n FPC = "<<FPC1;}
else if (CPC1==run && FPC1==run && CPC2==idle && FPC2==run)
{cout<<"\nCPC is preparing now";cout<<"\t\t\tFPC is preparing now";
Sleep(AOCS_cycles);cout<<"\nCPC is set to Ready";
cout<<"\t\t\tFPC is set to Ready";cout<<"\n CPC = "<<CPC1;
cout<<"\t\t\t\t FPC = "<<FPC2;
cout<<"\nFPC is preparing now";cout<<"\t\t\t CPC = "<<CPC2;
Sleep(AOCS_cycles);cout<<"\nFPC is set to Ready";
cout<<"\n FPC = "<<FPC1;}
else if (CPC1==idle && FPC1==idle && CPC2==run && FPC2==run)
{cout<<"\n CPC = "<<CPC1;cout<<"\t\t\t\tCPC is preparing now";
cout<<"\n FPC = "<<FPC1;Sleep(AOCS_cycles);
cout<<"\t\t\t\tCPC is set to Ready";cout<<"\n\t\t\t\t\t\t CPC = "<<CPC2;
cout<<"\n\t\t\t\t\tFPC is preparing now\n"; Sleep(AOCS_cycles);
cout<<"\t\t\t\t\tFPC is set to Ready";
cout<<"\n\t\t\t\t\t FPC = "<<FPC2;}
else if (CPC1==idle && FPC1==idle && CPC2==idle && FPC2==run)
{cout<<"\n FPC = "<<FPC1;cout<<"\t\t\t\t CPC = "<<CPC2;
cout<<"\n CPC = "<<CPC1; Sleep(AOCS_cycles);
cout<<"\t\t\t\tFPC is preparing now\n";
cout<<"\t\t\t\t\tFPC is set to Ready";
cout<<"\n\t\t\t\t\t FPC = "<<FPC2;}
else if (CPC1==idle && FPC1==idle && CPC2==run && FPC2==idle)
{cout<<"\n CPC = "<<CPC1;cout<<"\t\t\t\t FPC = "<<FPC2;
cout<<"\n FPC = "<<FPC1;cout<<"\t\t\t\tCPC is preparing now\n";
Sleep(AOCS_cycles);cout<<"\t\t\t\t\tCPC is set to Ready";
cout<<"\n\t\t\t\t\t CPC = "<<CPC2;}else {cout<<"\n\n";}}
void operation(int ES1,int SS1,int GPS1,int STR1,int RW1,int THR1,
   int PLI1,int ES2,int SS2,int GPS2,int STR2,int RW2,
   int THR2,int PLI2,int CPC1,int FPC1,int CPC2,int FPC2,
   int scenario,bool u_e_MM1,bool u_e_MM2,bool c_e_MM1,
   bool c_e_MM2)
{HANDLE hOut;hOut = GetStdHandle(STD_OUTPUT_HANDLE);
if(scenario==1)
{SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"To the Off Mode of MM1";cout<<"\t\t\t";
SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"To the Off Mode of MM2\n";
SetConsoleTextAttribute(hOut,FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
cout<<" ES = "<<ES1;cout<<"\t\t\t\t\t ES = "<<ES2;
cout<<"\n SS = "<<SS1;cout<<"\t\t\t\t\t SS = "<<SS2;
cout<<"\n RW = "<<RW1;cout<<"\t\t\t\t\t RW = "<<RW2;
```

```
cout<<"\n GPS = "<<GPS1;cout<<"\t\t\t\t GPS = "<<GPS2;
cout<<"\n STR = "<<STR1;cout<<"\t\t\t\t STR = "<<STR2;
cout<<"\n THR = "<<THR1;cout<<"\t\t\t\t THR = "<<THR2;
cout<<"\n PLI = "<<PLI1;cout<<"\t\t\t\t PLI = "<<PLI2;
controller_phase(CPC1,FPC1,CPC2,FPC2);
cout<<"\n Unit Error Flag = "<<u_e_MM1;
cout<<"\t\t\t Unit Error Flag = "<<u_e_MM2;
cout<<"\n Controller Error Flag = "<<c_e_MM1;
cout<<"\t\t Controller Error Flag = "<<c_e_MM2;
handshake(u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if(scenario==2)
{SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"\n\nTo the Standby Mode of MM1";cout<<"\t\t";
SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"To the Standby Mode of MM2\n";
SetConsoleTextAttribute(hOut,FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
cout<<" ES = "<<ES1;cout<<"\t\t\t\t\t ES = "<<ES2;
cout<<"\n SS = "<<SS1;cout<<"\t\t\t\t\t SS = "<<SS2;
cout<<"\n RW = "<<RW1;cout<<"\t\t\t\t\t RW = "<<RW2;
cout<<"\n GPS = "<<GPS1;cout<<"\t\t\t\t GPS = "<<GPS2;
cout<<"\n STR = "<<STR1;cout<<"\t\t\t\t STR = "<<STR2;
cout<<"\n THR = "<<THR1;cout<<"\t\t\t\t THR = "<<THR2;
