P-3498

BONAFIDE CERTIFICATE

Certified that this project report entitled "**PC BASED CONTROL SYSTEM REDUNDANCY MANAGEMENT**" is the bonafide work of J.BEULAH JOYSON [Reg. no. 0920107002] who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

K. Thilagavathi 5/4/11

Project Guide

Ms.K.Thilagavathi

[illegible] 5/4/11

Head of the Department

Dr. (Mrs.) Rajeswari Mariappan

The candidate with university Register no. 0920107002 is examined by us in the project viva-voce examination held on ...21-4-2011....

[illegible] 21/4/11

Internal Examiner

[illegible] 21/4/11

External Examiner

भारत सरकार
अंतरिक्ष विभाग
द्रव नोदन प्रणाली केन्द्र
महेन्द्रगिरि पी. ओ. तिरुनेलवेली जिला - 627 133,
तमिलनाडु, भारत.

4637-281900
4637-281 (Extn.) फैक्स - प्रशा : 04637-222490
फैक्स - क्रय : 04637-222496
फैक्स - भण्डार : 04637-281567
फैक्स - लेखा : 04637-281547

सत्यमेव जयते

Government of India
Department of Space
Liquid Propulsion Systems Centre
Mahendragiri P.O.,Tirunelveli District-627 133,
Tamil Nadu, India.
Telephone : 04637-281900 (Operator)
Fax : 04637-281 (Extn.)

Fax-Admn : 04637-222490
Fax-Pur : 04637-222496
Fax-Sts : 04637-281567
Fax-Accts : 04637-281547

S. KANNU, M.E., MBA,
. MANAGER, PM&E.

31.03.2011

Programme Monitoring & Evaluation (PM&E)

<u>Certificate</u>

This is to certify that **Ms. J. Beulah Joyson** of IInd year ME (Communication Systems) from Kumaraguru College of Technology, Coimbatore has done ***Project Work*** at Data Acquisition and Auxiliary System, LPSC, Mahendragiri as per details given below:

Name of the Guide	:	Shri. J. Muruganantha Bhaskaran, Manager, DA&AS
Name of the Group	:	Data Acquisition & Auxiliary System
Details of Project Work	:	"PC Based Control System Redundancy Managemen
Quality of performance and Interest shown	:	Excellent
Period of Project work	:	01.12.2010 to 31.03.2011
Conduct	:	Excellent

[illegible]

(R.S. Kannu)

R.S. KANNU
Dy. Manager, PM&
LPSC, ISRO
Mahendragiri - 627

ACKNOWLEDGEMENT

I express my profound gratitude to our Chairman **Padmabhusan Arutselvar Dr.N.Mahalingam B.Sc.,F.I.E.** for giving this opportunity to pursue this course.

At this pleasing moment of having successfully completed the project work, I wish to acknowledge my sincere gratitude and heartfelt thanks to our beloved Principal **Dr.S.Rama Chandran Ph.D.,** for having given me the adequate support and opportunity for completing this project work successfully.

I express my sincere thanks to **Dr.Rajeswari Mariyappan Ph.D.,** Head of the Department of Electronics and Communication Engineering, for her valuable ideas and encouragement throughout this project

In particular, I wish to thank and express my gratitude to our project coordinator **Associate Prof.D.Mohanageetha (Ph.D).,**Associate Professor,Department of Electronics and Communication Engineering, for her expert counseling and guidance to make this project to a great deal of success with her careful supervision.

I extend my heartfelt thanks to my guide **Asst. Prof.K.Thilagavathi M.E.,** Assistant Professor (SRG), Department of Electronics and Communication Engineering for her ideas and suggestion, which have been very helpful for the completion of this project work.

I am greatly thankful to express my deep sense of gratitude to respected **Mr.Asir Packiaraj, DGM, A&IS/LPSC-M,** for giving me an opportunity to work at LPSC, Mahendragiri.

I am thankful to **Mr. Murugenantha Baskaran, Manager, SE&ML/E&I/LPSC-M,** for having his consent to do this project work.

I am greatly indebted to my external guides **Mr.S.S. Murugan, SE&ML/E&I/LPSC-M and Mr.Ashish Kumar Jain, SE&ML/E&I/LPSC-M,** who boosted me to do this project work and for giving their excellent and valuable guidance and encouragement for successfully carrying out this training.

Last, but not the least, I would like to express my gratitude to my family members, friends and to all my staff members of Electronics and Communication Engineering department for their encouragement and support throughout the course of this project.

ABSTRACT

In LMF (Liquid Mahendragiri Facility, ISRO, Mahendragiri) spacecraft thruster facility is envisaged for qualifying satellite thrusters. For controlling the field elements remotely PC based control system is installed with 1ms cycle time. The control system includes one server, one main controller, one remote controller and the field elements. The system operates under two modes of operation namely auto mode and manual mode. The installed system is working without any redundancy. Since the TCP/IP communication link between the main controller and remote controller is the weakest part in the PC based control system, it has to be strengthened by developing a redundant link in such a way that if one link fails during operation, the other link will automatically carry over the communication without any failure. Also in order to strengthen the control system as a whole, system redundancy has to be established that includes two main controllers and two remote controllers. So that during operation, if one of the main/remote controllers gets failed, the other main/remote controller will detect the system failure and take over the operation and so the failure in testing can be avoided.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF ABBREVIATIONS

LMF	------	Liquid Mahendragiri Facility
PXI	------	PCI Extension For Instrumentation
PCI	------	Peripheral Component Interconnect
PLC	------	Programmable Logic Controller
TCP	------	Transmission Control Protocol
UDP	------	User Datagram Protocol
XMLHTTP	------	Extensible Markup Language Hyper Text Transfer Protocol
LPSC	------	Liquid Propulsion System Centre
PHP	------	Hyper Text Preprocessor
HMI	------	Human Machine Interface
AJAX	------	Asynchronous JavaScript and XML
XML	------	Extensible Markup Language
HTML	------	Hyper Text Markup Language
DAS	------	Data Acquisition System

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

Indian Space Research Organization (ISRO) promotes the development and applications of space technology and science for the socio economic benefits of the nation. It has several centers to realize execute its objectives. LPSC is the leading centre in the area of liquid propulsion for ISRO's launch vehicles like PSLV and GSLV. LPSC also supplies liquid thrusts for its satellite programs. LPSC has set up three units, at Valiamala near Thiruvananthapuram, at Mahendragiri near Kanyakumari and at Bangalore to execute its tasks.

This centre is engaged in research and development of earth storable and cryogenic propulsion systems for launch vehicles and aircrafts. All the facilities for assembly, integration, testing and propellant production are also available here. PSLV uses both solid and liquid propellants. The second and fourth stages use liquid propellant. These stages consist of liquid engines designed for a particular thrust with propellant tanks, pressurization system and other sub system for their functional control. LPSC is prime responsible for developing cryogenic engines.

PXI based control system under Linux environment is installed to replace costly PLCs for the future process control activities of liquid engine test facilities. The hardware realization of PLCs is time consuming and costly compared to software realization in PC based system. In this software era the PC based Control System can replace the PLCs with the better system performance, since the hardware and software line goes on decreasing drastically in real time application areas due to the development of high speed processors and real time operating system.

The PLC system is replaced with a computer and extension cards. The industrial control activities. Ethernet is used for networking the sub systems. The system includes one main controller a number of remote controllers less than 100. The number of remote controllers depends on the number of valves to be controlled and the number of feedback parameters to be taken back. The communication between the main controller and the remote controllers is done via real time Ethernet. Main station takes all the control decisions and remote stations execute it using I/O cards. A hot standby control system is visualized with proper synchronization. TCP and UDP are the communication protocols used in different layers of the system. Web based human machine interface is used as the front end of the system.

The development of the system is carrying out in Red Hat Enterprise Linux Advance Server WS -4, and the code is planning to port to real time Linux. GCC compiler is used as the compiler and TCP and UDP are used as communication protocols. To meet cold and hot pulse requirements (8msec on 120 sec off) control system is designed with the cycle time of 1msec.

1.2 ADVANTAGES OF PC BASED CONTROL SYSTEM:

- Cost Effective: PC based Control System is highly cost effective as compare with PLC. The upper hand is coming due to two things. One is its hardware is more general purpose than a PLC. Hence there will be a bigger market for the hardware and hence less price. The second is all about the software. The software is in-house developed one and hence there is no extra cost for the software development. This will provide a strong upper hand for the PC based Control System over the PLC based System.
- Reusability: The code is there in our hand and hence the entire code is reusable for any number of times. So if there is a new test facility is coming up, we need not go for a costly PLC hardware and software, instead of which we can buy the PXI hardware, which is much cheep and we can use the already available software.
- Flexibility:The PLC source code is invisible to the user. The user can only see what he is supposed to. So, if there is a complicated requirement in future, then there is a need to approach the vendor. But the same thing is possible in PC based system without much pain as the entire source code is available with us and we know the entire source code

- Future: The current industry trend is to update the systems with more universal cost effective generic systems instead of application specific systems. So PC based control systems are the future process control systems and we are running ahead of time.
- Easy Programmability: The auto sequence, abort sequence and interlock programming has been done in spread sheet and is the easiest way of doing things. Almost all the present requirement could be met by the same. Advanced logic if required can be implemented in the second version of the system where the C programming capability to the user is planned.

Implemented PC based control system is having the following capabilities.

- Manual commanding of Field valves
- Auto sequence program execution as per test chronology.
- Execution of manual abort sequence
- Execution of auto abort sequence
- Pulse generation for engine valve ON/OFF.
- Hold operations during auto sequencing

1.3 PROJECT GOAL

The goal of this project is to develop network redundancy in order to strengthen the communication link between the main controller and remote controller. Also to develop system redundancy in PC based control system so that the possibility of failure of the testing can be greatly reduced.

1.4 SOFTWARE USED

- LINUX
- Socket Programming in C Language

CHAPTER 2
SYSTEM ARCHITECTURE

2.1 BASIC ARCHITECTURE

The PC based control system consists of Remote Station, Main Controller ,Server and Operating Station. The layered structure of the PC based control system is explained in detail with the help of basic block diagram shown in Fig 2.1. The commands are issued from the main controller to the remote stations. The remote station activates the field components according to the commands issued by the main controller. Remote station also acquires different parameters of the system and sent it to the main controller. The main controller send these parameters to the server, where it is stored in the database. The operating station queries the data from the server and process it to some visual mode and displays it.

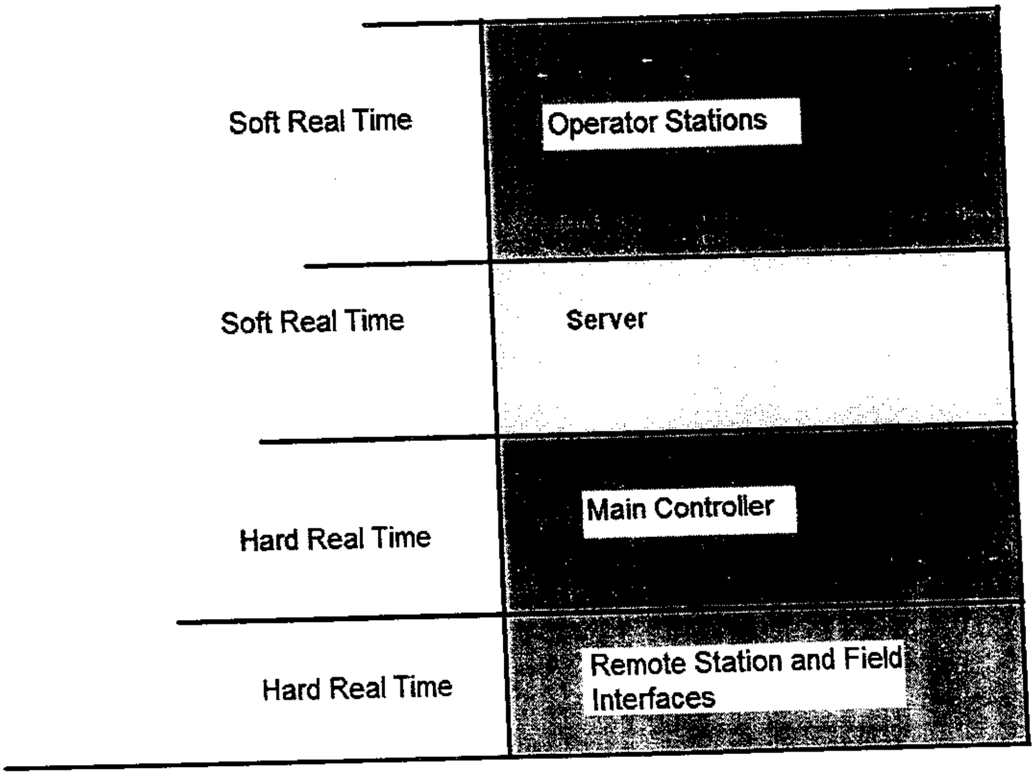


Figure 2.1 Layered Architecture

The topmost layer is a soft real time layer where the user interacts with the control system. It includes the MIMIC part and the real time display part. The second layer contains the server and data storage systems. This system is also soft real time system. The third layer contains the main controller, which make the heart of the control system and the last layer is the remote controller and the interfacing cards and the field devices, and is a hard real time system.

2.1.1 LAYER 1 – OPERATOR STATIONS

This is the layer where the user intervention comes into picture. This layer is implemented using web technologies. The server will hold all the codes for the MIMIC and the operator stations will simply use web browsers for accessing the same. The field status is to be updated in the MIMIC without the user intervention. AJAX based technology is used in this layer for achieving the auto refresh. The client machine which runs the browser sends an XMLHTTP request to the server and the server will replay to this with the latest updates. This update will be received by the client and it will update itself. The XMLHTTP request delay can be adjusted in the server program.

There will be three types of logins. If you logged in as a user, then you cannot give any commands, but can see the status only. If you are an operator then you can see the status of all the valves and you can give the commands. Administrator is a privileged user who can change the valve settings and. All the operations are being logged in the command history for future reference. Since this layer is implemented using web technology, there is no limit to the number of operator stations that can be connecting to the server, but due to security reasons this network will be limited physically to the control room only.

2.1.2 LAYER 2- THE SERVER

This layer makes an interface between the user and the hard real time layer. This layer acts as a storage system also for storing the feedback data from the field. It takes the command history for future reference too. Basically this subsystem contains two programs; one is the program for acting itself as a web server for the MIMIC activities. Second is to receive the data from the main controller and to send the manual commands to the main controller. This is done by a C program.

UDP is used for the communication between the server and the main controller. This server is in soft real time mode. So the communication between the server and the main station would not affect the hard real time operation of the server. This is achieved by using UDP as the communication protocol in between the main controller and the server. So in auto mode, the main controller will run the auto commands and the feedback data will be sending to the server and will continue with its operations without bothering about the server activities. The web server part is done using the LAMP (Linux, Apache Server, mySQL, PHP). The remaining program of the server is done in C language.

In manual mode we have to ensure the command availability and hence more reliable communication is required. This can be done either by TCP or by UDP. We have done it using UDP since UDP is already implemented in this layer for auto mode. This is done by a two way UDP. Both the server and the main controller acts in UDP server mode and UDP client mode and if there is a time out in this layer operation the fault tolerance program will take over the control activities.

2.1.3 LAYER 3- THE MAIN CONTROLLER

This is the heart of the control system, which takes all the control decisions. The auto sequence and abort sequence are available in the main controller only. The main controller reads the auto sequence file well before the control loop and stores everything in a data structure and will send to the remote controllers in proper time.

It is decided to avoid all the hard disk activities during the critical control cycle (The auto sequence mode). To do this we read all the auto sequence and abort sequence well before the control cycle execution and store it in a data structure. Only the delay in data structure traversal will occur in the control cycle and is totally deterministic.

In parallel to the auto sequence the system will check for the occurrence of any abort condition, and if occurred, it will stops executing the auto sequence and will start the abort sequence. Abort sequence is also like auto sequence where the t+0 is not defined initially. It is the cycle when the abort sequence gets triggered. Interlock will also run parallel, where the system checks for some particular conditions and if occurred then it triggers series of activities.

The remote nodes can be configured from the main station itself. The configuration file will be sending from the main station to all the remote stations after establishing the communication between the main station and all the remote stations. The remote stations validity of the configuration file. If there is any discrepancy in the configuration, the remote node will inform the main controller in next acknowledgment cycle and corresponding error message will also be displayed. Once the configuration is over, the actual control loop will run depending on the mode of operation (Auto/Manual Mode).

There are different modules in the main controller, which includes the modules for cycle-time management, auto sequence operations, abort sequence operation etc. No hard disk operation is permitted in the control cycle in the main station or remote stations. All the data will be available in the RAM only in the main controller and will be written in the server in the next logical cycle. Synchronization of the control loop is also done in this layer.

PID module is also implemented in this layer. The feedback data from the remote controller is stored in a FIFO register in the main controller and is used for PID calculation. The PID constants file is also available in this layer, which is user editable. The PID is applicable only for the analog valves, whose details are given in the specific PID file.

Command processing is another activity of this layer, which prepare the command string from the auto sequence data structure and will insert the manual commands if the system runs in auto-manual mode. Also the system changes the commands according to the abort program. Finally it sends the command strings prepared for each remote node to the respective remote nodes and waits for the feedback data. The loop will be repeating.

2.1.4 LAYER -4 -THE REMOTE STATIONS

The remote stations are simply dumb terminals, which implements the commands from its master the main station. Once the system starts its operations, the remote station receives the configuration data in the first 10 cycles and will configure itself using this data. If there is any discrepancy in the configuration data, and then it will inform the master in the next cycle. Once the configuration is over and the remote station is ready for the control activates, it receives the command string from the main controller and checks for the authenticity of the commands. Once the authenticity is confirmed, it separates the analog and digital commands separately and sends to the card driver modules which drive the cards using the given inputs. The remote stations will acquire the analog and digital data and will send it

2.2 PC BASED CONTROL SYSTEM WITHOUT REDUNDANCY

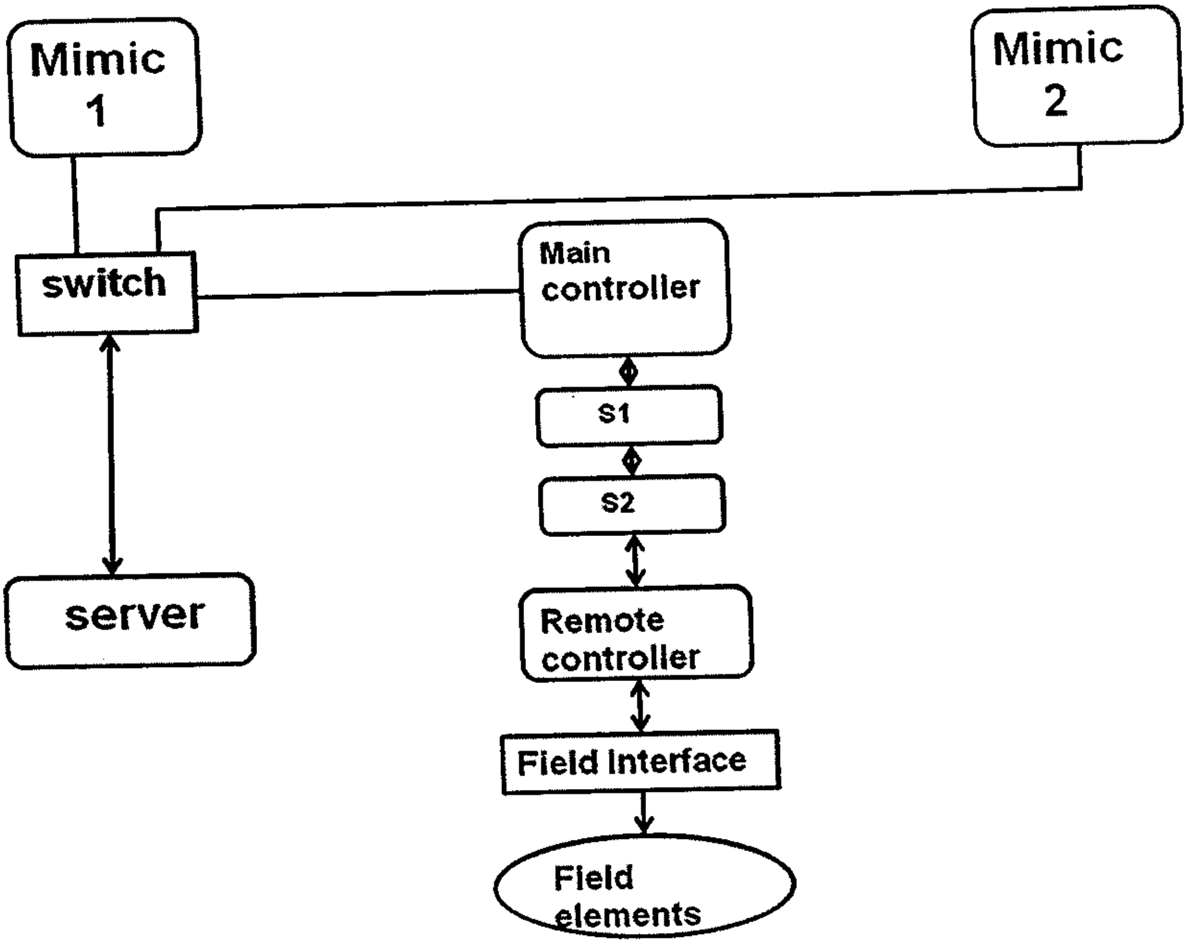


Figure 2.2 PC Based Control System without Redundancy

The implemented control system without redundancy includes Main controller, remote controller, operator consoles, server, Ethernet switches (separate switch for control and mimic layers) and MIMICS PCs (2 Nos). The main and remote controllers are PXI bus based systems with Digital Input (DI) and digital Output (DO) cards.

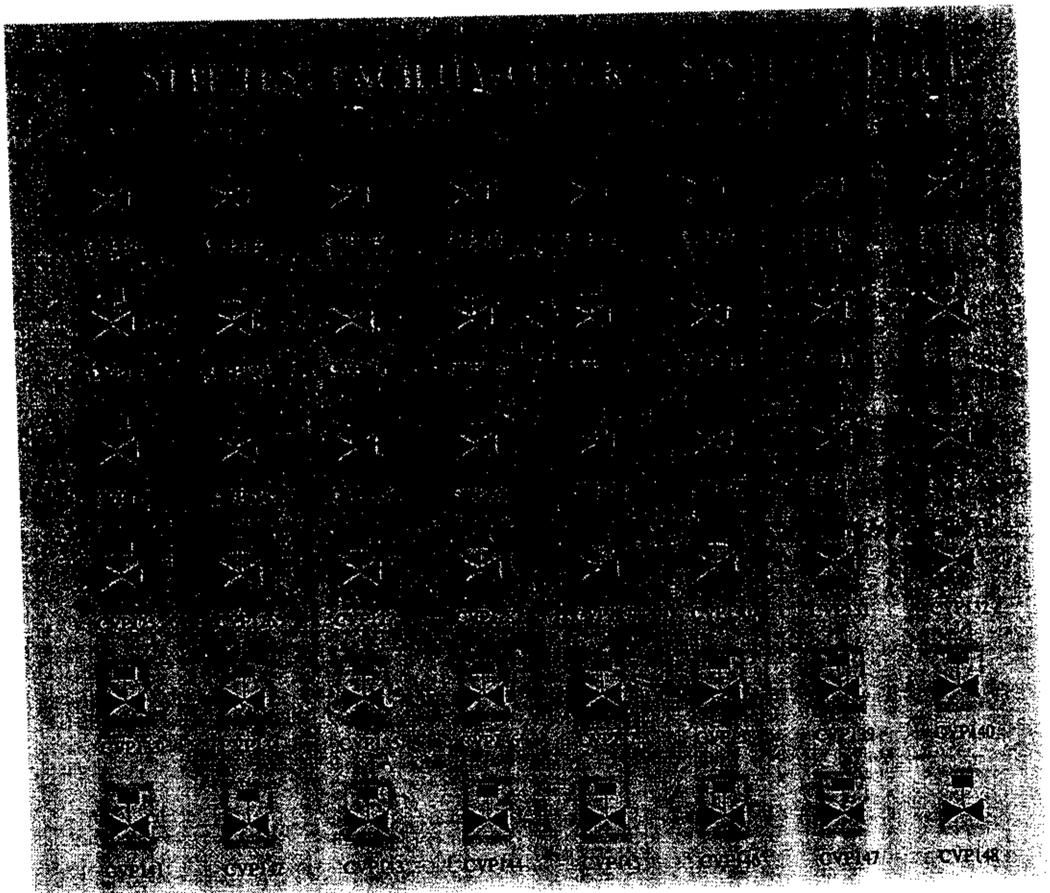
The Operator Console consists of switches for Authorize, Hold, Abort, Auto /Manual selection switch and reset. The commands from the Authorize, Hold, Abort, Auto, Manual switches are given to the Main controller from the console through the digital input card. The Authorize command will enable the auto sequence, the hold command will Hold the auto

2.3 MIMIC INTERFACE

Mimic station is the Human Machine Interface (HMI) of the STTF control system. Web based HMI is implemented hence mimic can be accessed through platform independent web browsers. Require program and data are available in the server and the system is working on web technologies. Latest web technology like AJAX is being implemented in this layer for auto refreshing of the MIMIC pages. The operator stations layer has the following functionalities.

