



LAUNCH COMPUTER SIMULATION SOFTWARE

By

R.RATHNA BHARATHI

Reg. No 71202702012

of

Kumaraguru College of Technology

A PROJECT REPORT

Submitted to the

FACULTY OF SCIENCE AND HUMANITIES

In partial fulfillment of the requirements

for the award of the degree

of

MASTER OF SCIENCE

IN

APPLIED SCIENCE-COMPUTER TECHNOLOGY

June, 2004



BONAFIDE CERTIFICATE

**Certified that this project report titled
LAUNCH COMPUTER SIMULATION SOFTWARE**

IS A BONAFIDE WORK OF

Ms. R.RATHNA BHARATHI (Reg. No: 71202702012)

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 report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

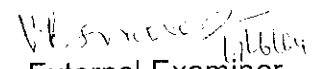

Project Guide


Head of the Department

The Candidate with University Register No. 71202702012 was examined

by us in the project Viva-Voce examination held on 17.06.2004


Internal Examiner
(17.6.2004)


External Examiner



सं. आर सि आई /
No. RCI /
अनुसंधान एवं विकास संगठन (रक्षा मंत्रालय)
Research & Development Organisation (MOD)
अनुसंधान केन्द्र इमारत
RESEARCH CENTRE IMARAT
विज्ञान काँचा - पि. ओ.
VIGNYANA KANCHA - P.O.
हैदराबाद - 500 069.
HYDERABAD - 500 069.

CERTIFICATE

This is to certify that this project work entitled “Launch Computer Simulation Software”, through serial link in Linux using ‘C’ Language was successfully carried out by

R.Rathna Bharathi

Reg.No:-71202702012

in record of bonafide work completed at **Research Center Imarat** in the partial fulfillment of the award of Master of Computer Technology from Kumaraguru College of Technology Affiliated to Anna University under the guidance of Smt. J. Sujatha, Sc- ‘C’, On Board Computer Division, R.C.I.

(Smt. J.Sujatha)
Scientist – ‘C’
OBCD



(Shri BHVSN Murthy)
Scientist – ‘E’
Head OBCD

BHVS Narayana Murthy, Sc ‘E’
Head OBCD
RESEARCH CENTRE IMARAT
Ministry of Defence
HYDERABAD-500 069.

ABSTRACT

The aim of the project is to develop PC based simulation software, which will simulate the launch control PC operations and communicate with the On-Board Computer through serial link in non-real mode. This software does the check-out functions in a systematic manner so that the Avionics system can be launched in a healthy condition.

There are two specific requirements for this purpose. The first objective is to establish a communication between Launch Control PC and On-Board Computer. A detailed analysis of serial communication is to be carried out which satisfies all the requirements for the communication.

The second objective is issuing the commands to clear the OBC & checking for its response using serial drivers. In this project work all the above requirements are successfully met with serial cable.

In particular it is developed with a user friendly GUI facilitating the user to select the options for COM port setting of the PC for communication with On-Board Computer, read the missile related data files and to perform the functions of Prelaunch (Checkout) and Autolaunch.

ACKNOWLEDGEMENT

I consider myself for having been permitted to do project work at **Defence Research and Development Organization (DRDO)**. It is a privilege for me to do the project in **Research Center Imarat (RCI)**, as it is an organization which does the supportive work in Defence research supporting the armed forces in INDIA.

I am indebted to **Dr.K.K.Padmanabhan**, Principal, Kumaraguru college of Technology, Coimbatore for permitting me to undertake the project work at DRDO.

I express my sincere gratitude to my project guide **Smt. J.Sujatha**, Scientist 'C', of On Board Computer Division (OBCD), RCI for her valuable guidance and constant encouragement at each stage of the project.

I wish to express my heart felt gratitude and thanks to my guide **D.Chandrakala**, Senior Lecturer for her valuable guidance and moral support during the entire tenure of the project work.

I am very grateful to **Mr. BHVS Narayan Murthy**, Scientist 'E', Head of the Department and Director **Mr. B.Rajendiran**, Scientist 'G' for giving me a chance to undertake a project in the OBCD/ISG of RCI. Also I would like to thank **Mr. Deepak Bekal Sc-'C'** of OBCD for his encouragement throughout the project.

Finally, I would like to express my thanks to all those who have directly or indirectly helped to realize my goals through this project.