cout<<"\n PLI = "<<PLI1;cout<<"\t\t\t\t PLI = "<<PLI2;
controller_phase(CPC1,FPC1,CPC2,FPC2);
cout<<"\n Unit Error Flag = "<<u_e_MM1;
cout<<"\t\t\t Unit Error Flag = "<<u_e_MM2;
cout<<"\n Controller Error Flag = "<<c_e_MM1;
cout<<"\t\t Controller Error Flag = "<<c_e_MM2;
handshake(u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if (scenario==3)
{SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"\n\nTo the Safe Mode of MM1";cout<<"\t\t\t";
SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"To the Safe Mode of MM2\n";
SetConsoleTextAttribute(hOut,FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
cout<<" GPS = "<<GPS1;cout<<"\t\t\t\t GPS = "<<GPS2;
cout<<"\n STR = "<<STR1;;cout<<"\t\t\t\t STR = "<<STR2;
cout<<"\n THR = "<<THR1;cout<<"\t\t\t\t THR = "<<THR2;
cout<<"\n PLI = "<<PLI1;cout<<"\t\t\t\t PLI = "<<PLI2;
Sleep(2500);
cout<<"\n ES = "<<ES1;cout<<"\t\t\t\t\t ES = "<<ES2;
```

```
cout<<"\n SS = "<<SS1;cout<<"\t\t\t\t\t SS = "<<SS2;
cout<<"\n RW = "<<RW1;cout<<"\t\t\t\t\t RW = "<<RW2;
controller_phase(CPC1,FPC1,CPC2,FPC2);
cout<<"\n Unit Error Flag = "<<u_e_MM1;
cout<<"\t\t\t Unit Error Flag = "<<u_e_MM2;
cout<<"\n Controller Error Flag = "<<c_e_MM1;
cout<<"\t\t Controller Error Flag = "<<c_e_MM2;
handshake(u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if (scenario==4)
{SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"\n\nTo the Nominal Mode of MM1";cout<<"\t\t";
SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"To the Nominal Mode of MM2\n";
SetConsoleTextAttribute(hOut,FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
cout<<" RW = "<<RW1;cout<<"\t\t\t\t\t RW = "<<RW2;
cout<<"\n PLI = "<<PLI1;cout<<"\t\t\t\t PLI = "<<PLI2;
Sleep(2500);
cout<<"\n GPS = "<<GPS1;cout<<"\t\t\t\t GPS = "<<GPS2;
cout<<"\n STR = "<<STR1;cout<<"\t\t\t\t STR = "<<STR2;
cout<<"\n THR = "<<THR1;cout<<"\t\t\t\t THR = "<<THR2;
Sleep(12500);
cout<<"\n ES = "<<ES1;cout<<"\t\t\t\t\t ES = "<<ES2;
cout<<"\n SS = "<<SS1;cout<<"\t\t\t\t\t SS = "<<SS2;
controller_phase(CPC1,FPC1,CPC2,FPC2);
cout<<"\n Unit Error Flag = "<<u_e_MM1;
cout<<"\t\t\t Unit Error Flag = "<<u_e_MM2;
cout<<"\n Controller Error Flag = "<<c_e_MM1;
cout<<"\t\t Controller Error Flag = "<<c_e_MM2;
handshake(u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if (scenario==5)
{SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"\n\nTo the Preparation Mode of MM1";cout<<"\t\t";
SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"To the Preparation Mode of MM2\n";
SetConsoleTextAttribute(hOut,FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
cout<<" ES = "<<ES1;cout<<"\t\t\t\t\t ES = "<<ES2;
cout<<"\n SS = "<<SS1;cout<<"\t\t\t\t\t SS = "<<SS2;
cout<<"\n RW = "<<RW1;cout<<"\t\t\t\t\t RW = "<<RW2;
cout<<"\n STR = "<<STR1;cout<<"\t\t\t\t STR = "<<STR2;
cout<<"\n THR = "<<THR1;cout<<"\t\t\t\t THR = "<<THR2;
Sleep(2500);
```

```
cout<<"\n GPS = "<<GPS1;cout<<"\t\t\t\t GPS = "<<GPS2;
cout<<"\n PLI = "<<PLI1;cout<<"\t\t\t\t PLI = "<<PLI2;
controller_phase(CPC1,FPC1,CPC2,FPC2);
cout<<"\n Unit Error Flag = "<<u_e_MM1;
cout<<"\t\t\t Unit Error Flag = "<<u_e_MM2;
cout<<"\n Controller Error Flag = "<<c_e_MM1;
cout<<"\t\t Controller Error Flag = "<<c_e_MM2;