- It reflects the field changes for the user for monitoring.
- It gives the manual commands for manual operation.

The MIMIC for 96 valves is split in to two, such that each of them contains 48 valves. The valves are digital valves and can operate in all three modes. Sample mimic implemented is given in Figure 2.3



CHAPTER 3
MODES OF OPERATION

3.1 MODES OF OPERATION

There are two modes of operation for the process control activities in LPSC-M.

- Auto mode
- Manual Mode

3.1.1 AUTO MODE

In auto mode, the main controller will execute a pre programmed control operation without any user intervention. In this mode of operation, the server and the MIMIC stations are simply the dumb listeners of what is happening in the field. The server gets all the feedback data and will be written in its hard disk. The MIMIC stations will get the status updating and the operator can see the valve status in the display. The auto sequence programming is to be done in a spread sheet in the main controller and the main controller will implement the same at proper time. In auto mode, the system will take care the interlock and the abort conditions as per the interlock and abort programming.

3.1.1.1 AUTO SEQUENCE

Auto sequence is being implemented in the main controller. There are basically two auto sequence files, one is the digital auto sequence and the second is for analog auto sequence.

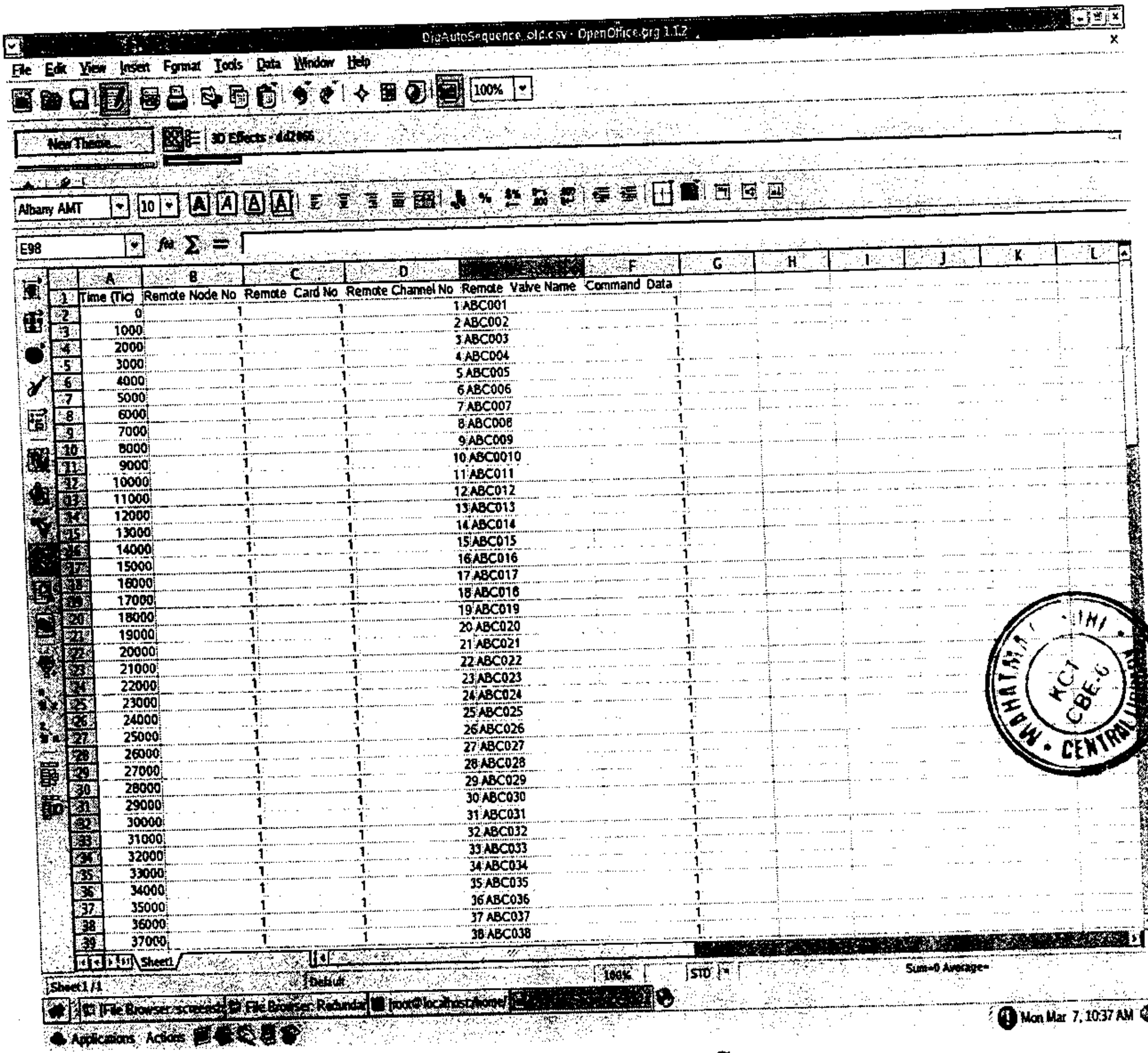
- DigAutoSequence.csv
- AnlgAutoSequence.csv

These are the files available for auto sequence operation. Digital auto sequence file contains the auto sequences for digital valve operations, while analog auto sequence commands are available in AnlgAutoSequence.csv file. Both of these are the spread sheet file (CSV files) and are user editable. A sample command file is shown below. The file contains basically six fields.

- Time
- Remote Node No
- Remote Card No
- Remote Channel No
- Remote Valve Name

P- 3498

3.1.1.2 DIGITAL AUTO SEQUENCE


DigAutoSequence_old.csv - OpenOffice.org 1.1.2

E98

	A	B	C	D	E	F
1	Time (Tic)	Remote Node No	Remote Card No	Remote Channel No	Remote Valve Name	Command Data
2	0	1	1	1	ABC001	1
3	1000	1	1	2	ABC002	1
4	2000	1	1	3	ABC003	1
5	3000	1	1	4	ABC004	1
6	4000	1	1	5	ABC005	1
7	5000	1	1	6	ABC006	1
8	6000	1	1	7	ABC007	1
9	7000	1	1	8	ABC008	1
10	8000	1	1	9	ABC009	1
11	9000	1	1	10	ABC0010	1
12	10000	1	1	11	ABC011	1
13	11000	1	1	12	ABC012	1
14	12000	1	1	13	ABC013	1
15	13000	1	1	14	ABC014	1
16	14000	1	1	15	ABC015	1
17	15000	1	1	16	ABC016	1
18	16000	1	1	17	ABC017	1
19	17000	1	1	18	ABC018	1
20	18000	1	1	19	ABC019	1
21	19000	1	1	20	ABC020	1
22	20000	1	1	21	ABC021	1
23	21000	1	1	22	ABC022	1
24	22000	1	1	23	ABC023	1
25	23000	1	1	24	ABC024	1
26	24000	1	1	25	ABC025	1
27	25000	1	1	26	ABC026	1
28	26000	1	1	27	ABC027	1
29	27000	1	1	28	ABC028	1
30	28000	1	1	29	ABC029	1
31	29000	1	1	30	ABC030	1
32	30000	1	1	31	ABC031	1
33	31000	1	1	32	ABC032	1
34	32000	1	1	33	ABC033	1
35	33000	1	1	34	ABC034	1
36	34000	1	1	35	ABC035	1
37	35000	1	1	36	ABC036	1
38	36000	1	1	37	ABC037	1
39	37000	1	1	38	ABC038	1

Figure 3.1 Digital Auto Sequence

Among these, only one of the fields is not required for the control system operation. That is the remote valve name. All the remaining fields play crucial role in the implementation of the auto sequence command. The valve name field is not required as it is the field useful for the mechanical team as the valve name is something related with the process.

The valves are being addressed by the instrumentation engineer by the following phrase: "The Xth channel of the Yth card of the Zth remote node is in the state of **C** at the time third one is Y, the forth one is X, while the last one is C. the fifth channel is not that important to an instrumentation engineer. The instrumentation engineer can edit this file according to his process requirement.

3.1.1.3 ANALOG AUTO SEQUENCE

This is almost equivalent to the digital auto sequence. The difference here is instead of the pneumatic valves; it operates the control valves and hence controls the percentage opening of the control valve. The command file pattern is exactly same as that of the digital auto sequence file (DigAutoSequence.csv), but the difference is that instead of the ones and zeros in the command data field in the digital auto sequence file, this particular file contains the percentage opening of the control valve. This can be from zero percentage to 100 percentages. The percentage can be decimals also. Two decimal points are allowed in the percentage opening of the control valves. So the command of 10.12% is perfectly a valid command.

The following figure shows a screen shot of an analog auto sequence command file. The parameters of this file are same as that of the digital auto sequence file. The last one is the lone different field. So I feel that there is no need of explaining once again about all the other fields. Only explanation required is about the last field, where the percentage opening of the valve is determined. So kindly consider the digital auto sequence part of this document for programming all the fields other than the last one. The last one is being programmed as per following explanation.

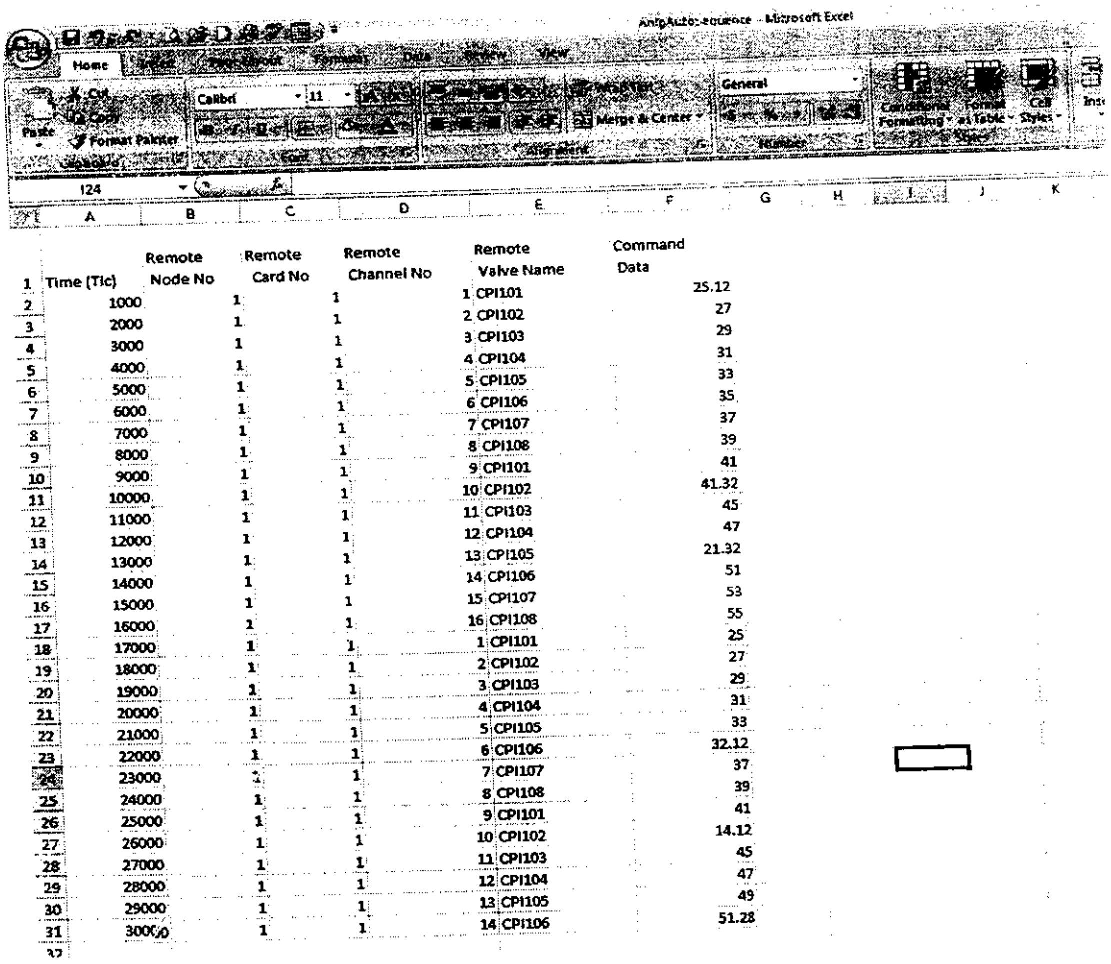
Suppose the maximum flow rate is X and minimum flow rate is 0. The maximum percentage opening of the valve is 100%, while the minimum is 0%. So if the valve is opened by 100% then the flow rate will be X, and if the valve is opened by 0% then the flow rate also will be 0. So the percentage opening corresponding for the flow rate Y can be calculated by the following equation.

$100/X=Y/P$, where

X=Flow rate for 100% opening of the valve

Y= Required flow rate at present condition

P=Percentage opening required for the flow rate of Y



Time (Tic)	Remote Node No	Remote Card No	Remote Channel No	Remote Valve Name	Command Data
1000	1	1	1	CPI101	25.12
2000	1	1	2	CPI102	27
3000	1	1	3	CPI103	29
4000	1	1	4	CPI104	31
5000	1	1	5	CPI105	33
6000	1	1	6	CPI106	35
7000	1	1	7	CPI107	37
8000	1	1	8	CPI108	39
9000	1	1	9	CPI101	41
10000	1	1	10	CPI102	41.32
11000	1	1	11	CPI103	45
12000	1	1	12	CPI104	47
13000	1	1	13	CPI105	21.32
14000	1	1	14	CPI106	51
15000	1	1	15	CPI107	53
16000	1	1	16	CPI108	55
17000	1	1	1	CPI101	25
18000	1	1	2	CPI102	27
19000	1	1	3	CPI103	29
20000	1	1	4	CPI104	31
21000	1	1	5	CPI105	33
22000	1	1	6	CPI106	32.12
23000	1	1	7	CPI107	37
24000	1	1	8	CPI108	39
25000	1	1	9	CPI101	41
26000	1	1	10	CPI102	14.12
27000	1	1	11	CPI103	45
28000	1	1	12	CPI104	47
29000	1	1	13	CPI105	49
30000	1	1	14	CPI106	51.28

Figure 3.2 Analog Auto Sequence

3.1.2 MANUAL MODE

Manual mode is used for the evaluation of the system and for the health check for making the system ready for the control activities. This is mainly used for pre test preparation activities. In manual mode the user can give commands from the operator terminals and the system will implement the commands. All the four layers of the control system are active participants of this mode of operation. The manual command is coming from the operator station and will be written in the manual command file in the server. The server sends the manual commands as UDP packets to the main controller and the main controller receives the same and will use it for the manual command packets generation. The manual command packets will be sent to the remote stations. The remote stations will execute the commands and will return the feedback data. The feedback data is received in the main controller and

CHAPTER 4
HARDWARE LEVEL SYSTEM IMPLEMENTATION

4.1 HARDWARE REQUIREMENTS

S.No	Layer	Hardware
1.	MIMIC Layer	Operator station PCs
2.	Server Layer	Server PC
3.	Main Controller Layer	PXI
4.	Remote Controller Layer	PXI

4.2 PXI SYSTEM

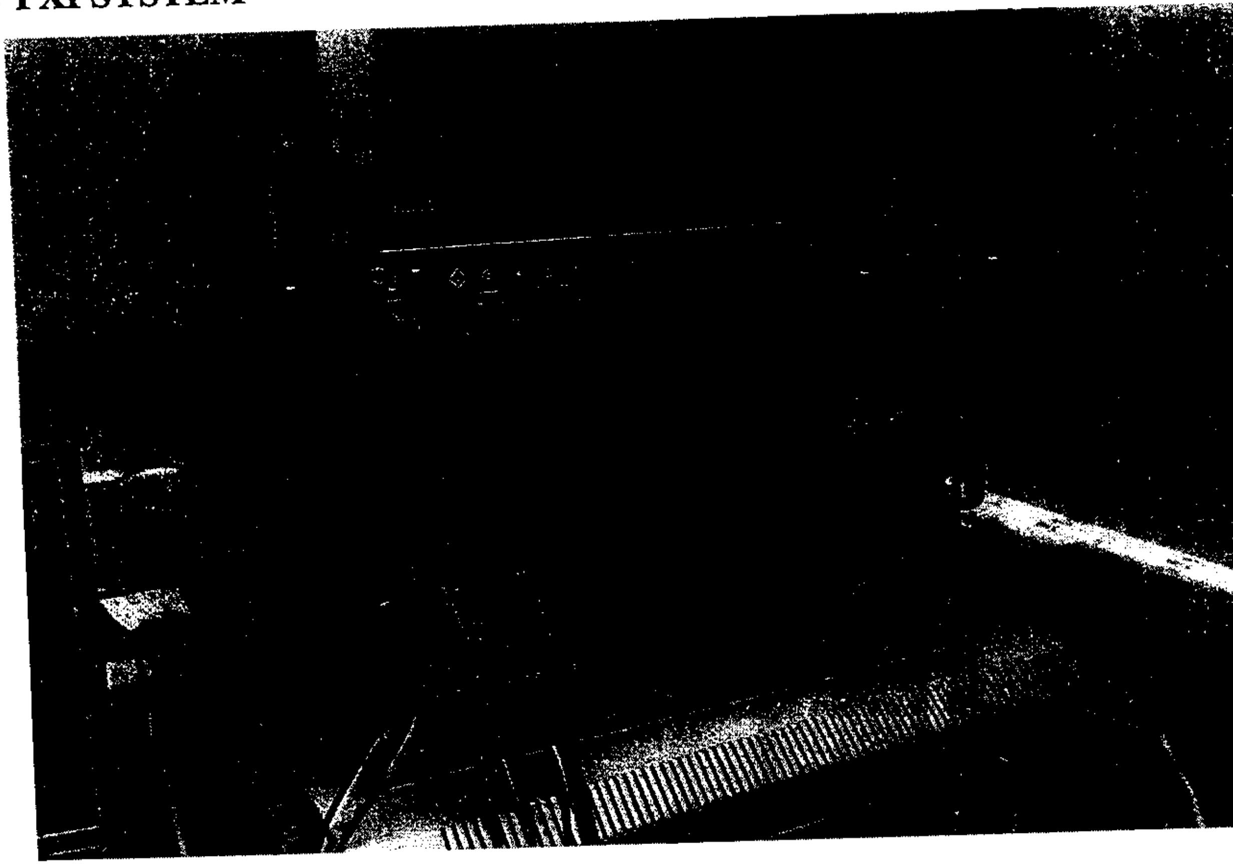


Figure 4.1 PXI Controller

PCI Extension for Instrumentation (PXI) specification is created in response to the needs of variety of instrumentation and automation users who require ever increasing integrate and use. Existing industry standards can be replaced by PXI at lower costs. By maintaining software compatibility with industry-standard personal computers, PXI allows industrial customers to use the same software tools and environments with which they are familiar. PXI has the electrical features of Peripheral Component Interconnect(PCI) specification with Euro card mechanical packaging and high-performance connectors.

The major advantages of PC based control system are,

- Low Cost(20% that of PLC system-since s/w is in house) and less complex.
- Easy portability with hardware updates
- Since the function of System program is fully understood it can be altered to suit the critical applications.
- Latest computer speed and Operating systems performance can be fully utilized.
- Human machine interface programs developed by us can be given highest priority for manual command.
- Ease of trouble shooting and user friendly diagnostics interface

4.3 SYSTEM CHECKING

The following tests are conducted in the LMF control system

4.3.1 INTEGRITY CHECK

The Integrity checks of cable wiring for each channel, resistance and continuity checks are made.

4.3.2 FUNCTIONAL CHECK

The control system is functionally checked by executing predefined auto sequence programs and also manual commands issued through mimic pages. Corresponding command acknowledgement and status feedback are also verified.

4.3.3 TIME ACCURACY CHECK

The overall timing accuracy of the system is checked by executing auto sequence programs. The data is acquired for about 1000 seconds with low speed DAS(1000 samples per second) and high speed DAS(10000 samples per second).

Time drift over 1000 sec

Low speed DAS - 7msec

High speed DAS – 11msec

4.4 SPECIFICATIONS

4.4.1 MULTIPIN CONNECTOR

Make	: Radial
Type	: Crimp
Model	
Male Portion	:MCSR113M00;
Female Portion	:Mcsr113FV
Cable Hood	:612846

4.4.2 MATERIALS

Insulator Block	:Glass filled Phenolic
Fixing Plate	:Stainless steel
Guide Pins	:stainless steel
Contacts	:Copper alloy – gold over nickel

4.4.3 ELECTRICAL CHARACTERISTICS

Current Rating	:13 A
Operation Voltage	:350 v.rms at 50Hz
Test Voltage	:1500V.rms at 50hz
Insulation Resistance	:Greater than 5000 Meg Ohm
Contact Resistance	:Less than milli Ohm

4.4.4 ENVIRONMENTAL CHARACTERISTICS

Contact Retention	:Greater than 50 N

4.4.5 MOUNTING PLATE

Material	:Mild Steel
Thickness	:3 mm
Finish	:Powder Coated
Colour	:Siemens Grey
Size	:H 1500 x W 490 mm

4.4.6 CONNECTOR PLATE

Material	:Mild Steel
Thickness	:2 mm
Finish	:Powder Coated
Colour	:Siemens Grey
Size	:H 150 x W 450 mm

4.4.7 EARTH PLATE

Material	:Mild Steel
Thickness	:2 mm
Finish	:Powder Coated
Colour	:Siemens Grey
Size	:H 100 x W 50 mm

4.5 PXI CHASSIS

- 19" 6U PXI Rack mountable system
- PXI specifications Rev.2.0 compliant
- 13 lines local bus for data transfer between modules
- Trigger bus with 10MHz system clock reference
- Star trigger bus for synchronization
- One system controller and 12 to 15 slots for PXI peripheral
- Number of PXI to PXI bridges: 2
- 2x250 W + 2x250 W redundant power supply with universal AC
- Input with 230V + 10%, Frequency 50Hz + 5%
- Front-access hot swappable fan trays for in-take and validation
    1. Air flow for in-take >220 CFM
    2. Air flow for ventilation >220 CFM
- System monitoring
    1. Power LED
    2. Temperature LED
    3. Fan LED

5. Alarm reset button

- Temperature setting options: 50 deg C(default), 60 deg C,70 deg C

4.6 SYSTEM CONTROLLER

- 6 U form factor
- PICMG 2.8 PXI specifications 2.0 compliant
- PICMG 2.0 R3.0, PICMG 2.16 R1.0
- Dual-core Intel Xeon Processor
- 667MHz FSB
- Intel E7520 with 6300ESB chipset
- 2 GB DDR 266 ECC Registered Memory module upgradable up to 4GB
- Display interface XVGA with resolution 1600 x 1200
- 250MHz Memory speed with 2D Graphics accelerator
- IDE, dual channels, one channel for 2.5" HDD on-board and other channel for compact Flash with 40 pin IDE connector on Rear I/O Transition Module
- Software driver for Linux/Windows
- Termination board (DIN rail mountable) with 100 pin connector and 3 meter link cable.

4.7 DIGITAL OUTPUT CARD:

- Compact PCI Compliant (PICMG 2.0 Rev 2.1)
- 6 U Euro card form factor
- Provision for 64 channel isolated digital outputs
- 64 channel optically individual isolated digital output
- Output with Darlington sink driver
- Sink Current : 500 mA for single channel @100% duty
- Supply Voltage : 5 to 35 V
- Isolation Voltage : 5000 Vrms
- Programmed I/O data transfer
- Software driver for Linux/ Windows
- Termination board(DIN rail mountable) with 100 pin connector and 3m link cable

CHAPTER 5
HMI (HUMAN MACHINE INTERFACE) DEVELOPMENT

5.1 MIMIC SYSTEMS

The development of AJAX based process MIMIC System is used to control the operations performed in the Test Facility using web based control systems. The users of the system may be an Operator ,Visitor or Administrator in the control room of test facility. We have 'n' number of MIMIC pages for our requirements. One of them provides the user login and authentication. Only a privilege-d user is allowed to operate the MIMIC.

5.2 TECHNOLOGIES USED

5.2.1 AJAX:

AJAX (Asynchronous JavaScript and XML)is about creating more versatile and interactive web application by enabling web pages to make asynchronous calls to the server transparently while the user is working. It balances the load between the client and the server by allowing them to communicate in the background while the user is working on the page.

Ajax's primary contribution to web pages is user-experience improvement. Web pages usually require several applications to function. This can make it seem like a cumbersome operation where users have to wait for the separate applications to refresh before interacting with the complete page. Decreasing user delay, which is a direct result of Ajax technique, could make the Internet even more popular and pervasive than it already is.

Another advantage of Ajax is a decrease in bandwidth use. Bandwidth in web hosting refers to the amount of data that can be communicated between user and server/website. In Ajax, bandwidth is used only to accomplish a specific demand without requiring that the page be reloaded (which requires bandwidth, every time a request is made). Contents are loaded on demand and HTML is produced locally from the browser. Ajax also allows programmers to separate methods and formatting for specific information delivery functions on the Web.