TABLE OF CONTENTS

ABSTRACT	iii
LIST OF TABLES	vii
LIST OF FIGURES	vii
LIST OF ABBREVIATIONS	viii
CHAPTER 1: INTRODUCTION	1
1.1 Organization Profile	1
CHAPTER 2: GENERAL INFORMATION	4
2.1 Avionics Launch Control Computer System	4
2.2 Cyclic Redundancy Check (Checksum)	5
2.3 Serial communication	6
2.3.1 Serial Ports	6
2.3.2 Procedure to communicate using serial transmission	7
2.3.3 Serial Port Configuration	8
2.3.4 RS-232 standard	8
2.3.5 Signal Definition	10
2.3.6 Connecting up RS-232C Equipment	12
2.4 Linux	13
2.4.1 Overview	13
2.4.2 Features	14
2.4.3 Why Linux	14
2.5 Language and GUI	15
2.5.1 C programming Language	15
2.5.1.1 Features	15
2.5.1.2 The GNU C Compiler	16
2.5.2 GUI	17

2.5.2.1	GLADE Overview	17
2.5.2.2	Advantages	19
2.5.2.3	Disadvantages	19
CHAPTER 3: PROJECT DESCRIPTION		20
3.1	Software Design	20
3.1.1	Data Flow Diagram	20
3.1.2	Flow Chart Diagram	26
3.2	Software Description	27
3.2.1	System Description	27
3.2.2	Software Modules	31
3.2.3	GUI windows	32
3.2.4	Results	34
CHAPTER 4: SYSTEM CONFIGURATION		35
4.1	Hardware	35
4.2	Software	35
CHAPTER 5: CONCLUSION		36
APPENDIX		
REFERENCES		

LIST OF TABLES

	Page no.
TABLE 2.1 - PIN CONFIGURATIONS OF RS 232	10

LIST OF FIGURES

FIG.2.1 - BIT FORMAT	6
FIG. 3.1 - CONTEXT FLOW DIAGRAM	22
FIG.3.2 - 1 -LEVEL DATA FLOW DIAGRAM	22
FIG.3.3 - 2 -LEVEL DATA FLOW DIAGRAM	23
FIG.3.4 - 2 -LEVEL DATA FLOW FOR AUTOLAUNCH	25
FIG 3.5 - LC TO OBC DATA FLOW	26
FIG.3.6 - BLOCK DIAGRAM	29
FIG.3.7 - DETAILED BLOCK DIAGRAM	30

LIST OF ABBREVIATIONS

- | | |
|--------|-----------------------------|
| 1. LC | Launch Computer |
| 2. OBC | On Board Computer |
| 3. CRC | Cyclic Redundancy Check |
| 4. RB | Reset Board |
| 5. VIU | Vehicle Interface Unit |
| 6. NS | Navigation System |
| 7. ADC | Analog to Digital Converter |
| 8. RGP | Rate Gyro Package |

CHAPTER 1

INTRODUCTION

1.1 ORGANIZATION PROFILE

After Independence of India, a body called Defence Science Organization was established in 1948 for the development of Defence Science. A few other technical establishments were also formed. In 1958 a major reorganization was implemented by which the present DRDO (Defence Research Development Organization) was formed. The Defence science organization with some of the existing technical development establishments did the formation of DRDO. DRDO has gone through a phased program build up of infrastructures and expansion of various fields of Defence Science and Technology.

At the time of formation, DRDO consisted of science laboratories and small units attached to DRDO and there were only 11 laboratories all over the country attached to it. DRDO covers practically all the scientific and technological disciplines of defence interest. Among these laboratories RCI (Research Center Imarat) is one of the major laboratories.

As a supporting organization for armed forces, the main function of RCI and other laboratories is as follows:

- To design and develop weapons and equipments based on the operational requirements defined by the services and to help in the indigenous production.
- To render scientific advice to the service head quarters.
- To carry out basic and applied research to solve the problems.
- To evaluate and conduct the technical trails of new weapons and equipment's which are designed and developed in the country.
- To render technical supports to civil trade for the developments.
- RCI plays vital role in technical development for national security.

The Research Center Imarat is one of the biggest laboratories in the country under DRDL (Defence Research and Development Laboratory). RCI is a large Research and Development laboratory engaged in the development of missiles.