handshake(u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if (scenario==6)
{SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"\n\nTo the Science Mode of MM1";cout<<"\t\t";
SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"To the Science Mode of MM2\n";
SetConsoleTextAttribute(hOut,FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
cout<<" ES = "<<ES1;cout<<"\t\t\t\t\t ES = "<<ES2;
cout<<"\n SS = "<<SS1;cout<<"\t\t\t\t\t SS = "<<SS2;
cout<<"\n RW = "<<RW1;cout<<"\t\t\t\t\t RW = "<<RW2;
cout<<"\n GPS = "<<GPS1;cout<<"\t\t\t\t GPS = "<<GPS2;
cout<<"\n STR = "<<STR1;cout<<"\t\t\t\t STR = "<<STR2;
cout<<"\n THR = "<<THR1;cout<<"\t\t\t\t THR = "<<THR2;
Sleep(2500);
cout<<"\n PLI = "<<PLI1;cout<<"\t\t\t\t PLI = "<<PLI2;
controller_phase(CPC1,FPC1,CPC2,FPC2);
cout<<"\n Unit Error Flag = "<<u_e_MM1;
cout<<"\t\t\t Unit Error Flag = "<<u_e_MM2;
cout<<"\n Controller Error Flag = "<<c_e_MM1;
cout<<"\t\t Controller Error Flag = "<<c_e_MM2;
handshake(u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if (scenario==7)
{SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"Back To the Nominal Mode of MM1";cout<<"\t\t";
SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"Back To the Nominal Mode of MM2\n";
SetConsoleTextAttribute(hOut,FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
cout<<" RW = "<<RW1;cout<<"\t\t\t\t\t RW = "<<RW2;
cout<<"\n PLI = "<<PLI1;cout<<"\t\t\t\t PLI = "<<PLI2;
Sleep(2500);
cout<<"\n GPS = "<<GPS1;cout<<"\t\t\t\t GPS = "<<GPS2;
cout<<"\n STR = "<<STR1;cout<<"\t\t\t\t STR = "<<STR2;
cout<<"\n THR = "<<THR1;cout<<"\t\t\t\t THR = "<<THR2;
Sleep(12500);
```

```
cout<<"\n ES = "<<ES1;cout<<"\t\t\t\t\t ES = "<<ES2;
cout<<"\n SS = "<<SS1;cout<<"\t\t\t\t\t SS = "<<SS2;
controller_phase(CPC1,FPC1,CPC2,FPC2);
cout<<"\n Unit Error Flag = "<<u_e_MM1;
cout<<"\t\t\t Unit Error Flag = "<<u_e_MM2;
cout<<"\n Controller Error Flag = "<<c_e_MM1;
cout<<"\t\t Controller Error Flag = "<<c_e_MM2;
handshake(u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if(scenario==8)
{SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"Back To the Off Mode of MM1";cout<<"\t\t";
SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"Back To the Off Mode of MM2\n";
SetConsoleTextAttribute(hOut,FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
cout<<" ES = "<<ES1;cout<<"\t\t\t\t\t ES = "<<ES2;
cout<<"\n SS = "<<SS1;cout<<"\t\t\t\t\t SS = "<<SS2;
cout<<"\n RW = "<<RW1;cout<<"\t\t\t\t\t RW = "<<RW2;
cout<<"\n GPS = "<<GPS1;cout<<"\t\t\t\t GPS = "<<GPS2;
cout<<"\n STR = "<<STR1;cout<<"\t\t\t\t STR = "<<STR2;
cout<<"\n THR = "<<THR1;cout<<"\t\t\t\t THR = "<<THR2;
cout<<"\n PLI = "<<PLI1;cout<<"\t\t\t\t PLI = "<<PLI2;
controller_phase(CPC1,FPC1,CPC2,FPC2);
cout<<"\n Unit Error Flag = "<<u_e_MM1;
cout<<"\t\t\t Unit Error Flag = "<<u_e_MM2;
cout<<"\n Controller Error Flag = "<<c_e_MM1;
cout<<"\t\t Controller Error Flag = "<<c_e_MM2;
handshake(u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if (scenario==9){
SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"Back To the Safe Mode of MM1";cout<<"\t\t";
SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"Back To the Safe Mode of MM2\n";
SetConsoleTextAttribute(hOut,FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