5.3 THE CLIENT SIDE

5.3.1 HTML (HYPER TEXT MARKUP LANGUAGE)

HTML is the basic web page development language which creates static web pages. HTML along with scripting language provides dynamic web pages.

The html file includes either the actual field image or an integrated images of the valves, which will displays the status information from the field, and also used to give the command to the field. when body of this file loads it will automatically executes the JavaScript function from the mentioned JavaScript file.

If the MIMIC is used for the login purpose, the main page displays the login and authentication field, and the indication for the three users Administrator, Visitor and the Operator. The page will drives us to the actual MIMIC pages for the fields.

5.3.2 JAVASCRIPT

JavaScript is a scripting language created by Netscape. JavaScript can be inserted into a web page to add functionality. JavaScript is much easier to use and often found in wed development. JavaScript is supported by Netscape and Microsoft web browsers. The real trick of updating the segment of a web page without actually having to reload the entire page is often done by utilizing a JavaScript property known as inner HTML.

The file contains the main function which executes when the html file loads. It consist of a set of code segments which works together for the optimum performance. The segments includes

- Create an instance of the XMLHttpRequest object.
- Use the XMLHttpRequest object to make an asynchronous call to the server page, defining a call back function that will be executed automatically when the server response is received
- Deal with the server's response in the call back function.

5.3.3 CREATE AN INSTANCE OF XMLHTTPREQUEST OBJECT

The XMLHttpRequest is implemented in different way by browsers. In Internet Explorer 6 and older XMLHttpRequest is implemented as an ActiveX control, for other browsers it is native object.

5.3.4 MAKE A CALL TO THE SERVER

The methods we will use with every server request are open and send. The open method configures a request by setting various parameters, and send makes the request(accesses the server).When the request is made asynchronously, before call you will also need to set onreadystatechange property with the call back method is to be executed when the status of the request changes, Thus enabling AJAX mechanism.

The open method is used for initializing a request. It has two required parameters and few optional ones. The open method doesn't initiate a connection to the server. Its only used to set the connection options. The first parameter specifies the method used to send the data to the server page, and it can have the value GET,POST,or PUT. The second parameter is the URL, which specifies where we want to send the request.

This can be complete or relative. If the URL doesn't specify a resource that can be accessible via HTTP, the first parameter is ignored. The third parameter is async ,if its 'true' the script processing carries on after the send() method returns without waiting for the response, 'false' means that script waits for the response before continuing processing freezing the web page functionality.

5.3.5 HANDLING SERVER RESPONSE

The HandleServerResponse is a call back method that we set to handle request state changes. Usually this is called four times, for each time the request enters a new state. The ready state property can be

0-uninitialized

1-loading

2-loaded

3-interactive

4-complete

In AJAX application we use only complete state, which marks the response has been received from the server. We can use try-catch method to handle errors that could happen while initiating a connection to the server, or while reading the response from the server. After reading the response from the server, we can utilize the response to fulfill our

5.4 THE SERVER SIDE

5.4.1 PHP (HYPER TEXT PREPROCESSOR)

PHP is best summarized as embedded server-side Web scripting language that provides developers with the capability to quickly and efficiently build dynamic Web applications. Providing hundreds of pre defined functions,PHP is capable of handling just about anything a developer can dream of. Extensive support is offered for graphic creation and manipulation, mathematical calculations, ecommerce and burgeoning technologies such as XML(Extensible Markup language),ODBC(Open Database Connectivity), and macro media shock wave. One of the main strengths of PHP is that it can be embedded directly alongside HTML code, there is no need to write a program that has many commands just to output the HTML. Five important characteristics make PHP's practical nature possible.

- Familiarity
- Simplicity
- Efficiency
- Security
- Flexibility

5.4.1.1 FAMILIARITY

Many of the languages constructs are borrowed from C and Perl, and in many cases PHP code is almost indistinguishable from that found in the typical C or Pascal program. This minimizes the learning curve considerably.

5.4.1.2 SIMPLICITY

There is no need to include libraries, special compilation directives, or anything of sort. The PHP engine simply begins executing the code after the first escape sequence (<?) and continues until it passes the closing escape sequence(?>). If the code is syntactically correct, it will be executed exactly as it is displayed.

5.4.1.3 EFFICIENCY

Efficiency is an extremely important consideration when working in a multiuser environment. PHP introduce resource allocation mechanisms and more pronounced support

5.4.1.4 SECURITY

PHP can be run in what is known as safe mode, can limit users' attempts to exploit the PHP implementation in many important ways.Limits can also be placed on maximum execution time and memory usage. Several tested data encryption option are supported in PHP's pre defined function set. PHP source code is not viewable through the browser because the script is completely parsed before it sent back to the requesting user.

5.4.1 5 FLEXIBILITY

Because PHP is an embedded language, it is extremely flexible towards meeting the needs of the developer. Although PHP is generally touted as being used in conjunction solely with HTML , it can also be integrated alongside languages like Javascript,WML,XML,and many others.

On the web server the php file build the XML message to the client. This XML message consists of <response> that packages the message the server needs to send back to the client. The php file script generate the response with XML header and <response> tag. the text returned to the server is encapsulated by <response> element.

5.4.2 MYSQL

One of the world's most popular open source databases, MySQL Server powers many of the largest, high volume Websites and business critical systems. A fully integrated, transaction safe, ACID complaint database with commit, rollback, crash recovery, and row level looking capabilities, MySQL Server is a reliable foundation for e-commerce, Online Transaction Processing(OLTP),and data warehousing solutions. MySQL Server creates a dedicated user thread running at normal priority in the thread pools for each simultaneous client request. This dedicated user thread processes the user request and send back the result to each client once the result is ready. An additional, single user thread waits for input from the console , and a group of utility threads running at lower priority handle some background tasks.

5.5 HOW DOES AJAX WORK?

The core idea behind AJAX is to make the communication with the server can continue working on the other parts of the page without interruption. In an AJAX-enabled application only the relevant page elements are updated, only when this is necessary.

The AJAX-enabled applications, on the other hand, rely on a new asynchronous method of communication between the client and the server. It is implemented as a JavaScript engine that is loaded on the client during the initial page load. From there on, this engine serves as a mediator that sends only relevant data to the server as XML and subsequently processes server response to update the relevant page elements.

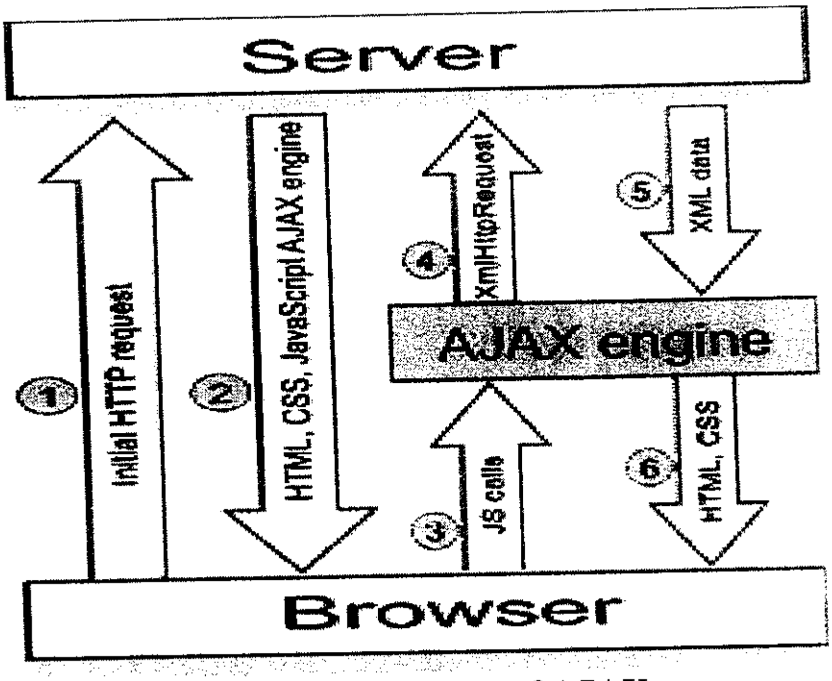


Figure 5.1 Working of AJAX

1. Initial request by the browser – the user requests the particular URL. The complete page is rendered by the server (along with the JavaScript AJAX engine) and sent to the client (HTML, CSS, JavaScript AJAX engine).
2. All subsequent requests to the server are initiated as function calls to the JavaScript engine.
3. The JavaScript engine then makes an XMLHttpRequest to the server.
4. The server processes the request and sends a response in XML format to the client (XML document). It contains the data only of the page elements that need to be changed. In most cases this data comprises just a fraction of the total page markup.
5. The AJAX engine processes the server response, updates the relevant page content or performs another operation with the new data received from the server. (HTML + CSS).

CHAPTER 6
SOCKET PROGRAMMING

Socket is an interface between an application process and transport layer. The application process can send/receive messages to/from another application process(local or remote) via a socket.

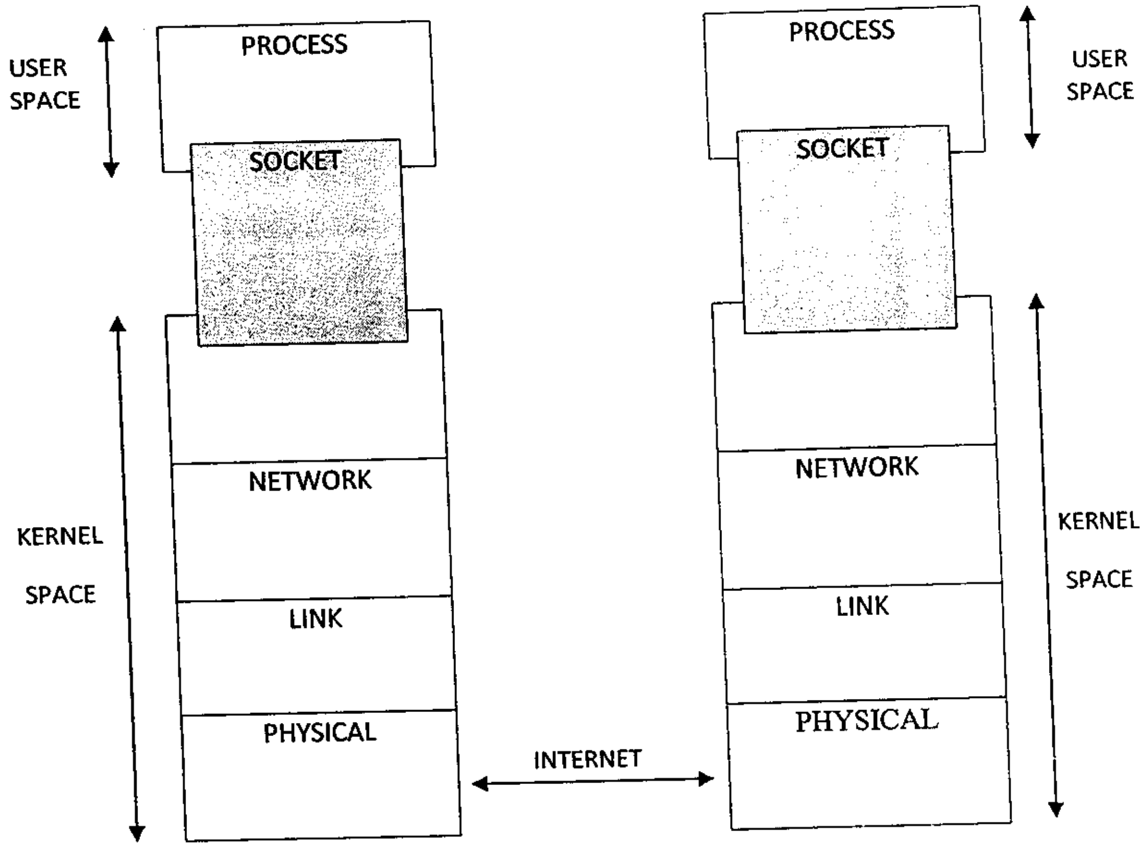


Figure 6.1 Socket Description

Sockets commonly are used for client/server interaction. Typical system configuration places the server on one machine, with the clients on other machines. The clients connect to the server, exchange information, and then disconnect. The PC based control system can also be modeled as a client-server model. Hence in this system socket programming is used for communication between different systems.

The types of sockets are,

1. Internet sockets

In PC based control system Internet sockets are used. They are characterized by IP address(4 bytes) and port numbers(2 bytes). The use of port numbers allows many simultaneous connections between the hosts with single IP address.

6.1 TYPES OF INTERNET SOCKETS

There are two types of Internet sockets available. They are,

1. Stream socket
2. Datagram socket

Stream sockets are otherwise called as "connection oriented sockets" and they rely on TCP protocol to provide reliable two-way connected communication. These sockets are used when the communication should be error free.

Datagram sockets are otherwise called as "connectionless sockets" and they rely on UDP protocol. These sockets are used when the communication is not needed to be an error free one and when the connection is unreliable.

The following functions are used in the socket programming.

1. socket() creates a new socket of a certain socket type, identified by an integer number, and allocates system resources to it.
2. bind() is typically used on the server side, and associates a socket with a socket address structure, i.e. a specified local port number and IP address.
3. listen() is used on the server side, and causes a bound TCP socket to enter listening state.
4. connect() is used on the client side, and assigns a free local port number to a socket. In case of a TCP socket, it causes an attempt to establish a new TCP connection.
5. accept() is used on the server side. It accepts a received incoming attempt to create a new TCP connection from the remote client, and creates a new socket associated with the socket address pair of this connection.
6. send() and recv(), or write() and read(), or sendto() and recvfrom(), are used for sending and receiving data to/from a remote socket.
7. close() causes the system to release resources allocated to a socket. In case of TCP, the connection is terminated.
8. gethostbyname() and gethostbyaddr() are used to resolve host names and addresses.
9. select() is used to prune a provided list of sockets for those that are ready to read, ready to

written to, read from or has errors.

11. getsockopt() is used to retrieve the current value of a particular socket option for the specified socket.
12. setsockopt() is used to set a particular socket option for the specified socket.

6.2 BLOCKING VS. NON-BLOCKING SOCKETS:

Sockets can operate in one of two modes: blocking or non-blocking. A blocking socket will not return control until it has sent (or received) all data specified for the operation. It also may cause problems if a socket continues to listen: a program may hang as the socket waits for data that may never arrive. So that in a client-server system, if the the client does not receive any byte during executing the control cycle, then the client socket will wait in that loop and will never come out of that loop. So the following bytes that are sent by the server will also not be received by the client and the whole communication will get affected. Hence to overcome this problem, non-blocking sockets are used.

In the case of non-blocking sockets, the socket can be made to wait in the receive/write mode for some particular time. If no data is available to send/ receive within that particular time, then the socket will come out of that loop and go into the next loop. These non-blocking sockets are used in PC based control system.

A socket is typically set to blocking or nonblocking mode using the fcntl() or ioctl() functions.

6.3 FUNCTIONS USED IN SOCKET PROGRAMMING:

socket()

socket() creates an endpoint for communication and returns a file descriptor for the socket. socket() takes three arguments:

- *domain*, which specifies the protocol family of the created socket. For example:
  - PF_INET for network protocol IPv4 or
  - PF_INET6 for IPv6.
  - PF_UNIX for local socket (using a file).
- *type*, one of:
  - SOCK_STREAM (reliable stream-oriented service or Stream Sockets)
  - SOCK_DGRAM (datagram service or Datagram Sockets)

- SOCK_RAW (raw protocols atop the network layer).

- *protocol*, specifying the actual transport protocol to use. The most common are IPPROTO_TCP, IPPROTO_SCTP, IPPROTO_UDP, IPPROTO_DCCP. These protocols are specified in <netinet/in.h>. The value "0" may be used to select a default protocol from the selected domain and type.

The function returns -1 if an error occurred. Otherwise, it returns an integer representing the newly-assigned descriptor.

Prototype:

int socket(int domain, int type, int protocol);

bind()

bind() assigns a socket to an address. When a socket is created using socket(), it is only given a protocol family, but not assigned an address. This association with an address must be performed with the bind() system call before the socket can accept connections to other hosts. bind() takes three arguments:

- sockfd, a descriptor representing the socket to perform the bind on.
- my_addr, a pointer to a sockaddr structure representing the address to bind to.
- addrlen, a socklen_t field specifying the size of the sockaddr structure.

  Bind() returns 0 on success and -1 if an error occurs.

Prototype:

int bind(int sockfd, const struct sockaddr *my_addr, socklen_t addrlen);

listen()

After a socket has been associated with an address, listen() prepares it for incoming connections. However, this is only necessary for the stream-oriented (connection-oriented) data modes, i.e., for socket types (SOCK_STREAM, SOCK_SEQPACKET). listen() requires two arguments:

- sockfd, a valid socket descriptor.
- backlog, an integer representing the number of pending connections that can be queued up at any one time. The operating system usually places a cap on this value. Once a connection is accepted, it is dequeued. On success, 0 is returned. If an error occurs, -1 is returned.

Prototype:

int listen(int sockfd, int backlog);

accept()

When an application is listening for stream-oriented connections from other hosts, it is notified of such events (cf. select() function) and must initialize the connection using the accept() function. Accept() creates a new socket for each connection and removes the connection from the listen queue. It takes the following arguments:

- sockfd, the descriptor of the listening socket that has the connection queued.
- cliaddr, a pointer to a sockaddr structure to receive the client's address information.
- addrlen, a pointer to a socklen_t location that specifies the size of the client address structure passed to accept(). When accept() returns, this location indicates how many bytes of the structure were actually used.

The accept() function returns the new socket descriptor for the accepted connection, or -1 if an error occurs. All further communication with the remote host now occurs via this new socket.

Datagram sockets do not require processing by accept() since the receiver may immediately respond to the request using the listening socket.

Prototype:

int accept(int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen);

connect()

The connect() system call connects a socket, identified by its file descriptor, to a remote host specified by that host's address in the argument list.

Certain types of sockets are connectionless, most commonly user datagram protocol sockets. For these sockets, connect takes on a special meaning: the default target for sending and receiving data gets set to the given address, allowing the use of functions such as send() and recv() on connectionless sockets.

connect() returns an integer representing the error code: 0 represents success, while -1 represents an error.

Prototype:

(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen);

gethostbyname() and gethostbyaddr()

The gethostbyname() and gethostbyaddr() functions are used to resolve host names and addresses in the domain name system or the local host's other resolver mechanisms (e.g., /etc/hosts lookup). They return a pointer to an object of type struct hostent, which describes an Internet Protocol host. The functions take the following arguments:

- name specifies the name of the host. For example: www.wikipedia.org
- addr specifies a pointer to a struct in_addr containing the address of the host.
- len specifies the length, in bytes, of addr.
- type specifies the address family type (e.g., AF_INET) of the host address.

The functions return a NULL pointer in case of error, in which case the external integer h_errno may be checked to see whether this is a temporary failure or an invalid or unknown host. Otherwise a valid struct hostent * is returned.

These functions are not strictly a component of the BSD socket API, but are often used in conjunction with the API functions. Furthermore, these functions are now considered legacy interfaces for querying the domain name system. New functions that are completely protocol-agnostic (supporting IPv6) have been defined. These new function are getaddrinfo() and getnameinfo(), and are based on a new addrinfo data structure.

Prototypes:

```
struct hostent *gethostbyname(const char *name);
struct hostent *gethostbyaddr(const void *addr, int len, int type);
```

send() and recv()

These two functions are for communicating over stream sockets

Prototypes:

```
int send(int sd,const void *msg,int len,int flags)
int recv(int sd,void *buf,int len,int flags)
```

sendto() and recvfrom()

These two functions are used for communicating over datagram sockets.

```
int flags const struct sockaddr *to,
```

int recvfrom(int sockfd, void *buf, int len, unsigned int flags,struct sockaddr *from, int *fromlen);

close() and shutdown()

These two functions are used for closing the communication and to make the socket free so that it coula be used for other applications.

Prototypes:

close(sockfd);

int shutdown(int sockfd, int how);

sockfd is the socket file descriptor you want to shutdown, and *how* is one of the following:

0 Further receives are disallowed

1 Further sends are disallowed

2 Further sends and receives are disallowed

fcntl()

This function is used to make the socket as non blocking socket.

Prototype:

int fcntl(int sockfd , F_SETFL , O_NONBLOCK)

select()

This function is used to make the socket to wait in the read mode/write mode/ exceptional mode for some desired time.

Int select(int sockfd, fd_set *read_fds, fd_set *write_fds, fd_set *except_fds, struct timeval *timeout);

6.4 CLIENT-SERVER MODEL USING TCP:

The Transmission Control Protocol (TCP) provides the concept of a connection, which is a stateful network association between two hosts with a variety of error correction and performance features. A process creates a TCP socket by calling the socket() function with the parameters for the protocol family (PF_INET, PF_INET6), SOCK_STREAM (Stream Sockets) and the IP protocol identifier IPPROTO_TCP.

6.4.1 SERVER SIDE:

Setting up a simple TCP server involves the following steps

- Creating a TCP socket, with a call to socket().
- Binding the socket to the listen port, with a call to bind(). Before calling bind(), a programmer must declare a sockaddr_in structure, clear it (with memset()), and the sin_family (AF_INET), and fill its sin_port (the listening port, in network byte order) fields. Converting a short int to network byte order can be done by calling the function htons() (host to network short).
- Preparing the socket to listen for connections (making it a listening socket), with a call to listen().
- Accepting incoming connections, via a call to accept(). This blocks until an incoming connection is received, and then returns a socket descriptor for the accepted connection. The initial descriptor remains a listening descriptor, and accept() can be called again at any time with this socket, until it is closed.
- Communicating with the remote host, which can be done through send() and recv() or write() and read().
- Eventually closing each socket that was opened, once it is no longer needed,using close(). Note that if there were any calls to fork(), each process must close the sockets it knew about (the kernel keeps track of how many processes have a descriptor open), and two processes should not use the same socket at once.