The whole infrastructure of RCI consists of large manpower, sophisticated machines and modern computer systems with advance management system. State of the art technology has been utilized wherever the need was felt for producing high performance missiles.

Development of missiles system is computerized and software intensive computers are being used for simulation, modeling of the system, development of subsystem and checkout, launching, control and guidance of the missiles. Majority of the software is developed in house in RCI/DRDL.

RCI is a unit under DRDO and On Board Computer Division (OBCD) is one of the divisions of RCI, involved in making different applications. OBCD has an infrastructure with heterogeneous hardware and software such as LINUX, QNX, PSOS, Windows NT etc.

The various technologies that OBCD Division is dealing are:

- Design and Development of Embedded systems.
- Real-Time systems
- Object-Oriented technology
- Software engineering and CASE tools
- Client-Server technology
- Networking and Internet
- Graphical User Interface.

CHAPTER 2

GENERAL INFORMATION

2.1 AVIONICS LAUNCH CONTROL COMPUTER SYSTEM

The Avionics System is composed of a large number of highly complex and interdependent subsystems. In order to successfully accomplish the mission for which the system is designed, each element must perform in accordance with its apportioned part of the total pattern. It is the function of the system checkout to determine whether the fully assembled system is operable within prescribed tolerance and all electrical circuits are proper or if not which element is failing in its assigned function. Hence check-out function is to check the Avionics System in a systematic manner so that the system can be launched in healthy condition.

The total checkout can be categorized in three phases:

1. Sub-System level (Non-real Time): This is to ensure the health and to diagnose a fault in individual subsystem, assembled avionics system and avionics system launcher for flight. Checkout of any subsystem has two stages. Checking its own resources and checking the I/O resources at its command.
2. Prelaunch (Checkout): During this phase, flight data is loaded onto the On-Board Computer, same is read back to verify data validity. During this process, health of all other systems is monitored.

3. Autolaunch: In this phase a sequence of events take place with continuous surveillance of various parameters, the human intervention being limited only to HOLD based on the performance of the systems external to Avionics System. Any out of range parameter internal to the Avionics System will raise HOLD automatically by Launch Computer. During this phase the Avionics System is launched from a given launch point to the desired target.

Launch Computer is required to perform checkout and launch of the Avionics System situated at a distance of 500m to 1000m through serial link.

2.2 CYCLIC REDUNDANCY CHECK (CHECKSUM)

The Cyclic Redundancy Check (CRC) is a very powerful but easily implemented technique to obtain data reliability. The CRC technique is used to protect blocks of data called frames. Using this technique, the transmitter appends an extra n-bit sequence to every frame called Frame Check Sequence (FCS). The FCS holds redundant information about the frame that helps the transmitter detect errors in the frame. The CRC is one of the most used techniques for error detection in data communications. The advantages of this technique are extreme error detection capabilities, little overhead and ease of implementation.

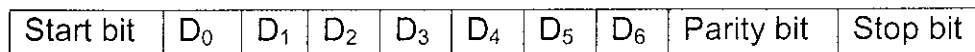
Different methods exist to calculate a check number for binary data, to be able to see if the data is not altered, for example, after being sent through some communication channel. CRC is a common method for protecting binary data that way. Different CRCs exist, which in the past has resulted in a naming scheme. Characteristic of this CRC is its 16 bits size and its initial value \$FFFF. It is also possible to encounter an initial value \$0000 too.

2.3 SERIAL COMMUNICATION:

Computers transfer information (data) one or more bits at a time. **Serial** refers to the transfer of data one bit at a time. Serial communications include most network devices, keyboards, mouse, modems, and terminals.

When doing serial communications each word (i.e. byte or character) of data sent or received is sent one bit at a time. Each bit is either *on* or *off*. The terms used are *mark* for the *on* state and *space* for the *off* state.

The speed of the serial data is most often expressed as bits-per-second ("bps") or baud rate ("baud"). It represents the number of ones and zeroes that can be sent in one second. It can have 1 Kbps or 1Mbps. The bit format for serial communication is given in fig. 1.



D₀ to D₆ refers to the data.