cout<<" GPS = "<<GPS1;cout<<"\t\t\t\t GPS = "<<GPS2;
cout<<"\n STR = "<<STR1;;cout<<"\t\t\t\t STR = "<<STR2;
cout<<"\n THR = "<<THR1;cout<<"\t\t\t\t THR = "<<THR2;
cout<<"\n PLI = "<<PLI1;cout<<"\t\t\t\t PLI = "<<PLI2;
Sleep(2500);cout<<"\n ES = "<<ES1;cout<<"\t\t\t\t\t ES = "<<ES2;
cout<<"\n SS = "<<SS1;cout<<"\t\t\t\t\t SS = "<<SS2;
cout<<"\n RW = "<<RW1;cout<<"\t\t\t\t\t RW = "<<RW2;
controller_phase(CPC1,FPC1,CPC2,FPC2);
```

```
cout<<"\n Unit Error Flag = "<<u_e_MM1;
cout<<"\t\t\t Unit Error Flag = "<<u_e_MM2;
cout<<"\n Controller Error Flag = "<<c_e_MM1;
cout<<"\t\t Controller Error Flag = "<<c_e_MM2;
handshake(u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);}
else if (scenario==10)
{SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"Back To the Preparation Mode of MM1";cout<<"\t";
SetConsoleTextAttribute(hOut,FOREGROUND_GREEN | FOREGROUND_BLUE);
cout<<"Back To the Preparation Mode of MM2\n";
SetConsoleTextAttribute(hOut,FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
cout<<" GPS = "<<GPS1;cout<<"\t\t\t\t GPS = "<<GPS2;
cout<<"\n STR = "<<STR1;;cout<<"\t\t\t\t STR = "<<STR2;
cout<<"\n THR = "<<THR1;cout<<"\t\t\t\t THR = "<<THR2;
cout<<"\n PLI = "<<PLI1;cout<<"\t\t\t\t PLI = "<<PLI2;
Sleep(2500);
cout<<"\n ES = "<<ES1;cout<<"\t\t\t\t\t ES = "<<ES2;
cout<<"\n SS = "<<SS1;cout<<"\t\t\t\t\t SS = "<<SS2;
cout<<"\n RW = "<<RW1;cout<<"\t\t\t\t\t RW = "<<RW2;
controller_phase(CPC1,FPC1,CPC2,FPC2);
cout<<"\n Unit Error Flag = "<<u_e_MM1;
cout<<"\t\t\t Unit Error Flag = "<<u_e_MM2;
cout<<"\n Controller Error Flag = "<<c_e_MM1;
cout<<"\t\t Controller Error Flag = "<<c_e_MM2;
handshake(u_e_MM1,u_e_MM2,c_e_MM1,c_e_MM2);} }
void error_recovery(int ES,int SS,int GPS,int STR,int RW,int THR,
int PLI,int CPC,int FPC, int m, bool e)
{char *msg1="Error Recovered in MM1";
char *msg2="Error Recovered in MM2";
char *msg111="Info Send to MM2:Error Recovered";
char *msg222="Info Receive from MM1:Error Recovered";
char *msg1111="Info Send to MM1:Error Recovered";
char *msg2222="Info Receive from MM2:Error Recovered";
char *msg3="MM1 is Ready to switch the mode";
char *msg4="MM2 is Ready to switch the mode";
if (m==1)
{internal_func1(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);cout<<"\n"<<msg1;
/* The code associated with the above function describes that MM1 has
an error and error recovery is carried out*/
cout<<"\n"<<msg111;cout<<"\t"<<msg222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if(m==2)
```

```
{internal_func2(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);
/* The code associated with the above function describes that MM2 has
an error and error recovery is carried out*/
cout<<"\n\t\t\t\t\t"<<msg2;
cout<<"\n\t\t\t\t\t"<<msg1111;cout<<"\n"<<msg2222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==3)
{internal_func3(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);cout<<"\n"<<msg1;
/* The code associated with the above function describes that MM1 has
an error and error recovery is carried out*/
cout<<"\n"<<msg111;cout<<"\t"<<msg222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==4)
{internal_func4(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);
/* The code associated with the above function describes that MM2 has