6.4.2 CLIENT SIDE:

Setting up a TCP client involves the following steps:

- Creating a TCP socket, with a call to socket().
- Connecting to the server with the use of connect(), passing a sockaddr_in structure with the sin_family set to AF_INET, sin_port set to the port the endpoint is listening (in network byte order), and sin_addr set to the IP address of the listening server (also in network byte order.)
- Communicating with the server by using send() and recv() or write() and read().
- Terminating the connection and cleaning up with a call to close(). Again, if there

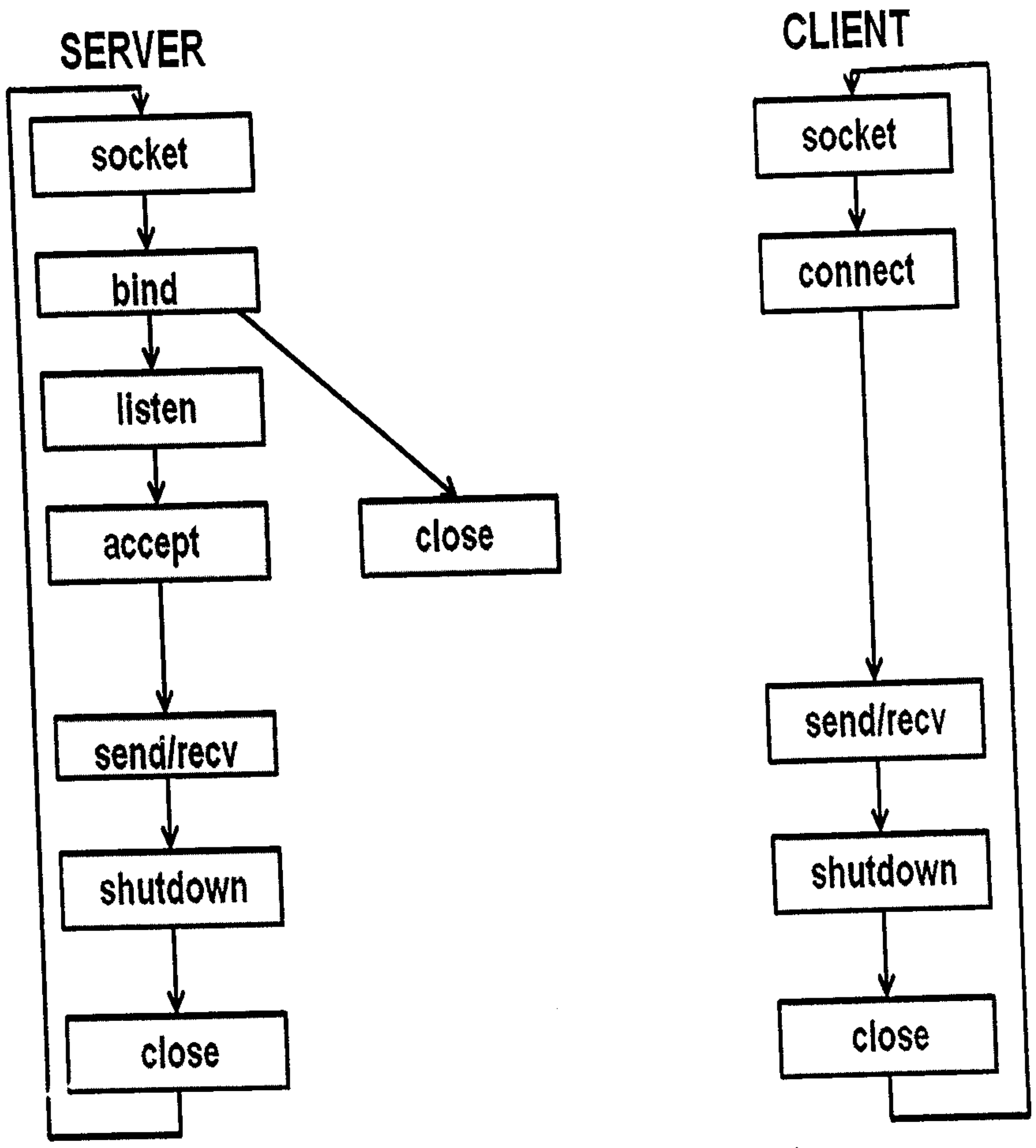
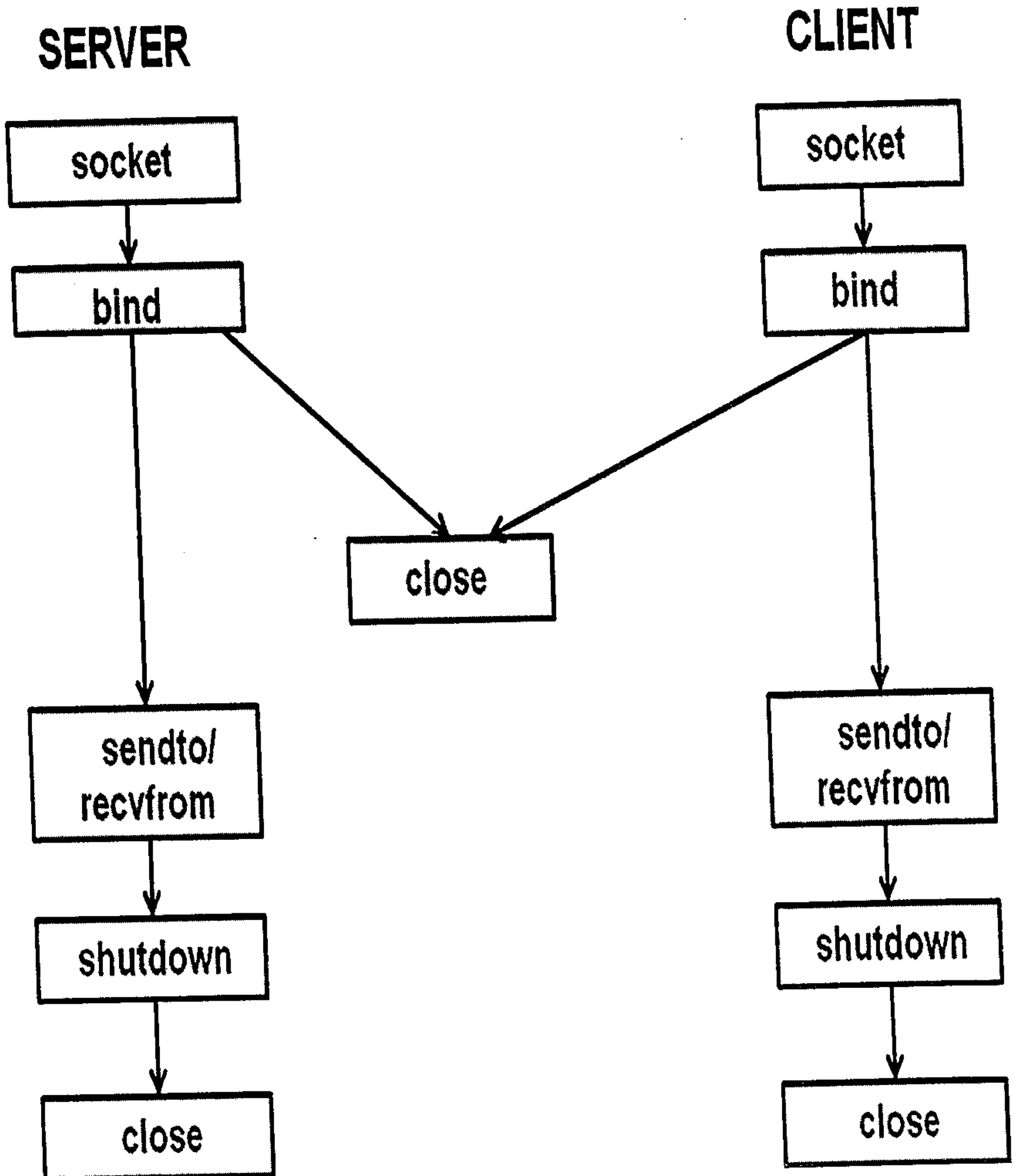


Figure 6.2 TCP based Socket Programming

6.5 UDP BASED SOCKET PROGRAMMING:

The udp based socket programming is based on the use of connectionless sockets between two systems. After creating the datagram sockets, the two applications can communicate with each other using the sendto/ recvfrom functions. The functions used in UDP based socket programming can be well explained by the following diagram Fig.6.3



CHAPTER 7
NEED FOR REDUNDANCY

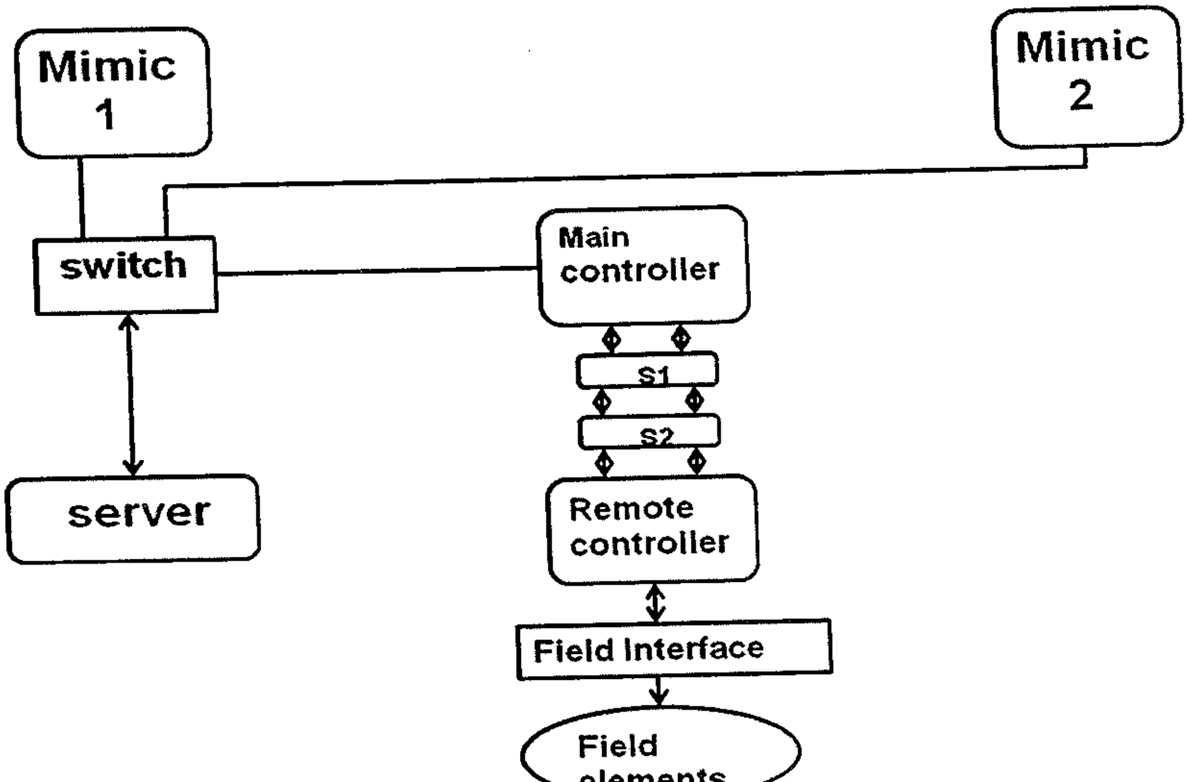
The implemented PC based control system is working without any redundancy ie. if there is any failure in the control system or in the communication link, it will affect the whole system and the system testing will fail immediately. Since the testing involves large amount of money and other sources, this failure should be avoided by providing a proper redundancy to the PC based control system.

7.1 PHASES OF REDUNDANCY

There are two phases of redundancy that can be developed in PC based control system. They are,

1. Network Redundancy with single system
2. Network Redundancy with redundant system

7.1.1 NETWORK REDUNDANCY WITH SINGLE SYSTEM



In the PC based control system, there is a single TCP/IP communication installed between the main controller and remote controller. This communication link is identified as the weakest element in the system. Since the main controller plays the major role in the control system, during testing if this communication link fails, it will result in the failure of the whole system and the system operation will get collapsed. Hence this communication link is strengthened by developing a redundant communication link between the main controller and remote controller in such a way that if one link fails, the other link will automatically carry over the communication and the system will continue to work without any disturbance in the operation. The flow chart for developing network redundancy is given in Fig 7.2.

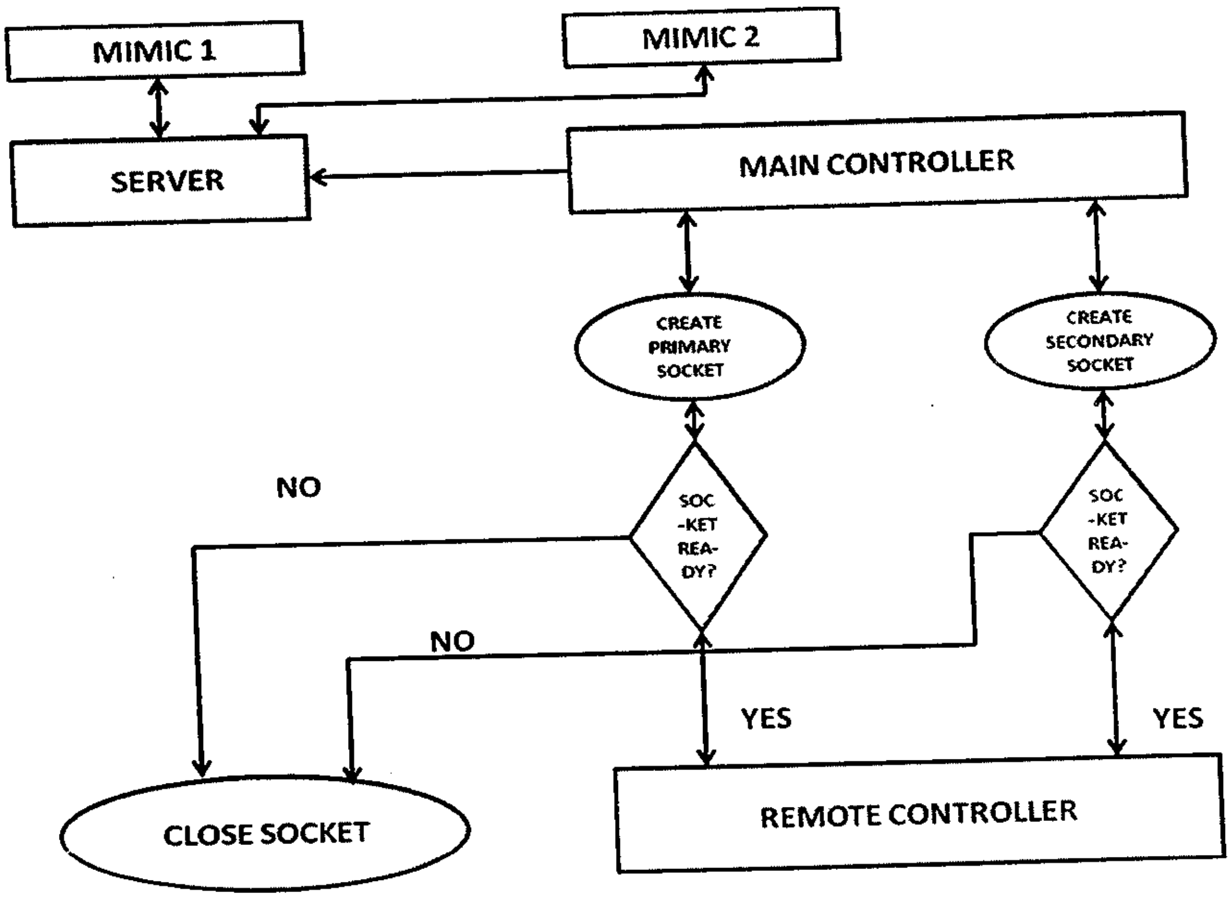


Figure 7.2 Flow Chart for Developing Network Redundancy with Single System

For developing network redundancy two tcp sockets are created in the main controller side and remote controller side for two different tcp links with different IP addresses. In auto mode of operation, the main controller will send the command string to the remote controller remote controller will use the command string from the primary link for communicating with the field elements and get back the feedback data. Once the primary link gets failed, the remote controller will detect it by comparing the received data from both the sockets, and use the data from secondary socket for communicating with the field elements. The primary socket will be closed once the primary link gets failed, so that the primary link will not affect the further communication even it becomes active again. Similarly if the secondary link fails, the corresponding socket will be closed immediately and so it will not affect the further communication using the primary socket.

7.1.2 NETWORK REDUNDANCY WITH REDUNDANT SYSTEM

The network redundancy involves the development of the redundant link between main controller and remote controller. It includes single main controller and single remote controller. It can provide protection against the failure of the communication link in the PC based control system. But if the main controller or remote controller itself gets failed, then it cannot give protection to the system operation. Hence in order to give additional strength to the control system, second type of redundancy namely system redundancy has to be developed in PC based control system.

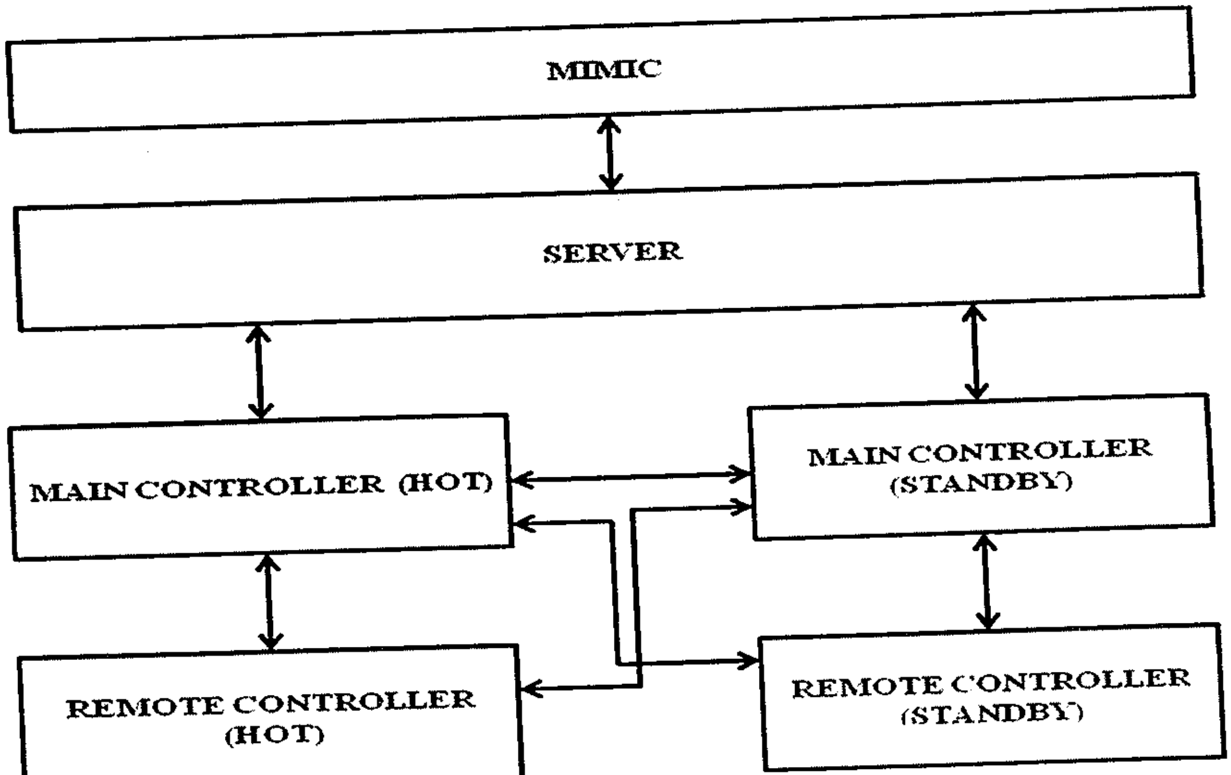


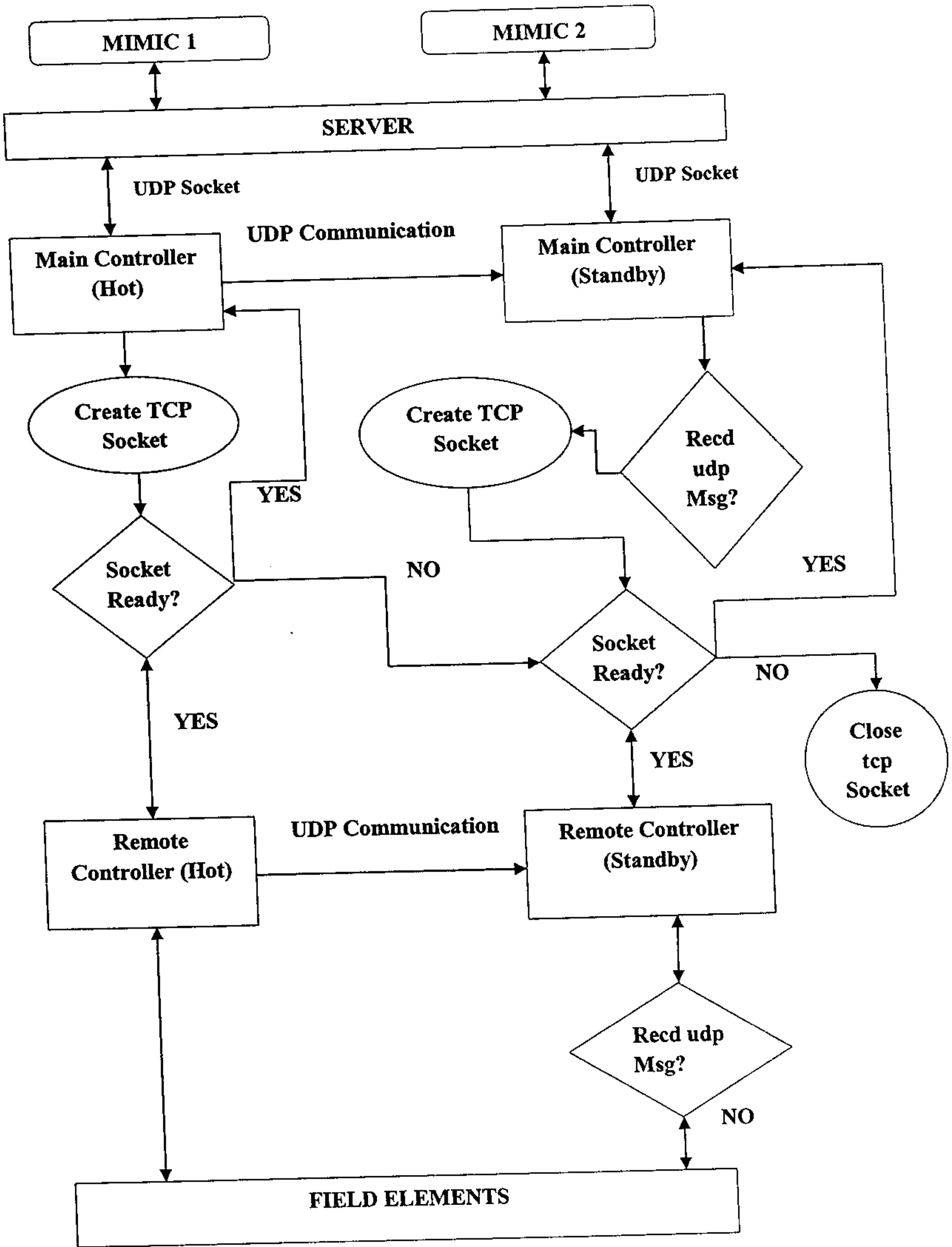
Fig 7.3 shows the development of system redundancy. It includes single server, two main controllers and two remote controllers. Here each controller is made to communicate with all the other controllers. The primary controllers are called as hot controllers and secondary controllers are called as standby controllers. TCP communication is used between the main controllers and remote controllers and also between hot remote controller and standby remote controller. UDP communication is used between the hot main controller and standby main controller.

The hot main controller will send the command string to both the remote controllers using tcp communication. It will also send a heart beat signal to the standby main controller using udp communication, to inform it that the hot main controller is active. The standby main controller will also send the command string to both the remote controllers. As long as the hot main controller is active, the remote controllers will use only the command string from the hot main controller for executing in the field.

Also the feedback will be sent back only to the hot main controller and finally it will reach the MIMIC display via the server. Once the hot main controller gets failed, the remote controllers will detect the failure and they will use the command string coming from the standby main controller for communicating with the field elements. The failure of the hot main controller will be known by the standby main controller by not receiving the heartbeat signal and so it will start to communicate with the server from the next cycle.

Similarly in the receiving side, tcp communication is used between the hot remote controller and standby remote controller for sending the heartbeat signal. As long as the hot remote controller is active, the standby remote controller will not send the received command sting to the filed elements for execution. Once the standby remote controller stopped to receive the heart beat signal from the hot remote controller, it will decide that the hot remote controller is not active anymore and it will start to communicate with the field elements. So that the whole system failure can be avoided even if one of the controllers gets failed.

The flow chart for developing network redundancy is given in Fig.7.4



Initially three sockets will be created in each of the main controller. two sockets are tcp sockets and the third socket is the udp socket. The tcp sockets are used for sending the command string to both the remote controllers and udp socket is used for sending the heartbeat signal from main controller to remote controller. Similarly in the remote controller side, three tcp sockets will be created in each controller. Two sockets will be used for sending the feedback to the main controllers and other socket is used for sending the heartbeat signal from the hot remote controller to standby remote controller.

For the purpose of establishing tcp connection between the main controllers and remote controllers, initially the hot main controller will wait for 10 seconds to get the connection request from the hot remote controller using one of the tcp sockets. If it receives the connection request within this particular time, it will accept that request and establish the tcp connection with the hot remote controller. If it is not receiving the connection request from the hot remote controller within this ten seconds. Then it will not wait in the listen function for infinite time. Since non blocking sockets are used here, after ten seconds the control will come out of that listen function and the hot main controller will start to wait for the connection request from the standby remote controller for the next ten seconds using the other tcp socket. Again if it receives the connection request from the standby remote controller, it will accept the request and establish the connection. Otherwise it will come out of that listen function after ten seconds. Even if one of the remote controllers gets connected with the main controller, the main controller will start to execute the control cycle. But if both the remote controllers are not getting connected means, the main controller will again wait in the loop for getting connection request from the remote controllers. Once it receives the connection request, it will come out of the loop and starts communication. In the standby main controller side also the same kind of operations will be carried out for establishing connection with the remote controllers.