Fig.2.1 Bit Format

2.3.1 Serial Ports

The serial ports are used primarily for devices that must communicate bidirectional with the system. Such devices include modems, mouse, scanners, digitizers and any other devices that "talk to" and receive information from the PC. Most of the PCs have two serial ports. The first port is called COM1 and second is called as COM2.

Each character sent over a serial connection is framed by a standard start and stop signal. A single 0 bit, called the start bit, precedes each character to tell the receiving system that the next 8 bits constitute a byte of data as shown in fig.1. One or two stop bits follow the character to signal that the character has been sent. At the receiving end of the communication, characters are recognized by the start and stop signals instead of by the timing of their arrival.

Serial refers to data sent over a single wire, with each bit lining up in a series as the bits are sent. This type of communication is used over the phone system, because this system provides one wire for data in each direction. Serial ports may connect to a variety of devices such as modem, plotters, printers, other computers, bar code readers, scales, and device control circuits. Anything that needs a two-way connection to the PC uses the industry standard *Recommended Standard* number 232 revision c (RS-232c) serial port.

The heart of any serial port is the *Universal Asynchronous Receiver/Transmitter* (UART) chip. This chip completely controls the process of breaking the native parallel data within the PC into serial format, and later converting serial data back into the parallel format.

2.3.2 Procedure to communicate using serial transmission

- Step 1: Accessing a serial port
- Step 2: Opening a serial port (Read-Write mode)
- Step 3: Writing and reading data from serial port
- Step 4: Closing a serial port

3 Serial port configuration

Each time a character is received by a serial port, it has to get the attention of the computer by raising an *Interrupt Request Line* (IRQ). Eight-bit ISA (Industry Standard Architecture) bus systems have eight of these lines, and systems with a 16-bit ISA bus have 16 lines. In a standard configuration, COM1 uses IRQ4, and COM2 uses IRQ3.

Configuring a serial port using C language involves

Control Options: It involves controlling

- Baud rate setting
- Data bits setting
- Stop bit setting
- Enable receiver
- Enable / Disable parity bit

Input Options:

- Parity check
- Software flow control (outgoing and incoming)

It allows user to specify the baud rate, parity, and the number of data bits and stop bits. By default, the baud rate is set at 9600. The parity setting is for data validation. It is commonly not used, and set to "N". The data bits setting specifies the number of bits that represent a chunk of data. The stop bit indicates when a chunk of data has been received.

2.3.4 RS-232 Standard

RS-232 is a standard electrical interface for serial communications defined by the Electronic Industries Association ("EIA"). RS-232 comes in 3 different flavors (A, B, and C) with each one defining a different voltage range for the *on* and *off* levels. The most commonly used variety is RS-232C, which defines a mark (*on*) bit as a voltage between -3V and -12V and a space (*off*) bit

between +3V and +12V. The RS-232C specification says these signals can travel about 50 feet (16.4m) before they become unusable. Signals can travel a bit farther than this as long as the baud is low enough.

Modems and other devices used to send serial data are often referred to as *Communication Equipment* or DCE. The terminals or computers that send or receive the data are referred to as *data terminal equipment* or DTE. In response to the need for signal and handshake standards between DTE and DCE, the EIA developed EIA standard RS-232.

An RS-232 port can supply only limited power to another device. The number of output lines, the type of interface driver IC, and the state of the output lines are important considerations. The mode of operation of RS-232 is single-ended. Data rates of up to 20k bits/second and distance up to 50 Ft. can be accommodated with RS-232. In RS-232 uses only one driver and one receiver. This standard describes the

- Functions of 25 signals and handshake pins for serial data transfer.
- Voltage levels
- Impedance levels
- Rise and Fall times
- Maximum Bit Rate
- Maximum capacitance for signal lines.

The most commonly used connectors are DB-25P and DB-25S.

Besides wires for incoming and outgoing data, there are others that provide timing, status, and handshaking:

	Description	Pin	Description
	Earth Ground	14	Secondary TXD
	TXD - Transmitted Data	15	Transmit Clock
	RXD - Received Data	16	Secondary RXD
	RTS - Request To Send	17	Receiver Clock
5	CTS - Clear To Send	18	Unassigned
6	DSR - Data Set Ready	19	Secondary RTS
7	GND - Logic Ground	20	DTR - Data Terminal Ready
8	DCD - Data Carrier Detect	21	Signal Quality Detect
9	Reserved	22	Ring Detect
10	Reserved	23	Data Rate Select
11	Unassigned	24	Transmit Clock
12	Secondary DCD	25	Unassigned
13	Secondary CTS		

Table 2.1 Pin Configurations of RS 232

2.3.5 Signal Definitions

The RS-232 standard defines some 18 different signals for serial communications. Of these, only six are generally available in the UNIX environment.