an error and error recovery is carried out*/
cout<<"\n\t\t\t\t\t"<<msg2;
cout<<"\n\t\t\t\t\t"<<msg1111;cout<<"\n"<<msg2222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==5)
{internal_func5(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);cout<<"\n"<<msg1;
/* The code associated with the above function describes that MM1 has
an error and error recovery is carried out*/
cout<<"\n"<<msg111;cout<<"\t"<<msg222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==6)
{internal_func6(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);
/* The code associated with the above function describes that MM2 has
an error and error recovery is carried out*/
cout<<"\n\t\t\t\t\t"<<msg2;
cout<<"\n\t\t\t\t\t"<<msg1111;cout<<"\n"<<msg2222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==7)
{internal_func7(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);cout<<"\n"<<msg1;
/* The code associated with the above function describes that MM1 has
an error and error recovery is carried out*/
cout<<"\n"<<msg111;cout<<"\t"<<msg222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==8)
{internal_func8(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);
/* The code associated with the above function describes that MM2 has
an error and error recovery is carried out*/
cout<<"\n\t\t\t\t\t"<<msg2;
```

```
cout<<"\n\t\t\t\t\t"<<msg1111;cout<<"\n"<<msg2222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==9)
{internal_func9(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);cout<<"\n"<<msg1;
/* The code associated with the above function describes that MM1 has
an error and error recovery is carried out*/
cout<<"\n"<<msg111;cout<<"\t"<<msg222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==10)
{internal_func10(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);
/* The code associated with the above function describes that MM2 has
an error and error recovery is carried out*/
cout<<"\n\t\t\t\t\t"<<msg2;
cout<<"\n\t\t\t\t\t"<<msg1111;cout<<"\n"<<msg2222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==11)
{internal_func11(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);cout<<"\n"<<msg1;
/* The code associated with the above function describes that MM1 has
an error and error recovery is carried out*/
cout<<"\n"<<msg111;cout<<"\t"<<msg222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==12)
{internal_func12(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);
/* The code associated with the above function describes that MM2 has
an error and error recovery is carried out*/
cout<<"\n\t\t\t\t\t"<<msg2;
cout<<"\n\t\t\t\t\t"<<msg1111;cout<<"\n"<<msg2222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==13)
{internal_func13(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);cout<<"\n"<<msg1;
/* The code associated with the above function describes that MM1 has
an error and error recovery is carried out*/
cout<<"\n"<<msg111;cout<<"\t"<<msg222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==14)
{internal_func14(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);
/* The code associated with the above function describes that MM2 has
an error and error recovery is carried out*/
cout<<"\n\t\t\t\t\t"<<msg2;