At the time of starting the control system suppose if the standby main controller is not ready, then the system can be made to operate only with the hot main controller. Similarly if the hot main controller is not ready, the system can be made to operate only with the standby main controller. Similar condition is applicable for remote controller side also. The sequence

CHAPTER 8
SIMULATION RESULTS

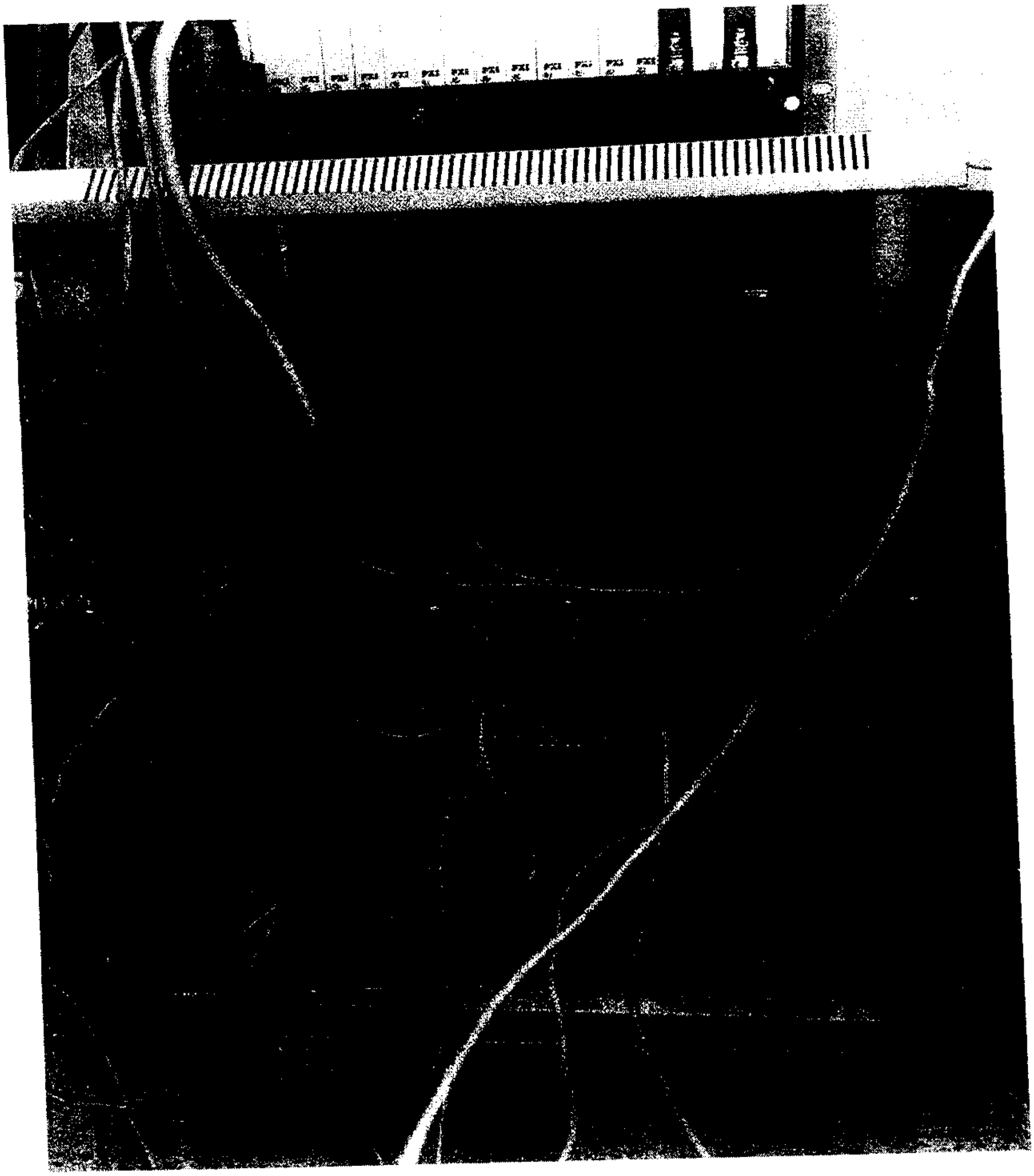


Figure 8.1 Main Controller – PXI System

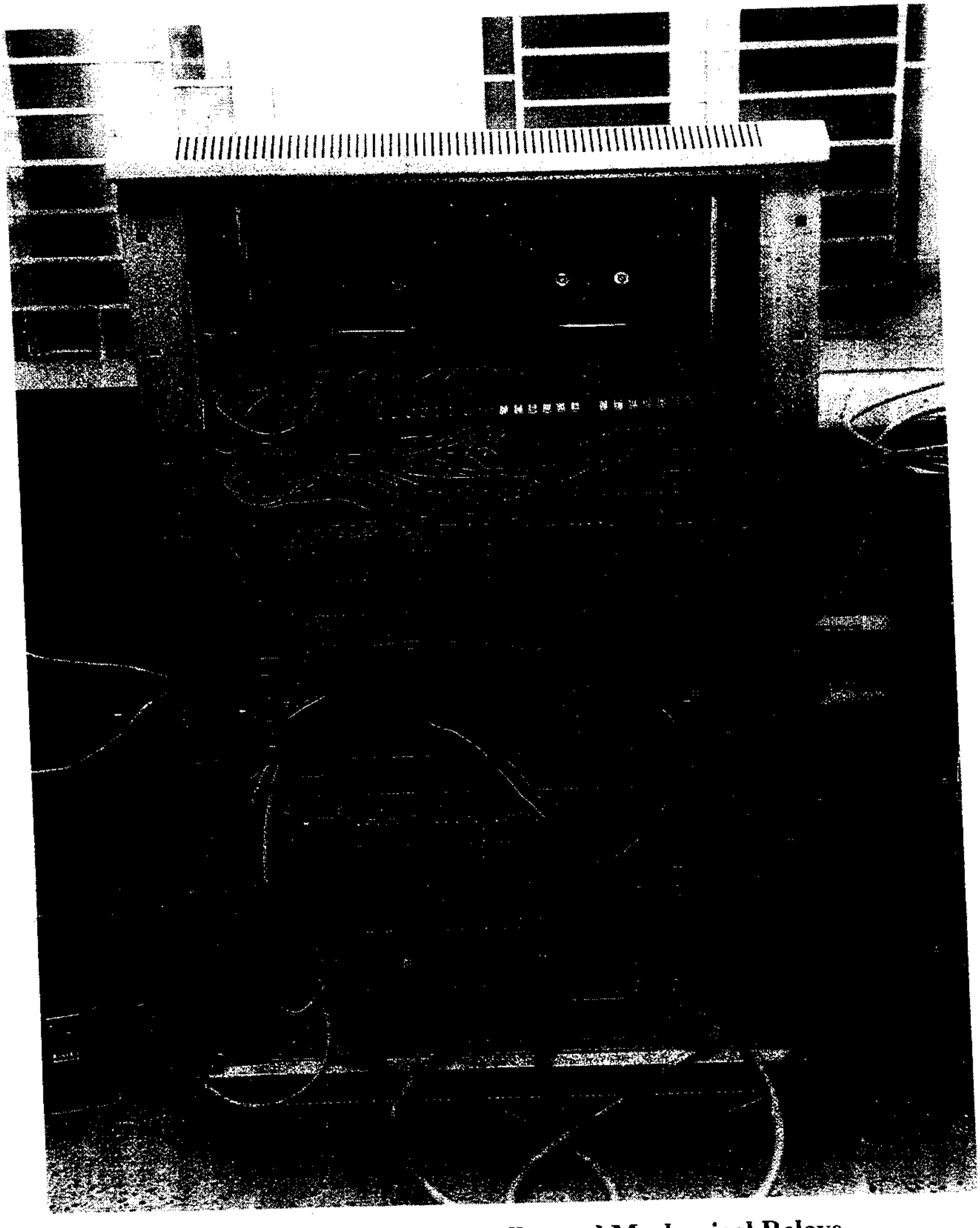
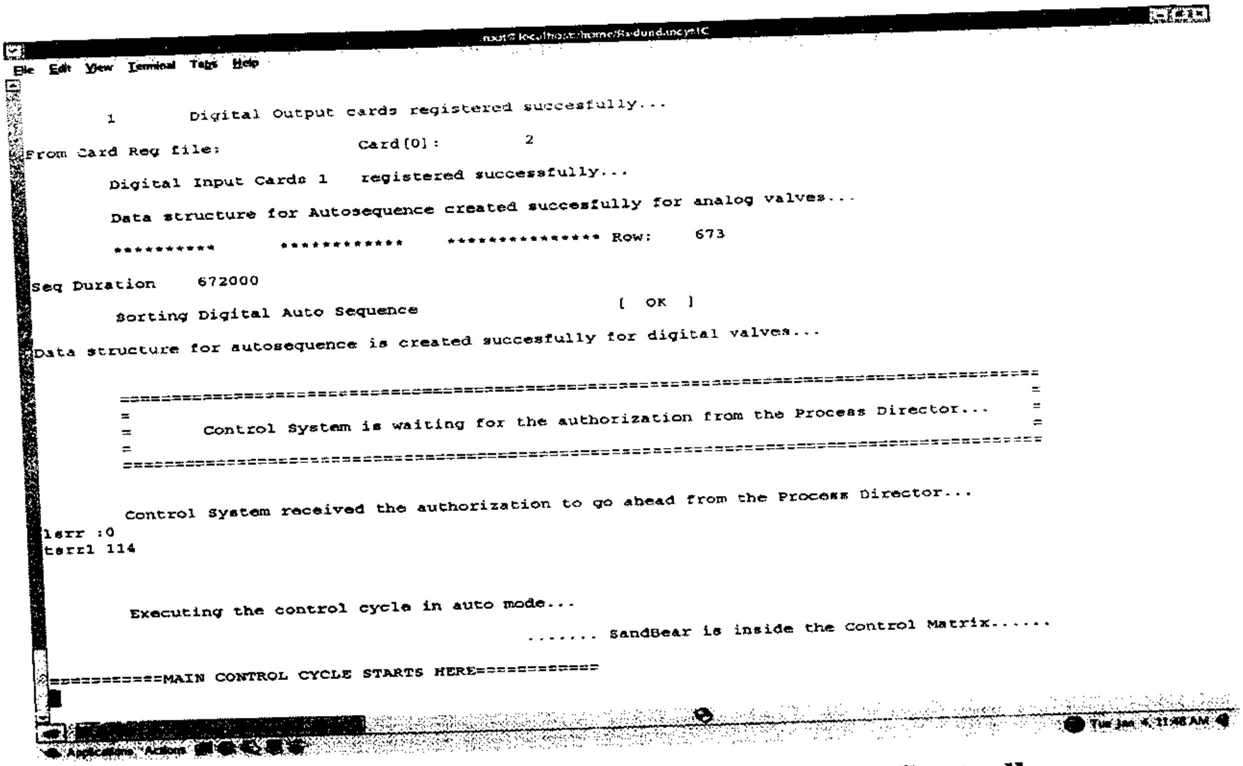


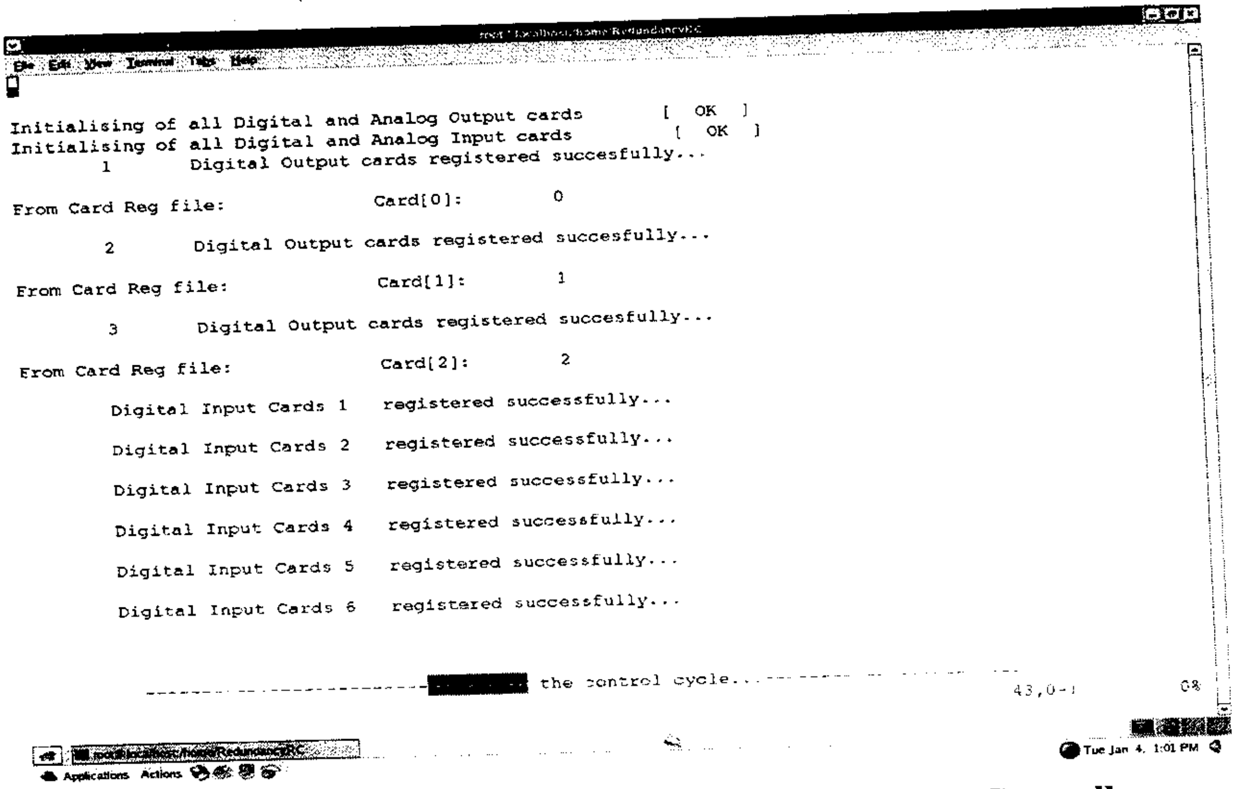
Figure 8.2 Remote Controller and Mechanical Relays

8.1 NETWORK REDUNDANCY WITH SINGLE SYSTEM

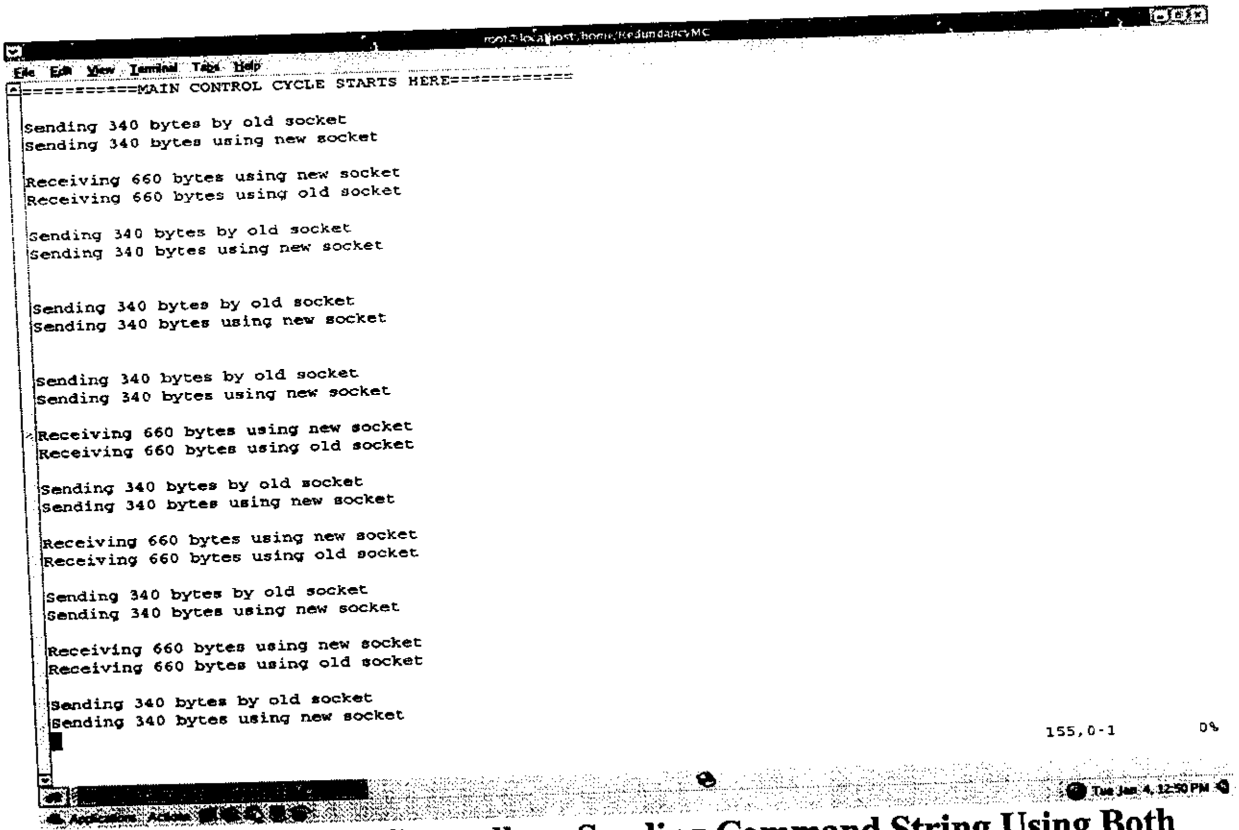


```
        1        Digital Output cards registered succesfully...
From Card Reg file:               Card[0]:        2
        Digital Input Cards 1    registered successfully...
        Data structure for Autosequence created succesfully for analog valves...
        **********        ************       ************** Row:     673

Seq Duration     672000
        Sorting Digital Auto Sequence                      [  OK  ]
Data structure for autosequence is created succesfully for digital valves...

        ===========================================================================
        =                                                                         =
        =        Control System is waiting for the authorization from the Process Director...   =
        =                                                                         =
        ===========================================================================

        Control System received the authorization to go ahead from the Process Director...
lerr :0
terrl 114


        Executing the control cycle in auto mode...
                                          ....... SandBear is inside the Control Matrix......

===========MAIN CONTROL CYCLE STARTS HERE============
```

Figure 8.3 Start of Execution at the Main Controller

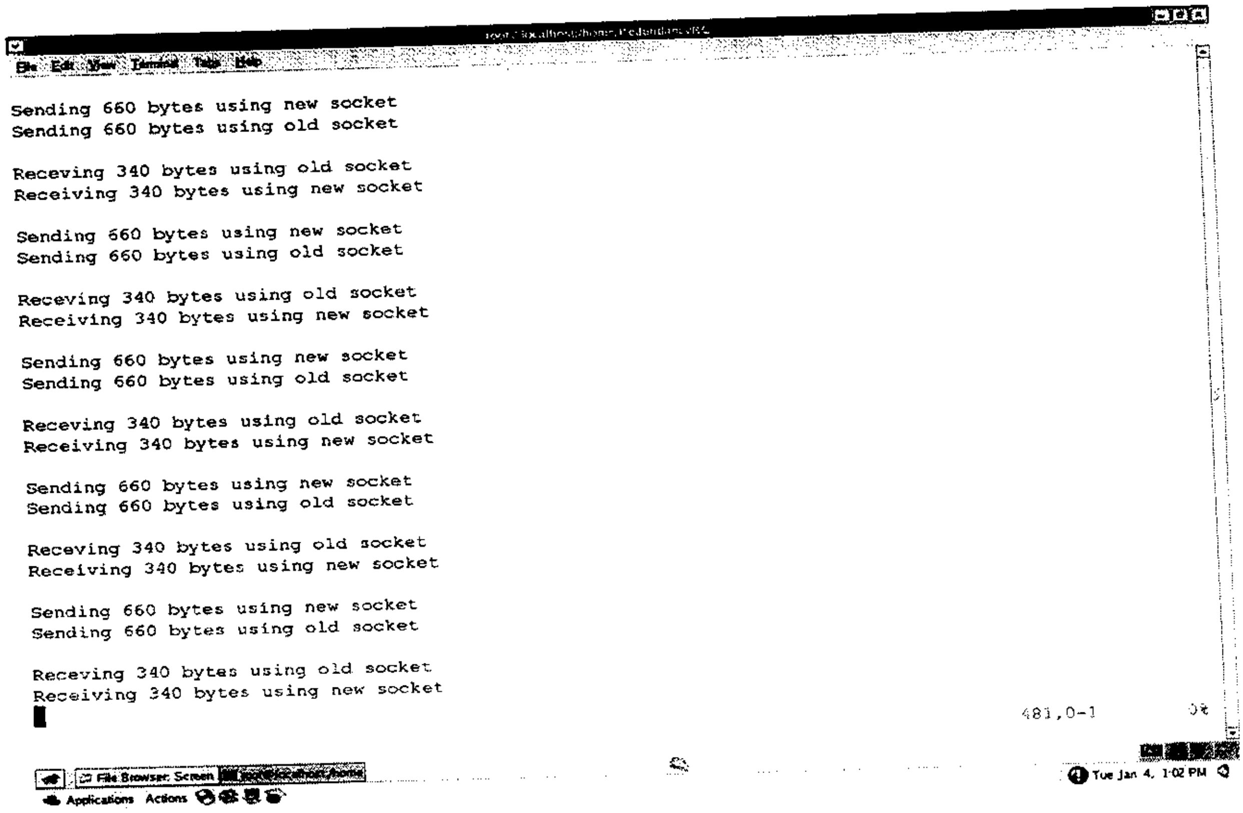


```
Initialising of all Digital and Analog Output cards        [  OK  ]
Initialising of all Digital and Analog Input cards          [  OK  ]
        1        Digital Output cards registered succesfully...

From Card Reg file:              Card[0]:        0

        2        Digital Output cards registered succesfully...

From Card Reg file:              Card[1]:        1

        3        Digital Output cards registered succesfully...

From Card Reg file:              Card[2]:        2

        Digital Input Cards 1    registered successfully...

        Digital Input Cards 2    registered successfully...

        Digital Input Cards 3    registered successfully...

        Digital Input Cards 4    registered successfully...

        Digital Input Cards 5    registered successfully...

        Digital Input Cards 6    registered successfully...


        -----------------------          the control cycle...-----------------      43,0-1          0%
```



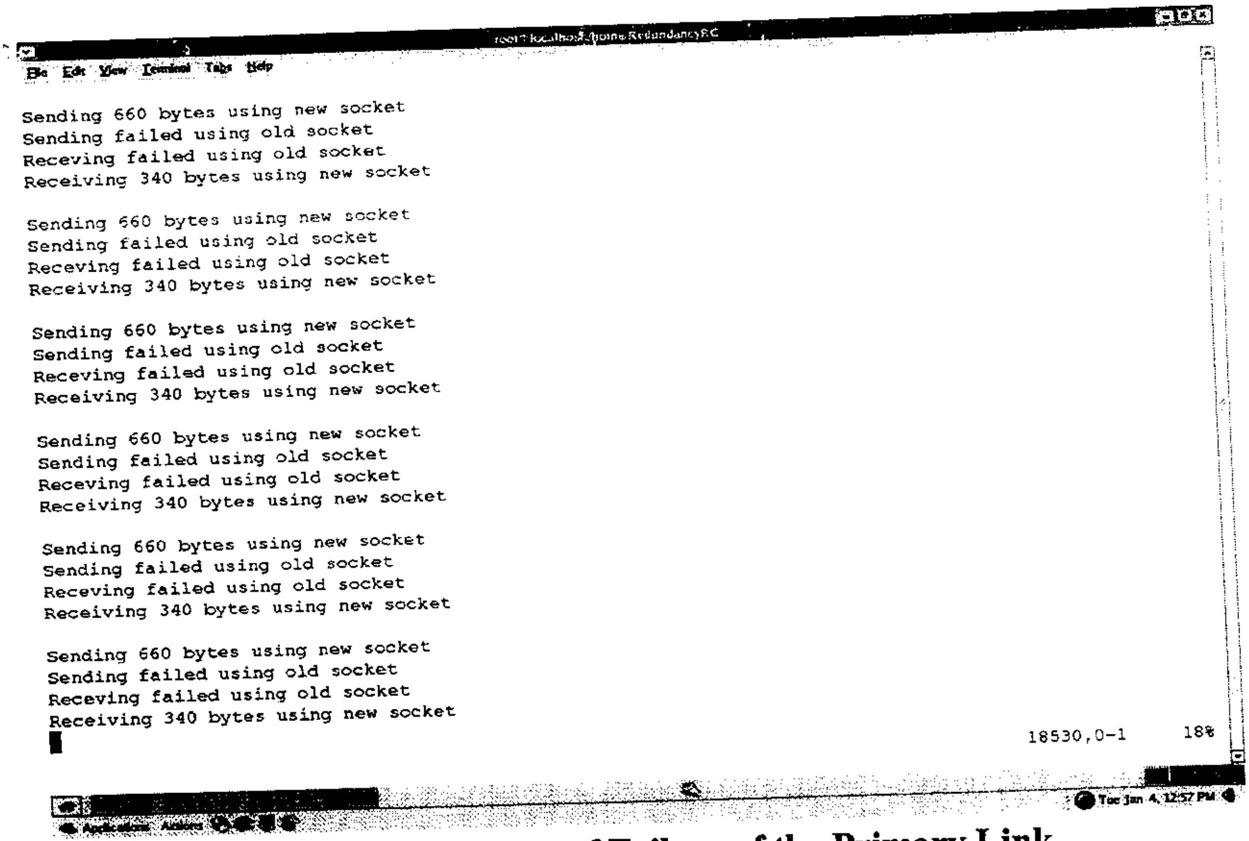
```
===========MAIN CONTROL CYCLE STARTS HERE===========

Sending 340 bytes by old socket
Sending 340 bytes using new socket

Receiving 660 bytes using new socket
Receiving 660 bytes using old socket

Sending 340 bytes by old socket
Sending 340 bytes using new socket


Sending 340 bytes by old socket
Sending 340 bytes using new socket


Sending 340 bytes by old socket
Sending 340 bytes using new socket

Receiving 660 bytes using new socket
Receiving 660 bytes using old socket

Sending 340 bytes by old socket
Sending 340 bytes using new socket

Receiving 660 bytes using new socket
Receiving 660 bytes using old socket

Sending 340 bytes by old socket
Sending 340 bytes using new socket

Receiving 660 bytes using new socket
Receiving 660 bytes using old socket

Sending 340 bytes by old socket
Sending 340 bytes using new socket
                                                              155,0-1        0%
```

Figure 8.5 Main Controller - Sending Command String Using Both Primary And Secondary Sockets