GND - Logic Ground

Technically the logic ground is not a signal, but without it none of the other signals will operate. Basically, the logic ground acts as a reference voltage so that the electronics know which voltages are positive or negative.

TXD - Transmitted Data

The TXD signal carries data transmitted from your workstation to the computer or device on the other end (like a MODEM). A mark voltage is interpreted as a value of 1, while a space voltage is interpreted as a value of 0.

RXD - Received Data

The RXD signal carries data transmitted from the computer or device on the other end to your workstation. Like TXD, mark and space voltages are interpreted as 1 and 0, respectively.

DCD - Data Carrier Detect

The DCD signal is received from the computer or device on the other end of your serial cable. A space voltage on this signal line indicates that the computer or device is currently connected or on line. DCD is not always used or available.

DTR - Data Terminal Ready

The DTR signal is generated by your workstation and tells the computer or device on the other end that you are ready (a space voltage) or not-ready (a mark voltage). DTR is usually enabled automatically whenever you open the serial interface on the workstation.

CTS - Clear To Send

The CTS signal is received from the other end of the serial cable. A space voltage indicates that it is alright to send more serial data from your workstation. CTS is usually used to regulate the flow of serial data from your workstation to the other end.

RTS - Request To Send

The RTS signal is set to the *space* voltage by your workstation to indicate that more data is ready to be sent. Like CTS, RTS helps to regulate the flow of data between your workstation and the computer or device on the other end of the serial cable. Most workstations leave this signal set to the space voltage all the time.

2.3.6 Connecting Up RS-232C Equipment

To connect the terminal directly to the computer rather than through the modem-modem link, both terminal and computer should have DB-25 type so that the terminal cable can be plugged into the computer. Here they both try to input or output data through the same line. A solution to this problem is to make an adapter with two connectors so that the signals cross over. This crossover connection is often called a *null modem*. The handshake signals are also crossed over so that each handshake output signal is connected to the corresponding input signals.

RS-422 Standard

The mode of operation in RS-422 is differential data transmission. Differential data transmission is adequate for communicating at high data rates, or over long distance in real world environments. Differential data transmission offers superior performance in most application.

RS-422 was designed for greater distance and higher Baud rates than RS-232. A pair of converters from RS-232 to RS-422 can be used to form an "RS-232 extension cord". Data rates of up to 100k bits/second and distance up to 4000 Ft. can be accommodated with RS-422. RS-422 is also specified for multi-drop applications where only one driver is connected to, and transmits on, a "bus" of up to 10 receivers.

LINUX

Overview

LINUX is an operating system that was developed based on UNIX kernel, the low level core of an operating system. UNIX operating system was originally developed at Bell Laboratories. Designed in 1970s for Digital Equipment PDP computers, it has become a very popular multi-user, multitasking operating system for wide variety of different hardware platforms. LINUX is freely distributed implementation of a UNIX like kernel. LINUX takes the inspiration from UNIX thus both LINUX and UNIX programs are similar. In fact, almost all programs written in UNIX can be compiled and run on Linux too.

Linus Torvalds developed LINUX at the University of Helsinki with the help of UNIX programmers from across the Internet. Thus Linux kernel doesn't use code from AT&T or any other proprietary source thus making it free software from restrictions subject to GNU general public license. Some software under this GPL in Linux includes:

- GCC A C compiler
- G++ A C++ compiler
- GDB A source code level debugger
- GNU make A version of UNIX make
- Bison A parser generator compatible with UNIX yacc
- Bash A command shell
- GNU Emacs A text editor and environment

LINUX utilities are very simple as a result its utilities are small and easy to understand. We have many reusable components and we can create our own reusable components and add as libraries thus reducing the code and increasing the reusability of the code. Linux is highly flexible and highly secured

environment. LINUX is interchangeably used in reference to the Linux kernel, a Linux system, or a Linux distribution. It controls all hardware and provides higher-level abstractions such as processes, sockets, and files to the different software running on the system. Linux can also be used to designate a hardware system running the Linux kernel and various utilities running on the kernel. LINUX is well known for socket programming and shell programming which facilitates the programmer to communicate over any network.