cout<<"\n\t\t\t\t\t"<<msg1111;cout<<"\n"<<msg2222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==15)
{internal_func15(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);cout<<"\n"<<msg1;
```

```
/* The code associated with the above function describes that MM1 has
an error and error recovery is carried out*/
cout<<"\n"<<msg111;cout<<"\t"<<msg222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==16)
{internal_func16(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);
/* The code associated with the above function describes that MM2 has
an error and error recovery is carried out*/
cout<<"\n\t\t\t\t\t"<<msg2;
cout<<"\n\t\t\t\t\t"<<msg1111;cout<<"\n"<<msg2222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==17)
{internal_func17(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);cout<<"\n"<<msg1;
/* The code associated with the above function describes that MM1 has
an error and error recovery is carried out*/
cout<<"\n"<<msg111;cout<<"\t"<<msg222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==18)
{internal_func18(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);
/* The code associated with the above function describes that MM2 has
an error and error recovery is carried out*/
cout<<"\n\t\t\t\t\t"<<msg2;
cout<<"\n\t\t\t\t\t"<<msg1111;cout<<"\n"<<msg2222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==19)
{internal_func19(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);cout<<"\n"<<msg1;
/* The code associated with the above function describes that MM1 has
an error and error recovery is carried out*/
cout<<"\n"<<msg111;cout<<"\t"<<msg222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==20)
{internal_func20(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);
/* The code associated with the above function describes that MM2 has
an error and error recovery is carried out*/
cout<<"\n\t\t\t\t\t"<<msg2;
cout<<"\n\t\t\t\t\t"<<msg1111;cout<<"\n"<<msg2222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==21)
{internal_func21(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);cout<<"\n"<<msg1;
/* The code associated with the above function describes that MM1 has
an error and error recovery is carried out*/
cout<<"\n"<<msg111;cout<<"\t"<<msg222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
```

```
else if (m==22)
{internal_func22(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);
/* The code associated with the above function describes that MM2 has
an error and error recovery is carried out*/
cout<<"\n\t\t\t\t\t"<<msg2;
cout<<"\n\t\t\t\t\t"<<msg1111;cout<<"\n"<<msg2222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==25)
{internal_func25(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);cout<<"\n"<<msg1;
/* The code associated with the above function describes that MM1 has
an error and error recovery is carried out*/
cout<<"\n"<<msg111;cout<<"\t"<<msg222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}
else if (m==26)
{internal_func26(ES,SS,GPS,STR,RW,THR,PLI,CPC,FPC,e);
/* The code associated with the above function describes that MM2 has
an error and error recovery is carried out*/
cout<<"\n\t\t\t\t\t"<<msg2;
cout<<"\n\t\t\t\t\t"<<msg1111;cout<<"\n"<<msg2222;
cout<<"\n"<<msg3;cout<<"\t\t"<<msg4;}else cout<<"\n\n";}