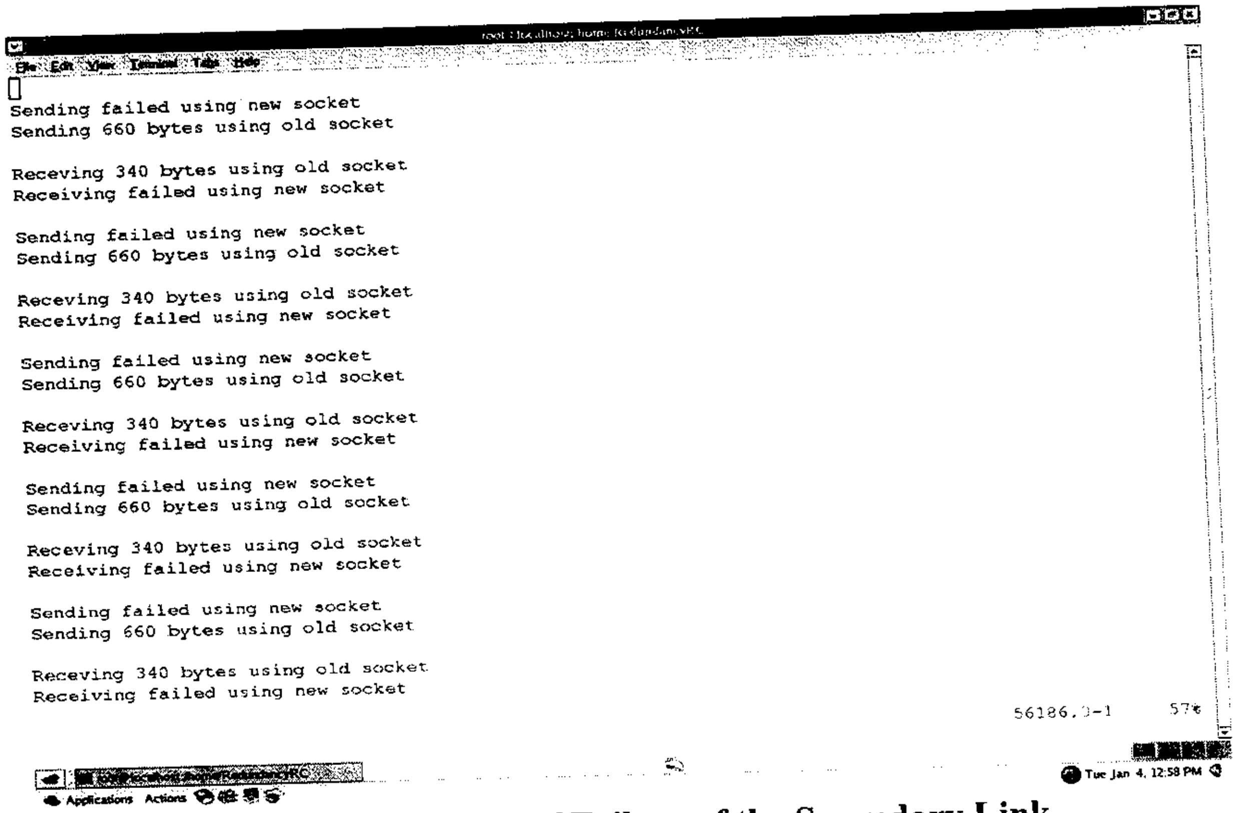
```
Sending 660 bytes using new socket
Sending 660 bytes using old socket

Receving 340 bytes using old socket
Receiving 340 bytes using new socket

Sending 660 bytes using new socket
Sending 660 bytes using old socket

Receving 340 bytes using old socket
Receiving 340 bytes using new socket

Sending 660 bytes using new socket
Sending 660 bytes using old socket

Receving 340 bytes using old socket
Receiving 340 bytes using new socket

Sending 660 bytes using new socket
Sending 660 bytes using old socket

Receving 340 bytes using old socket
Receiving 340 bytes using new socket

Sending 660 bytes using new socket
Sending 660 bytes using old socket

Receving 340 bytes using old socket
Receiving 340 bytes using new socket
                                                              481,0-1        0%
```

Figure 8.6 Remote Controller - Receiving Command String From Both The



```
Sending 660 bytes using new socket
Sending failed using old socket
Receving failed using old socket
Receiving 340 bytes using new socket

Sending 660 bytes using new socket
Sending failed using old socket
Receving failed using old socket
Receiving 340 bytes using new socket

Sending 660 bytes using new socket
Sending failed using old socket
Receving failed using old socket
Receiving 340 bytes using new socket

Sending 660 bytes using new socket
Sending failed using old socket
Receving failed using old socket
Receiving 340 bytes using new socket

Sending 660 bytes using new socket
Sending failed using old socket
Receving failed using old socket
Receiving 340 bytes using new socket

Sending 660 bytes using new socket
Sending failed using old socket
Receving failed using old socket
Receiving 340 bytes using new socket
                                                          18530,0-1     18%
```

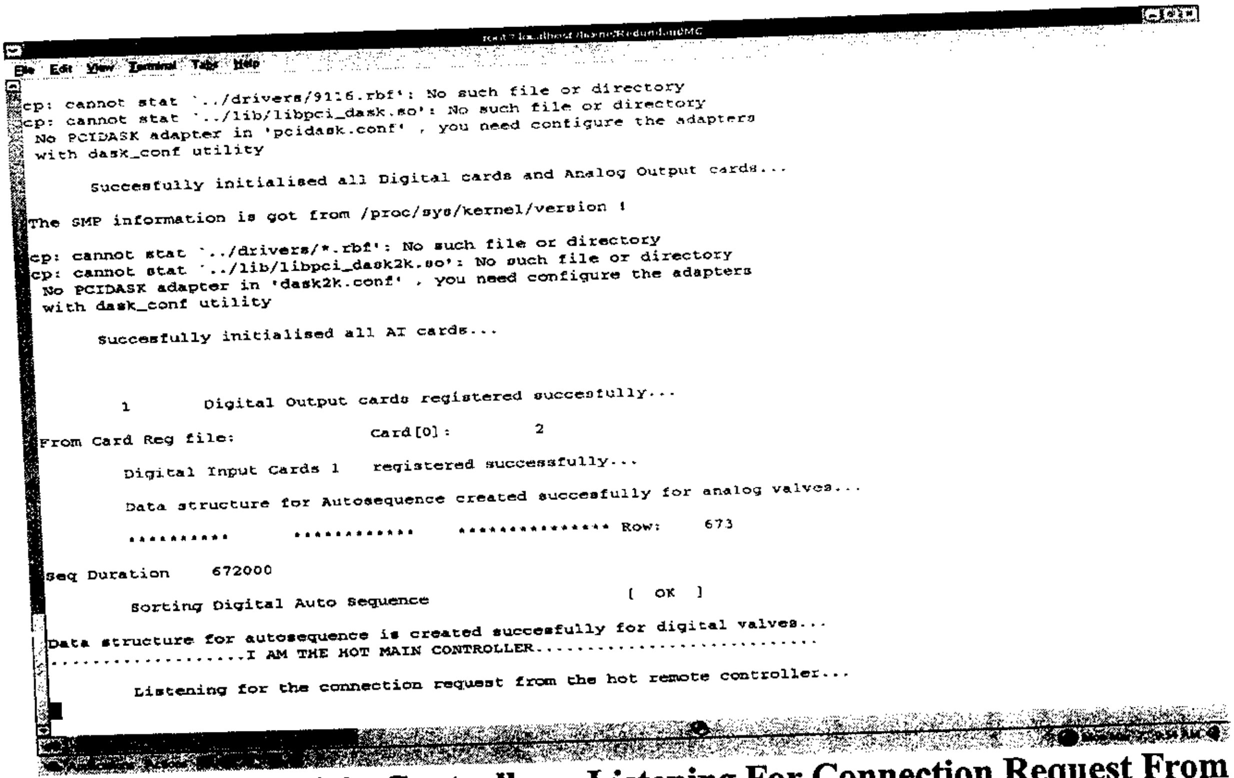
Figure 8.7 In Case of Failure of the Primary Link



```
Sending failed using new socket
Sending 660 bytes using old socket

Receving 340 bytes using old socket
Receiving failed using new socket

Sending failed using new socket
Sending 660 bytes using old socket

Receving 340 bytes using old socket
Receiving failed using new socket

Sending failed using new socket
Sending 660 bytes using old socket

Receving 340 bytes using old socket
Receiving failed using new socket

Sending failed using new socket
Sending 660 bytes using old socket

Receving 340 bytes using old socket
Receiving failed using new socket

Sending failed using new socket
Sending 660 bytes using old socket

Receving 340 bytes using old socket
Receiving failed using new socket
                                                          56186,0-1     57%
```

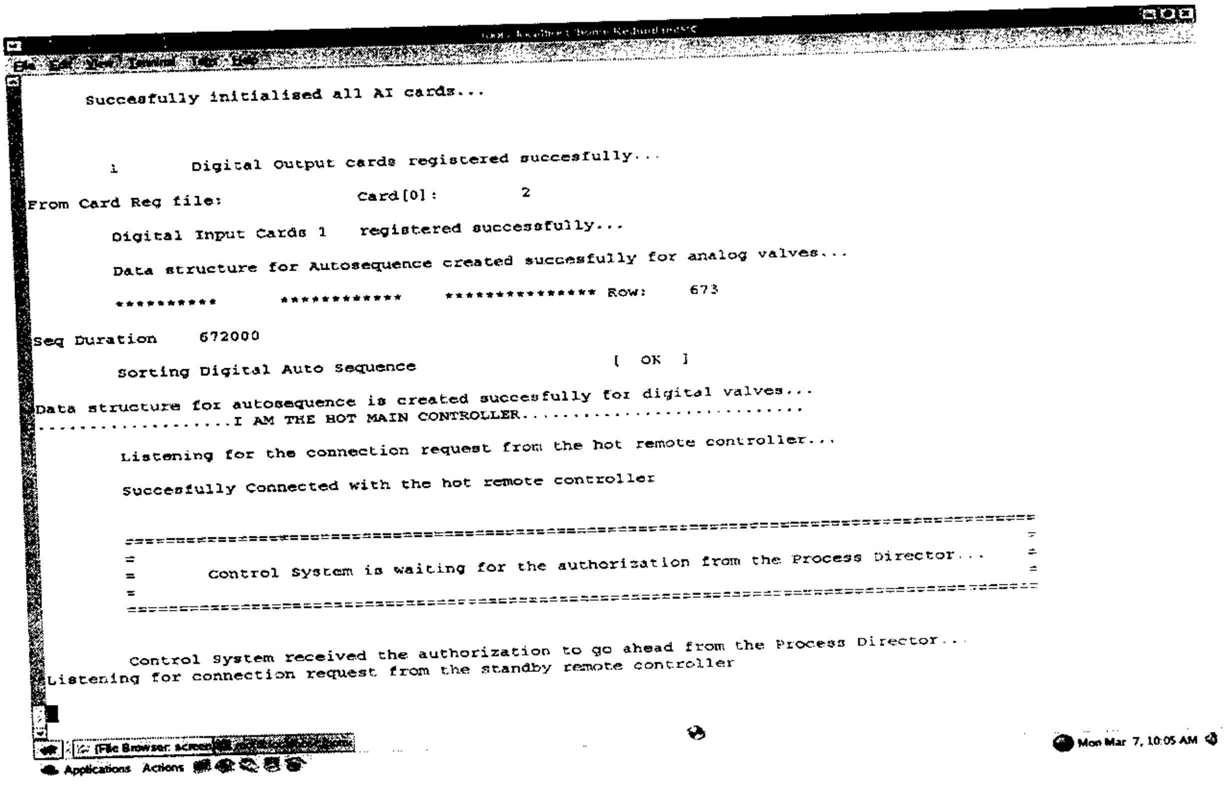
Figure 8.8 In Case of Failure of the Secondary Link

8.2 NETWORK REDUNDANCY WITH REDUNDANT SYSTEM

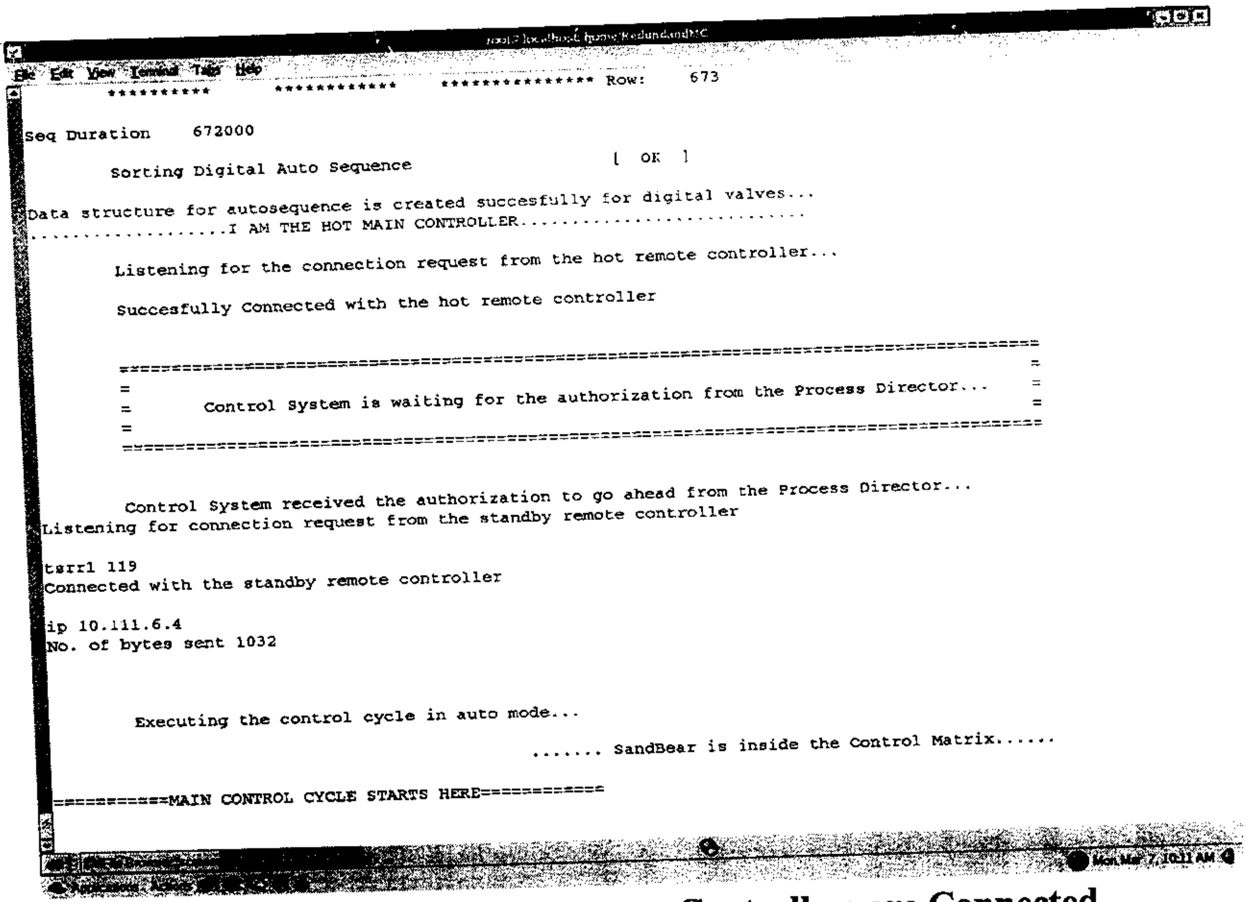


```
cp: cannot stat `../drivers/9116.rbf': No such file or directory
cp: cannot stat `../lib/libpci_dask.so': No such file or directory
 No PCIDASK adapter in 'pcidask.conf' , you need configure the adapters
 with dask_conf utility

       Succesfully initialised all Digital cards and Analog Output cards...

The SMP information is got from /proc/sys/kernel/version !

cp: cannot stat `../drivers/*.rbf': No such file or directory
cp: cannot stat `../lib/libpci_dask2k.so': No such file or directory
 No PCIDASK adapter in 'dask2k.conf' , you need configure the adapters
 with dask_conf utility

       Succesfully initialised all AI cards...


         1        Digital Output cards registered succesfully...

From Card Reg file:                  Card[0]:          2

         Digital Input Cards 1    registered successfully...

         Data structure for Autosequence created succesfully for analog valves...

         **********      ************     *************** Row:     673

Seq Duration     672000

         Sorting Digital Auto Sequence                          [  OK  ]

Data structure for autosequence is created succesfully for digital valves...
...................I AM THE HOT MAIN CONTROLLER...........................

         Listening for the connection request from the hot remote controller...
```

Figure 8.9 Hot Main Controller – Listening For Connection Request From Hot Remote Controller

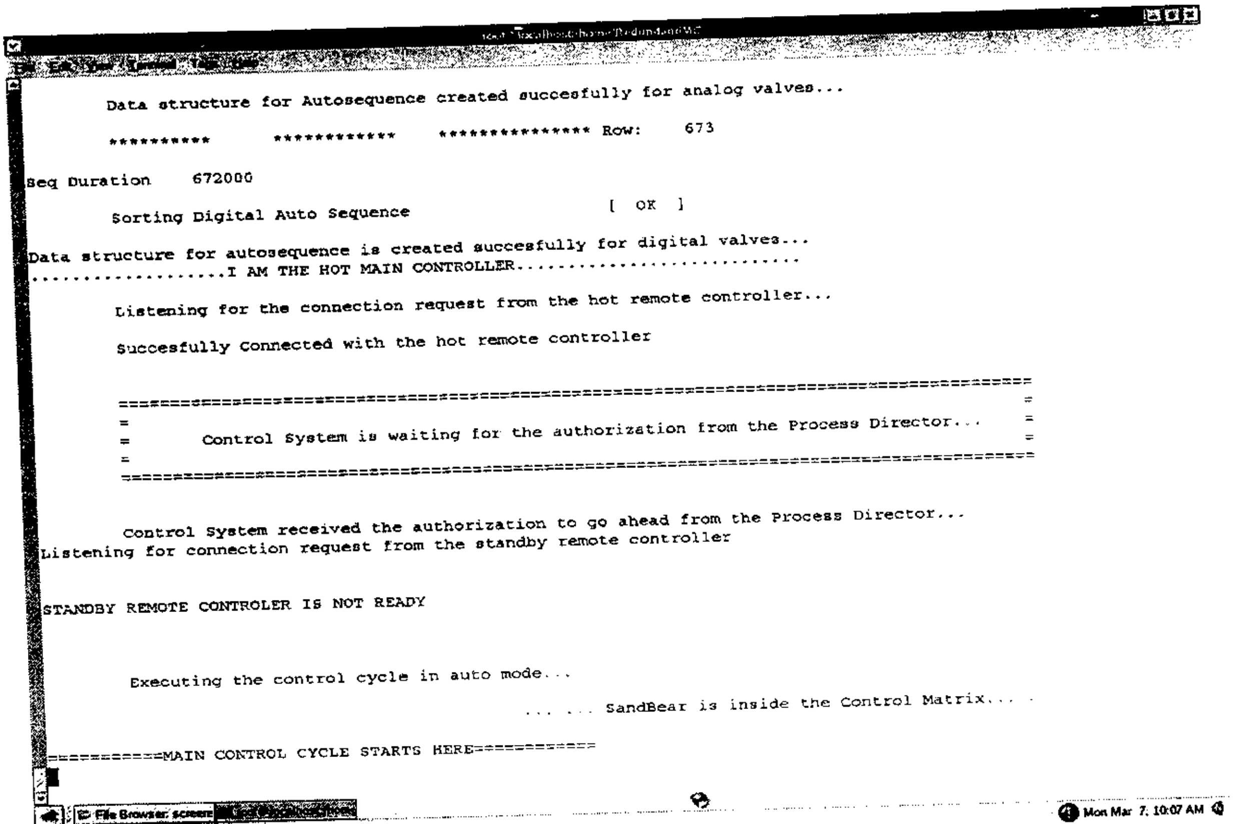


```
       Succesfully initialised all AI cards...


         1        Digital Output cards registered succesfully...

From Card Reg file:                  Card[0]:          2

         Digital Input Cards 1    registered successfully...

         Data structure for Autosequence created succesfully for analog valves...

         **********      ************     *************** Row:     673

Seq Duration     672000

         Sorting Digital Auto Sequence                          [  OK  ]

Data structure for autosequence is created succesfully for digital valves...
...................I AM THE HOT MAIN CONTROLLER...........................

         Listening for the connection request from the hot remote controller...

         Succesfully Connected with the hot remote controller

         ===========================================================================
         =                                                                         =
         =        Control System is waiting for the authorization from the Process Director...   =
         =                                                                         =
         ===========================================================================

         Control System received the authorization to go ahead from the Process Director...
Listening for connection request from the standby remote controller
```

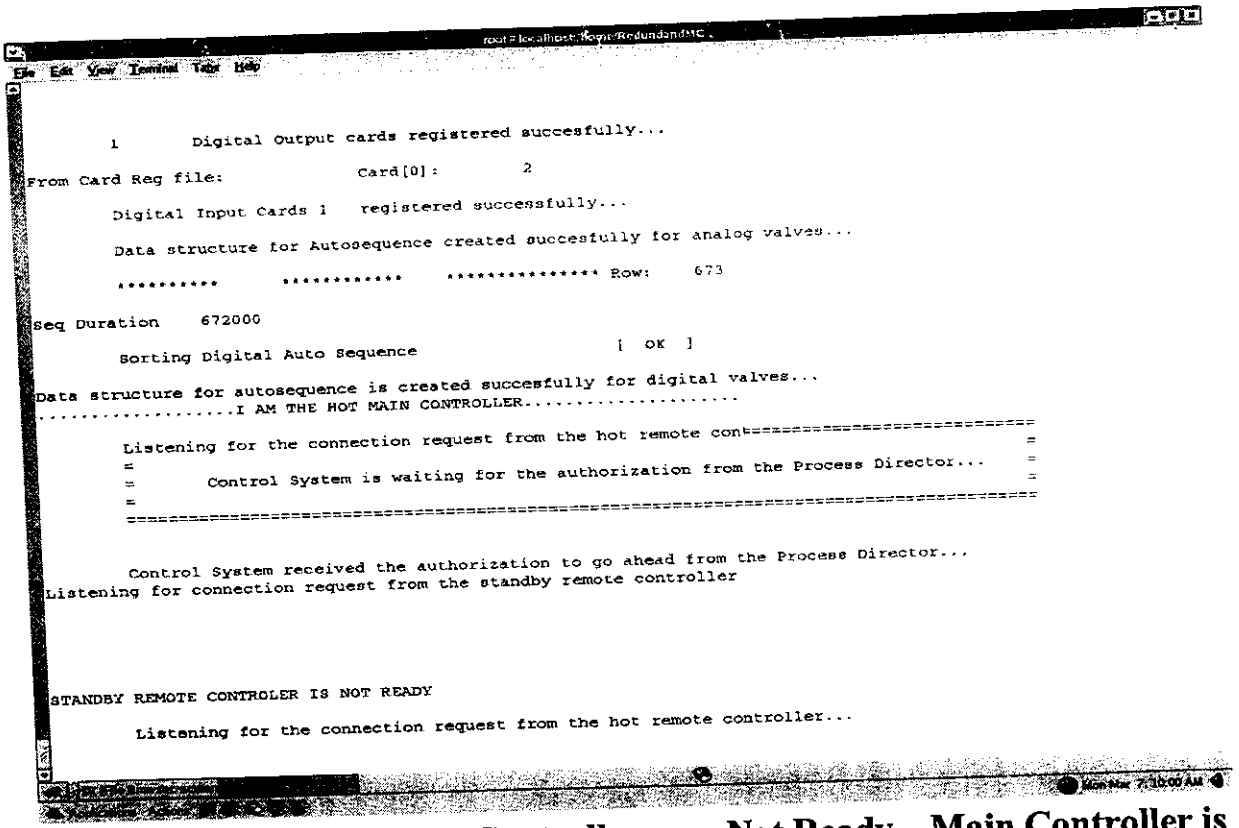


```
        **********          ************      *************** Row:     673

Seq Duration     672000

        Sorting Digital Auto Sequence                          [  OK  ]

Data structure for autosequence is created succesfully for digital valves...
..................I AM THE HOT MAIN CONTROLLER..........................

        Listening for the connection request from the hot remote controller...

        Succesfully Connected with the hot remote controller

        ====================================================================================
        =                                                                                  =
        =       Control System is waiting for the authorization from the Process Director...  =
        =                                                                                  =
        ====================================================================================

        Control System received the authorization to go ahead from the Process Director...
Listening for connection request from the standby remote controller

tsrrl 119
Connected with the standby remote controller

ip 10.111.6.4
No. of bytes sent 1032


        Executing the control cycle in auto mode...
                                                  ....... SandBear is inside the Control Matrix......


==========MAIN CONTROL CYCLE STARTS HERE============
```