2.4.2 Features

Linux supports a wide range of software, from TeX (a text formatting language) to X (a graphical user interface) to the GNU C/C++ compilers to TCP/IP networking. Linux is also compliant with the POSIX.1 standard, so porting applications between Linux and UNIX systems is a snap. UNIX is a trademark of X/Open. Linux is not a trademark, and has no connection to the trademark UNIX or X/Open. The following are some of its features:

- Linux helps to reuse yesterday's hardware.
- Linux is more stable
- Linux is more functional and feature-rich
- Linux comes with all the development tools.
- Linux is multi-purpose.

2.4.3 Why Linux:

Linux is a reliable and most stable Operating System. The following features make it unique:

- Full multitasking and 32-bit support.
- The X Window System. A complete version of the X Window System, known as XFree86, is available for Linux. The X Window System is a very powerful graphics interface, supporting many applications.
- TCP/IP (Transmission Control Protocol/Internet Protocol) support.

- Virtual memory support and shared libraries.
- The Linux kernel uses no code from AT&T or any other proprietary source.
- Linux supports (almost) all of the features of commercial versions of UNIX.
- GNU software support.
- Linux is compatible with the IEEE POSIX.1 standard.
- Built-in support for networking, multitasking, and other features.
- Linux is cheaper to get than most commercially available UNIX systems and UNIX clones.

The most important advantage of using Linux is that all of the kernel source code is available for Linux, and it is possible to modify it to suit the user's needs.

2.5. LANGUAGE AND GUI

2.5.1 C Programming Language

C is a general-purpose programming language that has been around since the early days of the UNIX operating system. It was originally created by Dennis Ritchie at Bell Laboratories to aid in the development of UNIX. The first versions of UNIX were written using assembly language and a language called B. C was developed to overcome some of the shortcomings of B. Since that time, C has become one of the most widely used computer languages in the world.

2.5.1.1 Features

- It is a very portable language. Almost all the computers has at least one C compiler available for it, and the language syntax and function libraries

are standardized across platforms. This is a very attractive feature for developers.

- Executable programs written in C are fast.
- C is the system language with all versions of UNIX.

The C compiler that is available for Linux is the GNU C compiler, abbreviated GCC. This compiler was created under the Free Software Foundation's programming license and is therefore freely distributable.

2.5.1.2 The GNU C Compiler

The GNU C Compiler (GCC) that is packaged with the Red Hat Linux distribution is a fully functional, ANSI C compatible compiler.

Invoking GCC

The GCC compiler is invoked by passing it a number of options and a number of filenames. The basic syntax for invoking gcc is this:

```
gcc [options] [filenames]
```

The operations specified by the command-line options will be performed on each of the files that are specified on the command line.

GCC Options

There are more than 100 compiler options that can be passed to GCC.

```
gcc -p -g test.c
```

```
gcc -pg test.c
```

The first command tells GCC to compile test.c with profile information for the profile command and also to store debugging information within the executable. The second command just tells GCC to compile test.c with profile information for the gprof command.

When a program is compiled using gcc without any command-line options, it will create an executable file (assuming that the compile was successful) and call it **a.out**. For example, the following command would create a file named a.out in the current directory.

```
gcc test.c
```

To specify a name other than **a.out** for the executable file, the **-o** compiler option can be specified. For example, to compile a C program file named count.c into an executable file named count, you would type the following command.

```
gcc -o count count.c
```

The following file extensions are assumed to be used when using the language compilers, including gcc:

1. .c C program file
2. .h A preprocessor file

2.5.2 GUI

2.5.2.1 Glade Overview:

Glade is a GUI builder for the Gimp Toolkit (GTK). Glade is entirely written in C. It operates on an XML file and can output C source code. The different versions of GTK+ are:

- The old stable release, standard between 1999 and 2002 is GTK+ 1.2. This corresponds to gnome 1.2 and 1.4. The matching glade version is 0.6.5.
- The new stable release and upcoming standard is GTK+ 2.2.