void handshake(bool u_e_MM1, bool u_e_MM2,bool c_e_MM1,bool c_e_MM2)
{char *send1="Info Send to MM1:No Error";
char *send2="Info Send to MM2:No Error";
char *rec1="Info Receive from MM1:No Error";
char *rec2="Info Receive from MM2:No Error";
char *send3="Info Send to MM1:Unit Error";
char *send4="Info Send to MM2:Unit Error";
char *rec3="Info Receive from MM1:Unit Error";
char *rec4="Info Receive from MM2:Unit Error";
char *send33="Info Send to MM1:Controller Error";
char *send44="Info Send to MM2:Controller Error";
char *rec33="Info Receive from MM1:Controller Error";
char *rec44="Info Receive from MM2:Controller Error";
char *msg1="MM1 is Ready to switch the mode";
char *msg2="MM2 is Ready to switch the mode";
char *msg3="Wait until Error recovered in MM1";
char *msg4="Wait until Error recovered in MM2";
if (u_e_MM1==0 && u_e_MM2==0 && c_e_MM1==0 && c_e_MM2==0)
{cout<<"\nNo Error in given mode of MM1";
cout<<"\t\tNo Error in given mode of MM2";
cout<<"\n"<<send2;cout<<"\t\t"<<send1;
cout<<"\n"<<rec2;cout<<"\t\t"<<rec1;
cout<<"\n"<<msg1;cout<<"\t\t"<<msg2;}
```

```
else if(u_e_MM1==1 && u_e_MM2==0 && c_e_MM1==0 && c_e_MM2==0)
{cout<<"\nError in given mode of MM1 ";
cout<<"\t\tNo Error in given mode of MM2\n";
cout<<send4;cout<<"\t\t"<<send1;
cout<<"\n"<<rec2;cout<<"\t\t"<<rec3;
cout<<"\nError Recovery Starts";cout<<"\t\t\t"<<msg3;
cout<<endl;}
else if(u_e_MM1==0 && u_e_MM2==1 && c_e_MM1==0 && c_e_MM2==0)
{cout<<"\nNo Error in given mode of MM1 ";
cout<<"\t\tError in given mode of MM2";
cout<<"\n"<<send2;cout<<"\t\t"<<send3;
cout<<"\n"<<rec4;cout<<"\t"<<rec1;
cout<<"\n"<<msg4;cout<<"\tError Recovery Starts\n";}
else if(u_e_MM1==1 && u_e_MM2==1 && c_e_MM1==0 && c_e_MM2==0)
{cout<<"\nError in given mode of MM1 ";
cout<<"\t\tError in given mode of MM2\n";
cout<<send4;cout<<"\t\t"<<send3;
cout<<"\n"<<rec4;cout<<"\t"<<rec3;
cout<<"\n"<<msg4;cout<<"\t"<<msg3;
cout<<"\nError Recovery Starts";
cout<<"\t\t\tError Recovery Starts\n";}
else if(u_e_MM1==0 && u_e_MM2==0 && c_e_MM1==1 && c_e_MM2==0)
{cout<<"\nError in given mode of MM1 ";
cout<<"\t\tNo Error in given mode of MM2\n";
cout<<send44;cout<<"\t"<<send1;
cout<<"\n"<<rec2;cout<<"\t\t"<<rec33;
cout<<"\nError Recovery Starts";cout<<"\t\t\t"<<msg3;
cout<<endl;}
else if(u_e_MM1==0 && u_e_MM2==0 && c_e_MM1==0 && c_e_MM2==1)
{cout<<"\nNo Error in given mode of MM1 ";
cout<<"\t\tError in given mode of MM2";
cout<<"\n"<<send2;cout<<"\t\t"<<send33;
cout<<"\n"<<rec44;cout<<"\t"<<rec1;
cout<<"\n"<<msg4;cout<<"\tError Recovery Starts\n";}
else if(u_e_MM1==0 && u_e_MM2==0 && c_e_MM1==1 && c_e_MM2==1)
{cout<<"\nError in given mode of MM1 ";
cout<<"\t\tError in given mode of MM2\n";
cout<<send44;cout<<"\t"<<send33;
cout<<"\n"<<rec44;cout<<"\t"<<rec33;
cout<<"\n"<<msg4;cout<<"\t"<<msg3;
cout<<"\nError Recovery Starts";
cout<<"\t\t\tError Recovery Starts\n";}
else {cout<<"\n\n";}}
```

Joukahaisenkatu 3-5 B, 20520 Turku, Finland | www.tucs.fi

University of Turku
- Department of Information Technology
- Department of Mathematics

Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

Turku School of Economics and Business Administration
- Institute of Information Systems Sciences

ISBN 978-952-12-2721-9
ISSN 1239-1891