Figure 8.11 Both The Remote Controllers are Connected

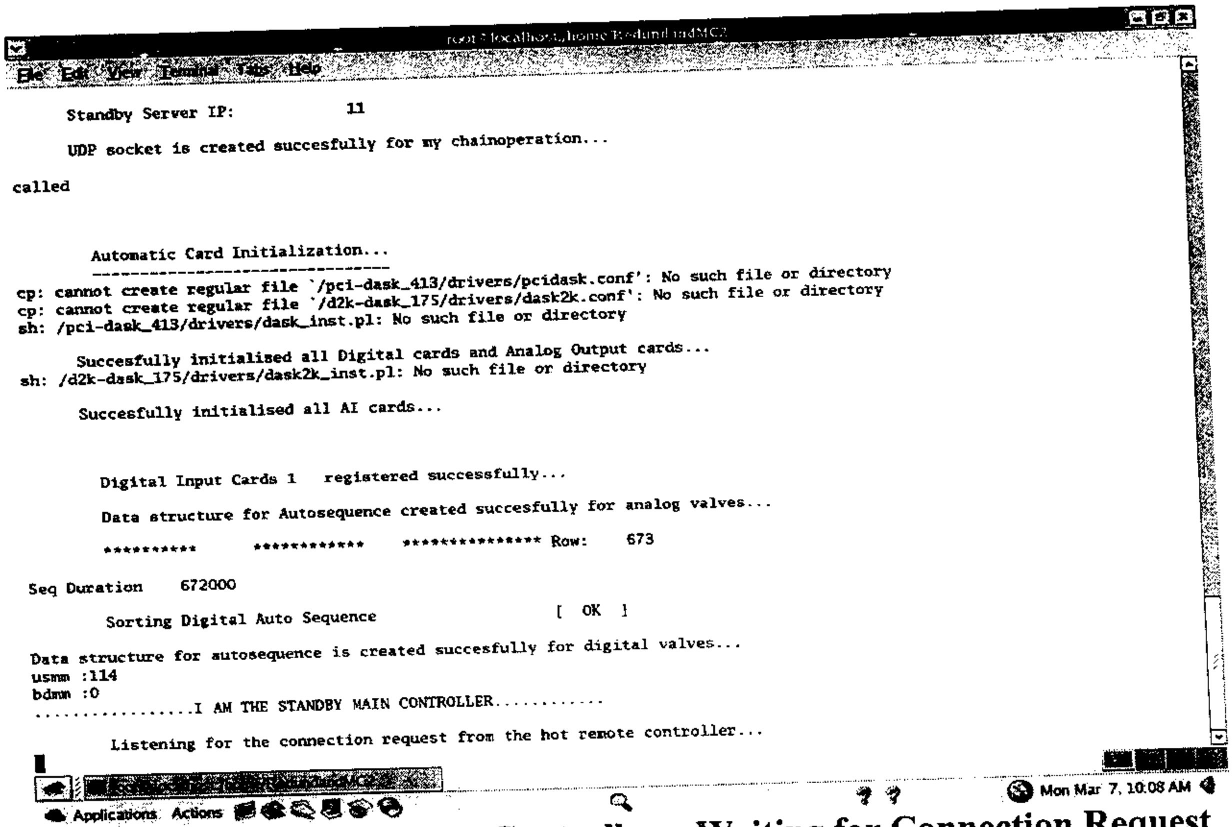


```
        Data structure for Autosequence created succesfully for analog valves...

        **********          ************      *************** Row:     673

Seq Duration     672000

        Sorting Digital Auto Sequence                          [  OK  ]

Data structure for autosequence is created succesfully for digital valves...
..................I AM THE HOT MAIN CONTROLLER..........................

        Listening for the connection request from the hot remote controller...

        Succesfully Connected with the hot remote controller

        ====================================================================================
        =                                                                                  =
        =       Control System is waiting for the authorization from the Process Director...  =
        =                                                                                  =
        ====================================================================================

        Control System received the authorization to go ahead from the Process Director...
Listening for connection request from the standby remote controller


STANDBY REMOTE CONTROLER IS NOT READY


        Executing the control cycle in auto mode...
                                                  ... ... SandBear is inside the Control Matrix...

==========MAIN CONTROL CYCLE STARTS HERE============
```

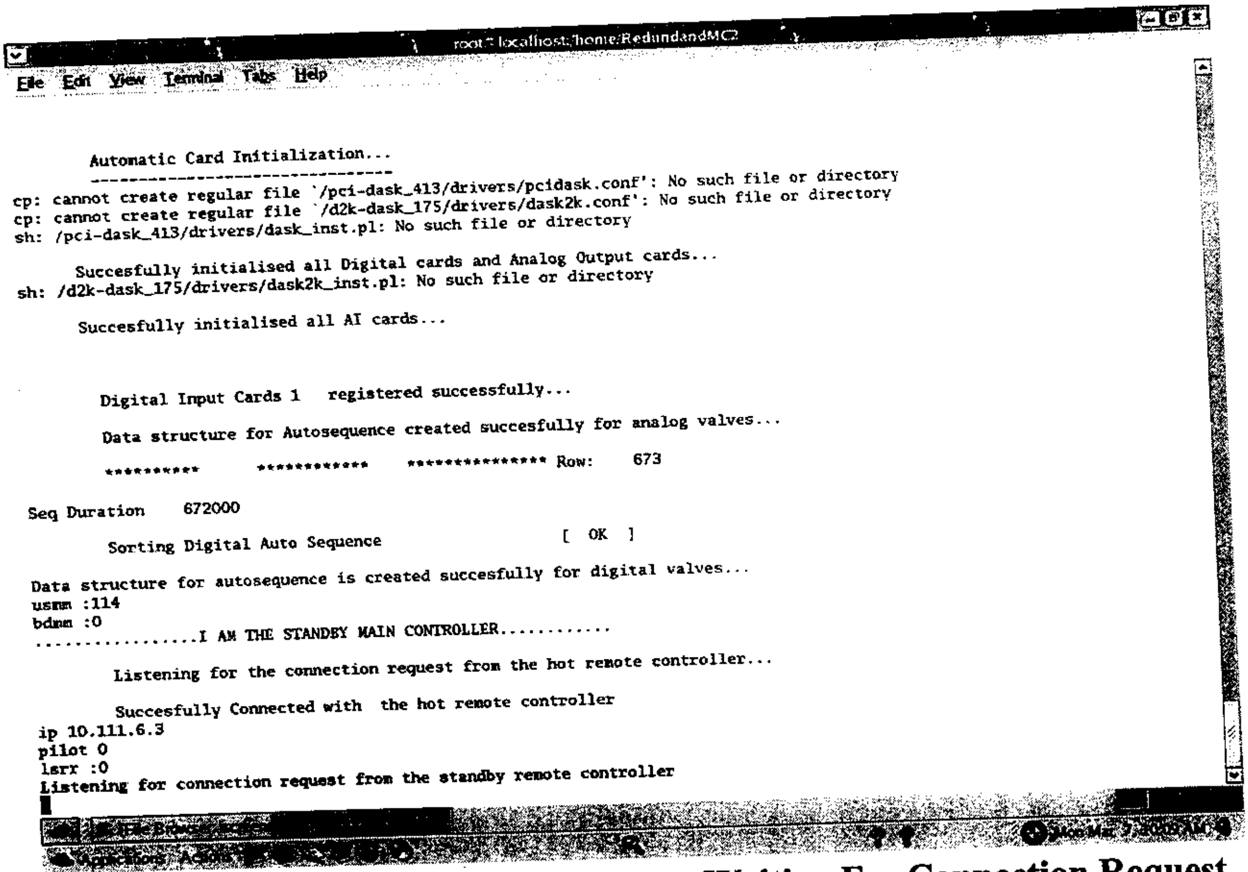


```
        1       Digital Output cards registered succesfully...

From Card Reg file:                  Card[0]:          2

        Digital Input Cards 1   registered successfully...

        Data structure for Autosequence created succesfully for analog valves...

        **********      ************    *************** Row:     673

Seq Duration    672000

        Sorting Digital Auto Sequence                          [  OK  ]

Data structure for autosequence is created succesfully for digital valves...
..................I AM THE HOT MAIN CONTROLLER....................

        Listening for the connection request from the hot remote con=========================
        =                                                                                   =
        =       Control System is waiting for the authorization from the Process Director... =
        =                                                                                   =
        ====================================================================================


        Control System received the authorization to go ahead from the Process Director...
Listening for connection request from the standby remote controller




STANDBY REMOTE CONTROLER IS NOT READY

        Listening for the connection request from the hot remote controller...
```

Figure 8.13 Both Remote Controllers are Not Ready – Main Controller is Waiting in the Loop

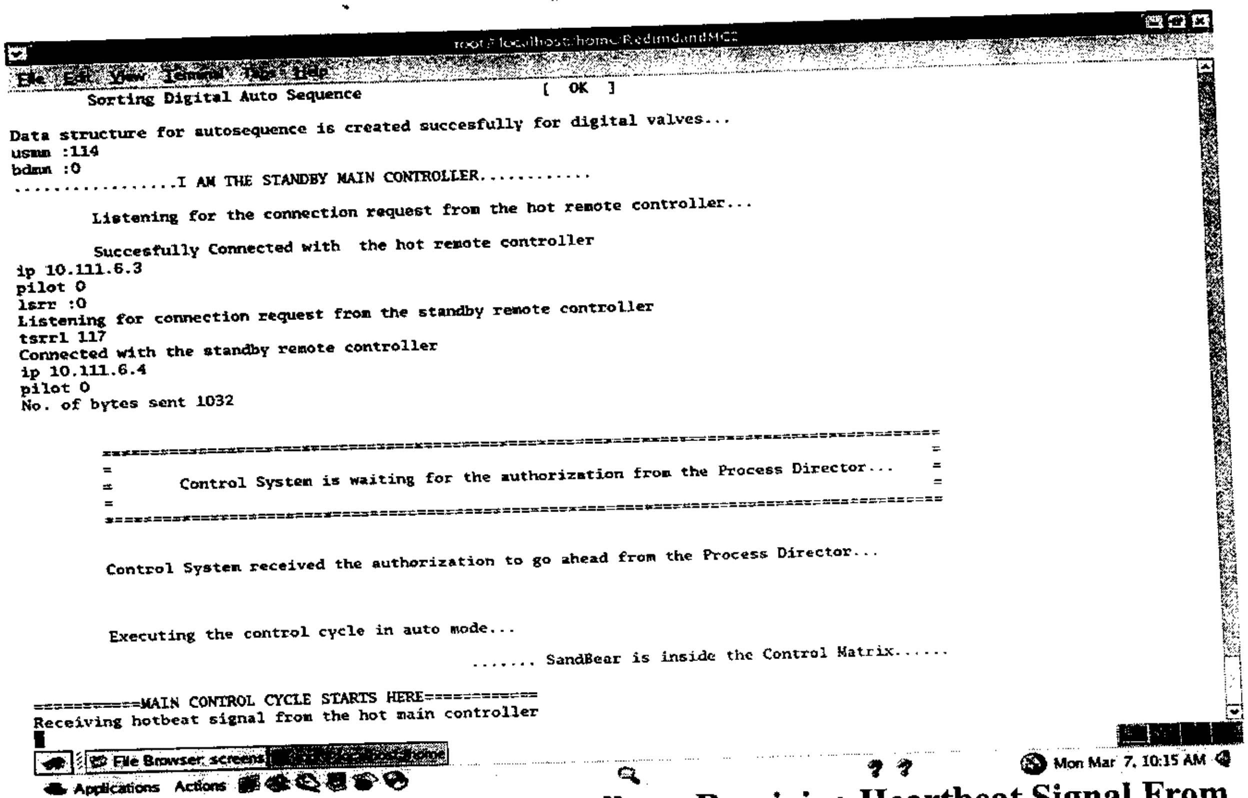


```
     Standby Server IP:           11

     UDP socket is created succesfully for my chainoperation...

called



        Automatic Card Initialization...
        ---------------------------------
cp: cannot create regular file `/pci-dask_413/drivers/pcidask.conf': No such file or directory
cp: cannot create regular file `/d2k-dask_175/drivers/dask2k.conf': No such file or directory
sh: /pci-dask_413/drivers/dask_inst.pl: No such file or directory

        Succesfully initialised all Digital cards and Analog Output cards...
sh: /d2k-dask_175/drivers/dask2k_inst.pl: No such file or directory

        Succesfully initialised all AI cards...



        Digital Input Cards 1   registered successfully...

        Data structure for Autosequence created succesfully for analog valves...

        **********      ************    *************** Row:     673

Seq Duration    672000

        Sorting Digital Auto Sequence                   [  OK  ]

Data structure for autosequence is created succesfully for digital valves...
usmm :114
bdmm :0
................I AM THE STANDBY MAIN CONTROLLER...........

        Listening for the connection request from the hot remote controller...
```


root@localhost:/home/RedundandMC2
File Edit View Terminal Tabs Help

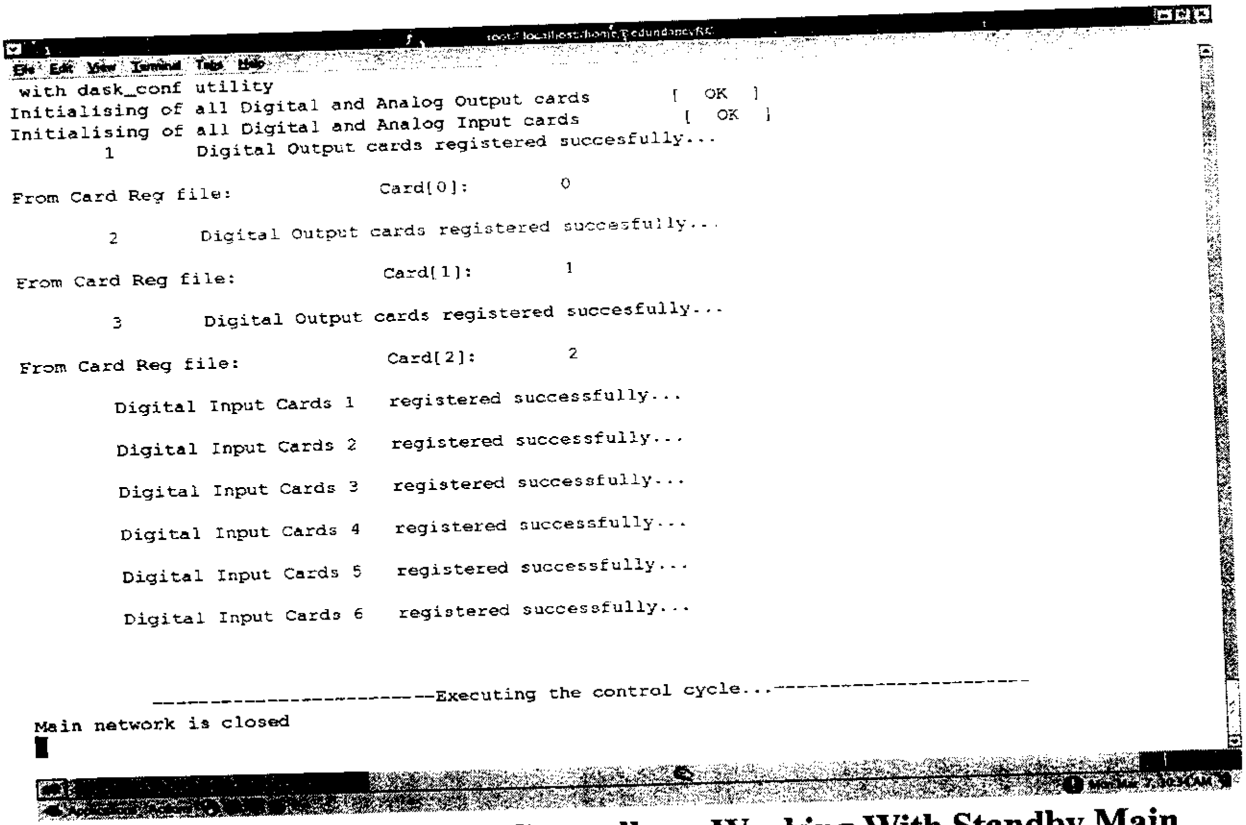
```
      Automatic Card Initialization...
      ---------------------------------
cp: cannot create regular file `/pci-dask_413/drivers/pcidask.conf': No such file or directory
cp: cannot create regular file `/d2k-dask_175/drivers/dask2k.conf': No such file or directory
sh: /pci-dask_413/drivers/dask_inst.pl: No such file or directory

     Succesfully initialised all Digital cards and Analog Output cards...
sh: /d2k-dask_175/drivers/dask2k_inst.pl: No such file or directory

     Succesfully initialised all AI cards...


       Digital Input Cards 1   registered successfully...

       Data structure for Autosequence created succesfully for analog valves...

       **********      ************     *************** Row:     673

 Seq Duration     672000

        Sorting Digital Auto Sequence                    [  OK  ]

Data structure for autosequence is created succesfully for digital valves...
usmm :114
bdmm :0
................I AM THE STANDBY MAIN CONTROLLER...........

        Listening for the connection request from the hot remote controller...

        Succesfully Connected with  the hot remote controller
ip 10.111.6.3
pilot 0
lsrr :0
Listening for connection request from the standby remote controller
█
```

Figure 8.15 Standby Main Controller – Waiting For Connection Request From The Standby Remote Controller


root@localhost:/home/RedundandMC2
File Edit View Terminal Tabs Help
File Browser: screens
Mon Mar 7, 10:15 AM
Applications Actions

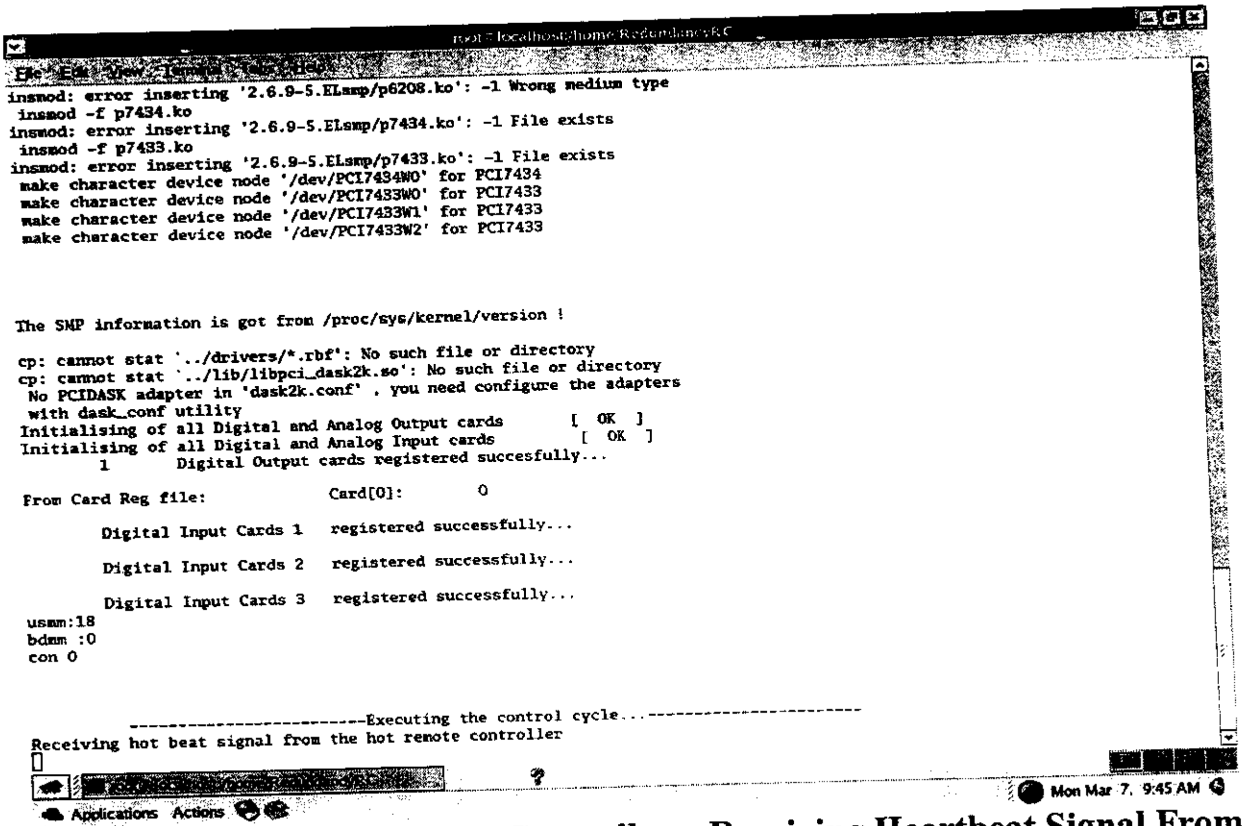
```
        Sorting Digital Auto Sequence                    [  OK  ]

Data structure for autosequence is created succesfully for digital valves...
usmm :114
bdmm :0
................I AM THE STANDBY MAIN CONTROLLER...........

        Listening for the connection request from the hot remote controller...

        Succesfully Connected with  the hot remote controller
ip 10.111.6.3
pilot 0
lsrr :0
Listening for connection request from the standby remote controller
tsrrl 117
Connected with the standby remote controller
ip 10.111.6.4
pilot 0
No. of bytes sent 1032


      ==================================================================
      =                                                                =
      =     Control System is waiting for the authorization from the Process Director...   =
      =                                                                =
      ==================================================================


      Control System received the authorization to go ahead from the Process Director...



      Executing the control cycle in auto mode...
                                            ....... SandBear is inside the Control Matrix......

   ==========MAIN CONTROL CYCLE STARTS HERE===========
   Receiving hotbeat signal from the hot main controller
   █
```

Receiving Heartbeat Signal From


root@localhost:/home/RedundancyRC
File Edit View Terminal Tabs Help

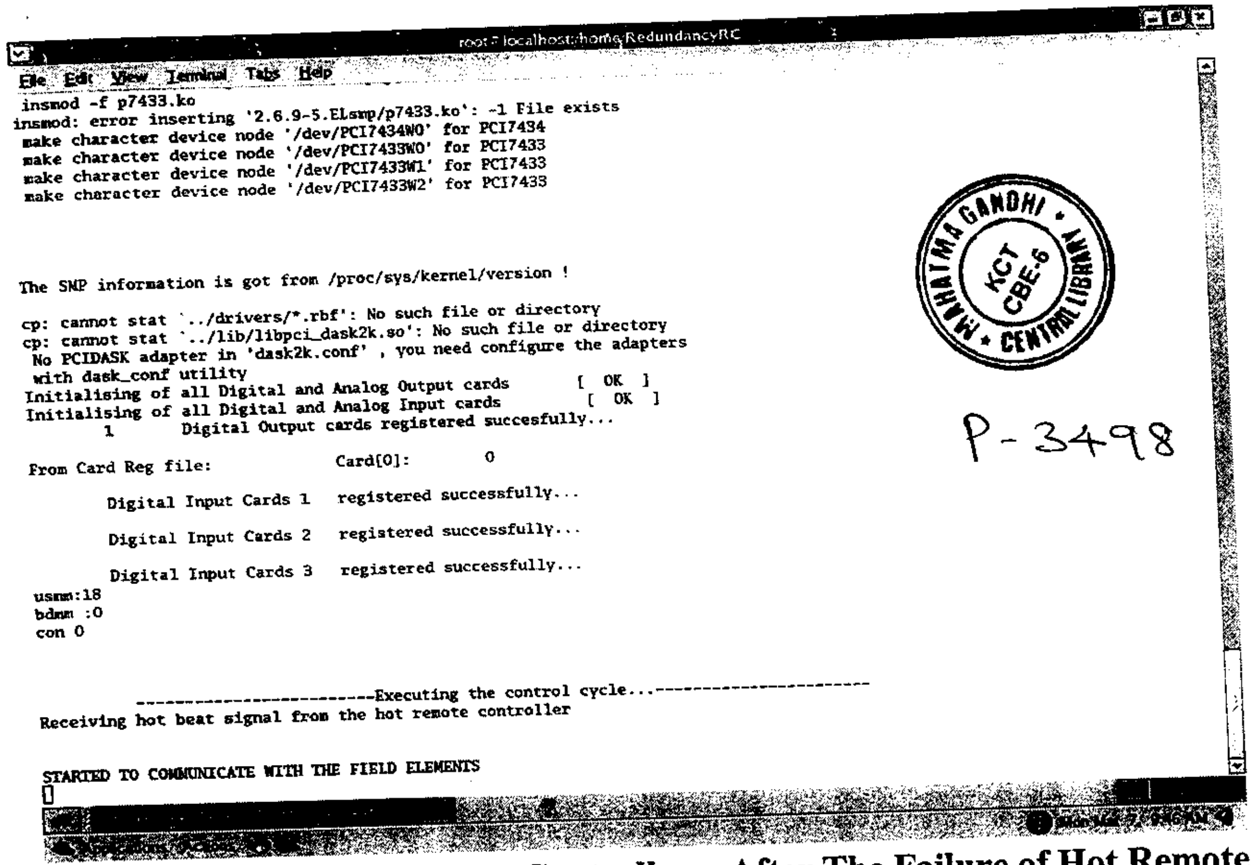
```
with dask_conf utility
Initialising of all Digital and Analog Output cards          [  OK  ]
Initialising of all Digital and Analog Input cards            [  OK  ]
       1        Digital Output cards registered succesfully...

From Card Reg file:                 Card[0]:          0

       2        Digital Output cards registered succesfully...

From Card Reg file:                 Card[1]:          1

       3        Digital Output cards registered succesfully...

From Card Reg file:                 Card[2]:          2

        Digital Input Cards 1   registered successfully...

        Digital Input Cards 2   registered successfully...

        Digital Input Cards 3   registered successfully...

        Digital Input Cards 4   registered successfully...

        Digital Input Cards 5   registered successfully...

        Digital Input Cards 6   registered successfully...


             -----------------------Executing the control cycle...---------------------
 Main network is closed
```

Figure 8.17 Hot Remote Controller – Working With Standby Main Controller


root@localhost:/home/RedundancyRC
File Edit View Terminal Tabs Help
Applications Actions
Mon Mar 7, 9:45 AM

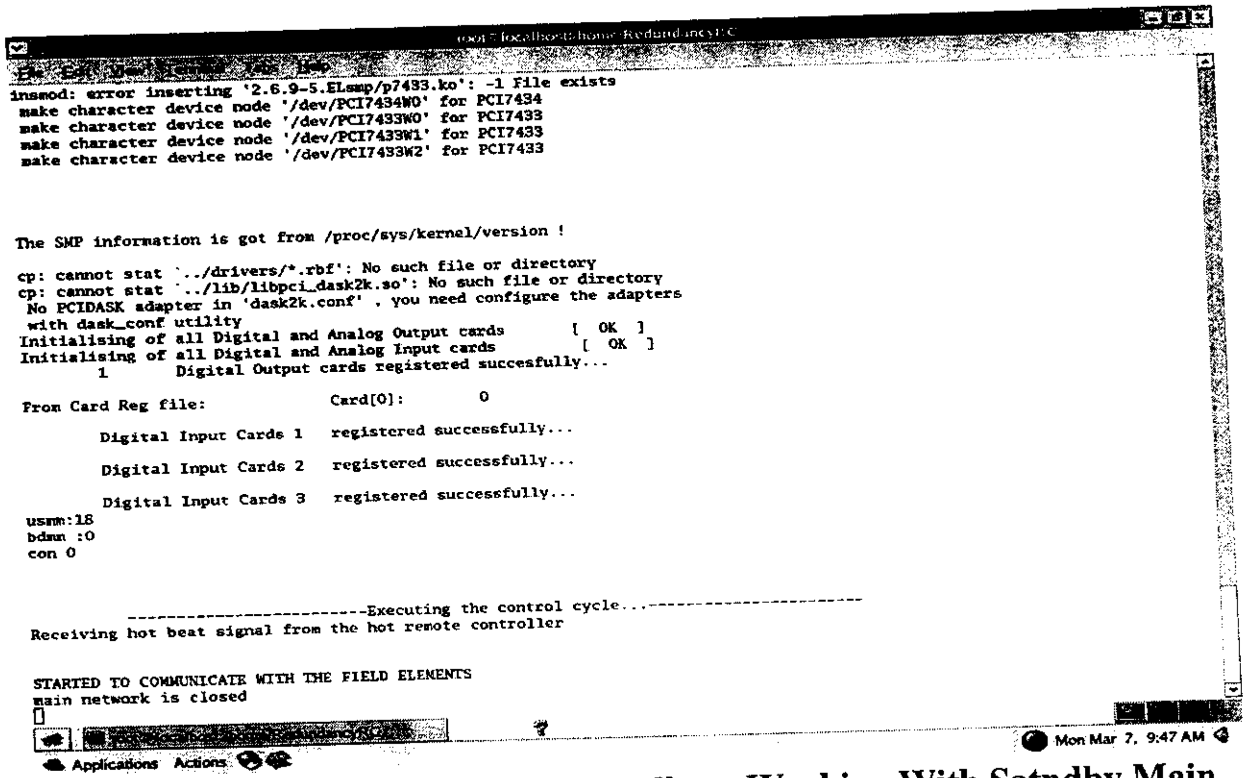
```
insmod: error inserting '2.6.9-5.ELsmp/p6208.ko': -1 Wrong medium type
 insmod -f p7434.ko
insmod: error inserting '2.6.9-5.ELsmp/p7434.ko': -1 File exists
 insmod -f p7433.ko
insmod: error inserting '2.6.9-5.ELsmp/p7433.ko': -1 File exists
 make character device node '/dev/PCI7434W0' for PCI7434
 make character device node '/dev/PCI7433W0' for PCI7433
 make character device node '/dev/PCI7433W1' for PCI7433
 make character device node '/dev/PCI7433W2' for PCI7433



The SMP information is got from /proc/sys/kernel/version !

cp: cannot stat `../drivers/*.rbf': No such file or directory
cp: cannot stat `../lib/libpci_dask2k.so': No such file or directory
 No PCIDASK adapter in 'dask2k.conf' , you need configure the adapters
 with dask_conf utility
Initialising of all Digital and Analog Output cards         [  OK  ]
Initialising of all Digital and Analog Input cards           [  OK  ]
      1       Digital Output cards registered succesfully...

From Card Reg file:              Card[0]:        0

        Digital Input Cards 1   registered successfully...

        Digital Input Cards 2   registered successfully...

        Digital Input Cards 3   registered successfully...
usmm:18
bdmm :0
con 0


          -----------------------Executing the control cycle...----------------------
Receiving hot beat signal from the hot remote controller
```

Figure 8.18 Standby Remote Controller – Receiving Heartbeat Signal From



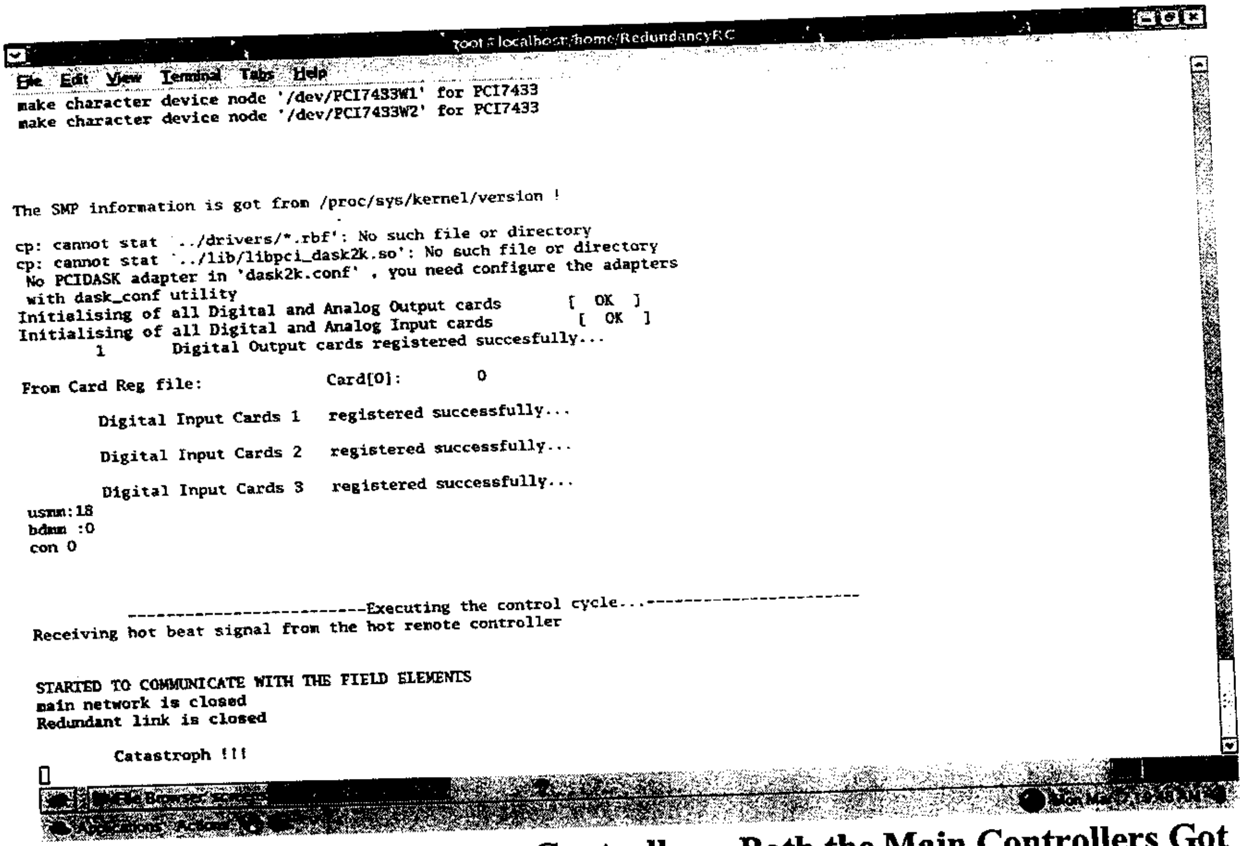
```
insmod -f p7433.ko
insmod: error inserting '2.6.9-5.ELsmp/p7433.ko': -1 File exists
 make character device node '/dev/PCI7434W0' for PCI7434
 make character device node '/dev/PCI7433W0' for PCI7433
 make character device node '/dev/PCI7433W1' for PCI7433
 make character device node '/dev/PCI7433W2' for PCI7433



The SMP information is got from /proc/sys/kernel/version !

cp: cannot stat `../drivers/*.rbf': No such file or directory
cp: cannot stat `../lib/libpci_dask2k.so': No such file or directory
 No PCIDASK adapter in 'dask2k.conf' , you need configure the adapters
 with dask_conf utility
Initialising of all Digital and Analog Output cards        [  OK  ]
Initialising of all Digital and Analog Input cards          [  OK  ]
        1       Digital Output cards registered succesfully...

From Card Reg file:              Card[0]:        0

        Digital Input Cards 1   registered successfully...

        Digital Input Cards 2   registered successfully...

        Digital Input Cards 3   registered successfully...
usmm:18
bdmm :0
con 0


           ------------------------Executing the control cycle...---------------------
Receiving hot beat signal from the hot remote controller


STARTED TO COMMUNICATE WITH THE FIELD ELEMENTS
```

Figure 8.19 Standby Remote Controller – After The Failure of Hot Remote Controller



```
insmod: error inserting '2.6.9-5.ELsmp/p7433.ko': -1 File exists
 make character device node '/dev/PCI7434W0' for PCI7434
 make character device node '/dev/PCI7433W0' for PCI7433
 make character device node '/dev/PCI7433W1' for PCI7433
 make character device node '/dev/PCI7433W2' for PCI7433



The SMP information is got from /proc/sys/kernel/version !

cp: cannot stat `../drivers/*.rbf': No such file or directory
cp: cannot stat `../lib/libpci_dask2k.so': No such file or directory
 No PCIDASK adapter in 'dask2k.conf' , you need configure the adapters
 with dask_conf utility
Initialising of all Digital and Analog Output cards        [  OK  ]
Initialising of all Digital and Analog Input cards          [  OK  ]
        1       Digital Output cards registered succesfully...

From Card Reg file:              Card[0]:        0

        Digital Input Cards 1   registered successfully...

        Digital Input Cards 2   registered successfully...

        Digital Input Cards 3   registered successfully...
usmm:18
bdmm :0
con 0


           ------------------------Executing the control cycle...---------------------
Receiving hot beat signal from the hot remote controller

STARTED TO COMMUNICATE WITH THE FIELD ELEMENTS
main network is closed
```

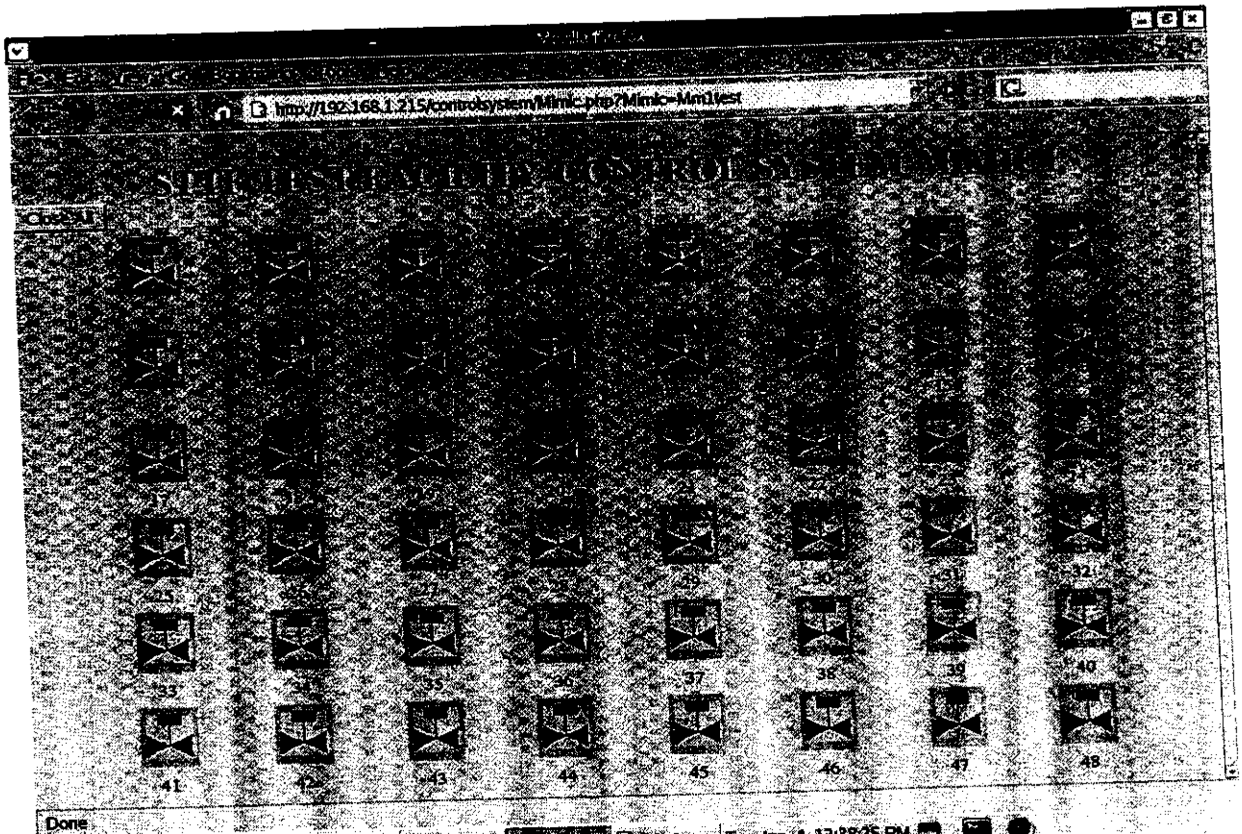
Figure 8.20 Standby Remote Controller – Working With Satndby Main



root@localhost:/home/RedundancyRC
File Edit View Terminal Tabs Help

```
make character device node '/dev/PCI7433W1' for PCI7433
make character device node '/dev/PCI7433W2' for PCI7433



The SMP information is got from /proc/sys/kernel/version !

cp: cannot stat `../drivers/*.rbf': No such file or directory
cp: cannot stat `../lib/libpci_dask2k.so': No such file or directory
 No PCIDASK adapter in 'dask2k.conf' , you need configure the adapters
 with dask_conf utility
Initialising of all Digital and Analog Output cards        [  OK  ]
Initialising of all Digital and Analog Input cards          [  OK  ]
        1        Digital Output cards registered succesfully...

From Card Reg file:              Card[0]:        0

        Digital Input Cards 1   registered successfully...

        Digital Input Cards 2   registered successfully...

        Digital Input Cards 3   registered successfully...
usmn:18
bdmn :0
con 0


          -------------------------Executing the control cycle...----------------------
Receiving hot beat signal from the hot remote controller


STARTED TO COMMUNICATE WITH THE FIELD ELEMENTS
main network is closed
Redundant link is closed

        Catastroph !!!
```

Figure 8.21 Standby Remote Controller – Both the Main Controllers Got Failed



Mozilla Firefox
http://192.168.1.215/controlsystem/Mimic.php?Mimic=Mimic1test
Done
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48

CHAPTER 9
CONCLUSION

Thus by developing network redundancy, the communication link in the PC based control system is strengthened in such a way that if one of the link fails, the other link will automatically carry over the communication and so the failure of the testing can be avoided. Also the development of the network redundancy with redundant system will strengthen the PC based control system as a whole. Since two main controllers and two remote controllers are used here, even if one of the main or remote controller gets failed, the other main or remote controller will take care of the operation and so the possibility of failure of the testing cab be reduced to a greater amount.

REFERENCES

[1] C.Scott Thode, "Distributed Substation Control System with PC Based Local Control" IEE 2nd International Conference on Advances in Power System Control, Operation and Management, December 1993, Hong Kong.

[2] P. Jines, E.Anzalone, B,Craft, A.Crappell, M.Fedurin, T. Miller, M.Smith, Y.Wang, and T.Zhao, "PLC and LINUX based control system for the CAMD LINAC" Proceedings of the Particle Accelerator Conference,2003,USA.

[3] Micaela casera Magro and Paolo Pinceti, "Measuring Real Time Performance of PC Based Industrial Control" University of Genova, Italy.

[4] Edward P. Miska and James R. Mahar, "PC Based Control Systems for Large Hydro Projects" IEEE Winter Power,1999.

[5] Kristina Ahlstrom and Jan Torin, "Redundancy Management in Distributed Flight Control Systems:Experince and Simulations" chalmers University of Technology,Sweden.

[6] Oleg Sokolsky, Mohamed Younis, Insup Leez, Hee-Hwan kwakz and jeff Zhouy, "Verification of the Redundancy Management System for Space launch Vehicle" University of Pennsylvania,Philadelphia.

[7] W.Richard Stevens, "Unix Network Programming" Volume 1-2, Prentice Hall.

[8] Doughlas E. Corner and david L, stevens, "Internetworking with TCP/IP" volumes 1 and 3, Prentice hall.

[9] Richard Stevens and Gray R. Wright, " TCP/IP Illustrated " Volume 1-3, Addison

[10] Dr.Sushil Dass Gupta, “Control System Theory” Edition 4.

[11] Beej's Guide to Network Programming,www.ecst.csuchico.edu/~beej/guide/net/.

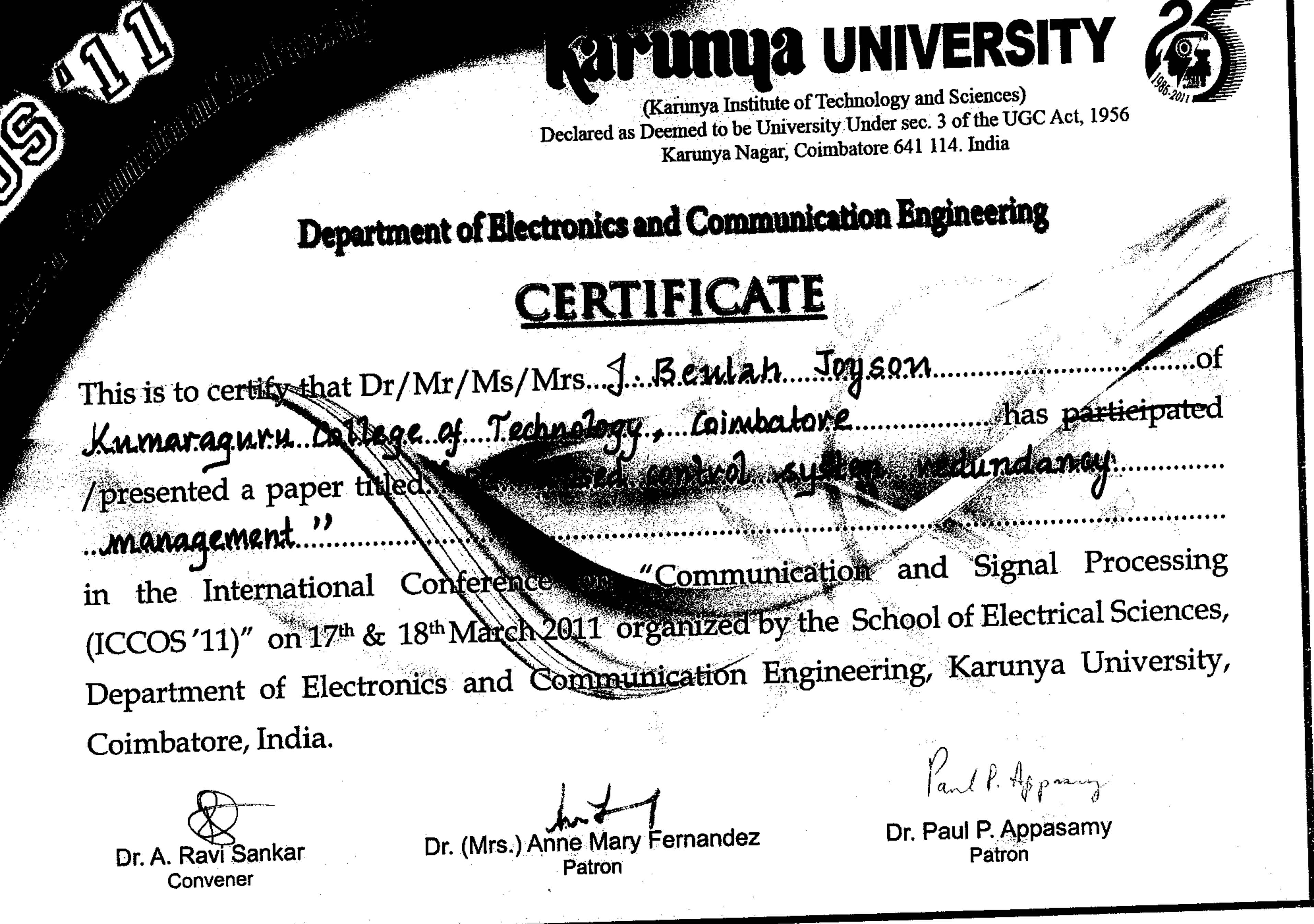
[12] http://ee.sharif.edu/~industrialcontrol/LADDER_LOGIC_Tutorial.pdf

[13] http://uuu.enseirb.fr/~kadionik/rmll2007/presentation/edouard_tisserant.pdf

[14] http://www.engineeringtalk.com/news/hay/hay112.html

[15] http://www.eod.gvsu.edu/~jackh/books/plcs/chapters/plc_function.pdf

[16] http://home.iitk.ac.in/~chebrolu/ee673-f06/sockets.pdf



Karunya UNIVERSITY

(Karunya Institute of Technology and Sciences)
Declared as Deemed to be University Under sec. 3 of the UGC Act, 1956
Karunya Nagar, Coimbatore 641 114. India

Department of Electronics and Communication Engineering

CERTIFICATE

This is to certify that Dr/Mr/Ms/Mrs J. Beulah Joyson of Kumaraguru College of Technology, Coimbatore has ~~participated~~ /presented a paper titled "[illegible] based control system redundancy management" in the International Conference on "Communication and Signal Processing (ICCOS'11)" on 17th & 18th March 2011 organized by the School of Electrical Sciences, Department of Electronics and Communication Engineering, Karunya University, Coimbatore, India.

[illegible]	[illegible]	Paul P. Appasamy
Dr. A. Ravi Sankar Convener	Dr. (Mrs.) Anne Mary Fernandez Patron	Dr. Paul P. Appasamy Patron

SUN COLLEGE OF ENGINEERING AND TECHNOLOGY

20[illegible]

3rd

INTERNATIONAL CONFERENCE ON INTELLIGENT SCIENCE & TECHNOLOGY

on

24th & 25th MARCH 2011

This is to Certify that

Shri / Dr / [✓ Ms] I. BEULAH JOYSON

[✓ Student] / Research Scholar/Faculty of KUMARAGURU College of Techno

has presented a paper entitled IMPLEMENTATION OF NETWORK [illegible]

USING SOCKET PROGRAMMING IN [illegible]

CONTROL S[illegible]

in the Third International Conference on Intel[illegible]

[illegible]

[illegible]

[illegible]

CONVENER

Notification of Acceptance of the ICNCS 2011

April 8-10, 2011, Kanyakumari, India

http://www.icncs.org

Dear Beulah Joyson.J and Thilagavathi.K,

Paper ID : S12125

Paper Title : NETWORK REDUNDANCY MANAGEMENT IN PC BASED CONTROL SYSTEM

Congratulations! The review processes for 2011 International Conference on Network and Computer Science (ICNCS 2011) have been completed. The conference received submissions from nearly 21 different countries and regions, which were reviewed by international experts, and about 250 papers have been selected for presentation and publication. Based on the recommendations of the reviewers and the Technical Program Committees, we are pleased to inform you that your paper identified above has been accepted for publication and oral presentation. You are cordially invited to present the paper orally at ICNCS 2011 to be held on, April 8-10, 2011, Kanyakumari, India.

ICNCS 2011 is sponsored by IACSIT, and hosted by Vi Institute of Technology, Chennai and Marthandam college of Engineering and Technology, and technically co-sponsored by IETE Trivandrum-Nagercoil PAC and IEEE.

(Important) So in order to the register the conference and have your paper included in the proceeding successfully, you must finish following SIX steps.

1. Revise your paper according to the Review Comments in the attachment carefully.

2. Format your paper according to the Template carefully.
http://www.icncs.org/ieee.doc (DOC Format)

3. Download and complete the Registration Form.
http://www.icncs.org/reg.doc (English)

4. Finish the payment of Registration fee by Credit Card. (The detailed information can be found in the