Creating initial program files (glade)

Glade will generate the following files:

- **myprog.cc:** This file contains a sample main function. It is simple and can be extended up to user's choice.
- **autogen.sh:** Invoking this shell script configures and compiles the program (standard for gnome projects)
- **Makefile.am:** Automake / autoconf are the make files.
- **glademm_support.hh, glademm_support.cc:** These files contain helper functions.
- **window1_glade.hh:** This is the GUI class declaration file. For the GUI-class name '_glade' is appended to the widget name. Each class contains its child-widgets' declarations as members. This file should not be modified.
- **window1.hh:** These are the program's classes, no GUI parts beside the derivation. Signal handlers are declared as member functions (private by default). The program's variables and functions can be added here. *Glade-- will not add new callbacks to this file, it will not change it.*
- **window1_glade.cc:** This file contains the constructors and destructors for the GUI part of the program. It should not be edited. This may be opened to see, what the code looks like.
- **window1.cc:** This file contains constructors, destructors, signal handlers, custom functions etc. Good stuff for a constructor is reading of data (e.g. database interaction) or anything else which has to be done

before the user starts interacting with your program. Signal handlers react to user's actions.

It is possible to include and wrap any other class inside the expected header file.

2.5.2.2 Advantages

- Glade will not overwrite any of the code. Designing the GUI and programming the application is separated.
- Glade is pure C.

2.5.2.3 Disadvantages

- Using different file naming schemes is not yet supported.
- Source Writer is not yet runtime customizable (indentation of generated programs).
- gnomemm support is still under development (especially 2.x)

CHAPTER 3

PROJECT DESCRIPTION

3.1 SOFTWARE DESIGN

Software design is a iterative process through which requirements are translated into a "blueprint" for constructing the software. Initially, the blueprint depicts a holistic view of software. That is, the design is represented at high level of abstraction-a level that can be directly traced to specific data, functional, and behavioral requirements. As design iteration occur, subsequent refinement leads to design representations at much lower levels of abstraction.

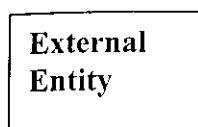
3.1.1 Data Flow Diagram

When information moves through software, it is modified by a series of transmissions. A data flow diagram (DFD) is a graphical technique that depicts information flow and the transforms that are applied as data move from input to output. The DFD is also known as a data flow graph or a bubble chart. The data flow diagram may be used to represent a system or software at any level of abstraction. In fact, DFDs may be partitioned into levels that represent increasing information flow and functional detail. The DFD provides a mechanism for functional modeling as well as information flow modeling.

A level 0 DFD, also called fundamental system model or context model, represents the entire software element as a single bubble with input and output data indicated by incoming and outgoing arrows, respectively. Additional

processes (bubbles) and information flow paths are represented as the level 0 DFD is the partitioned to reveal more detail. Each of the processes represented at level 1 are sub functions of the overall system depicted in the context model.

Basic DFD notations:



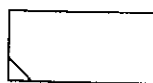
A producer or consumer of information that resides outside the bounds of the system to be modeled.



A transformer of information (a function) that resides within the bounds of the system to be modeled.



A data object; the arrowhead indicates the direction of data flow.



A repository of the data that is to be stored for use by one or more processes; may be as simple as a buffer or queue or as sophisticated as a relational database.

The following is the Context Flow Diagram which shows the main process interacting with its external entities representing the overall requirements of the system.

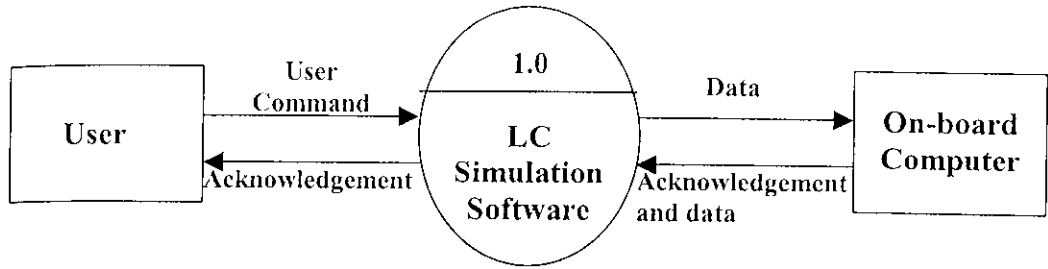


Fig. 3.1 Context Flow Diagram

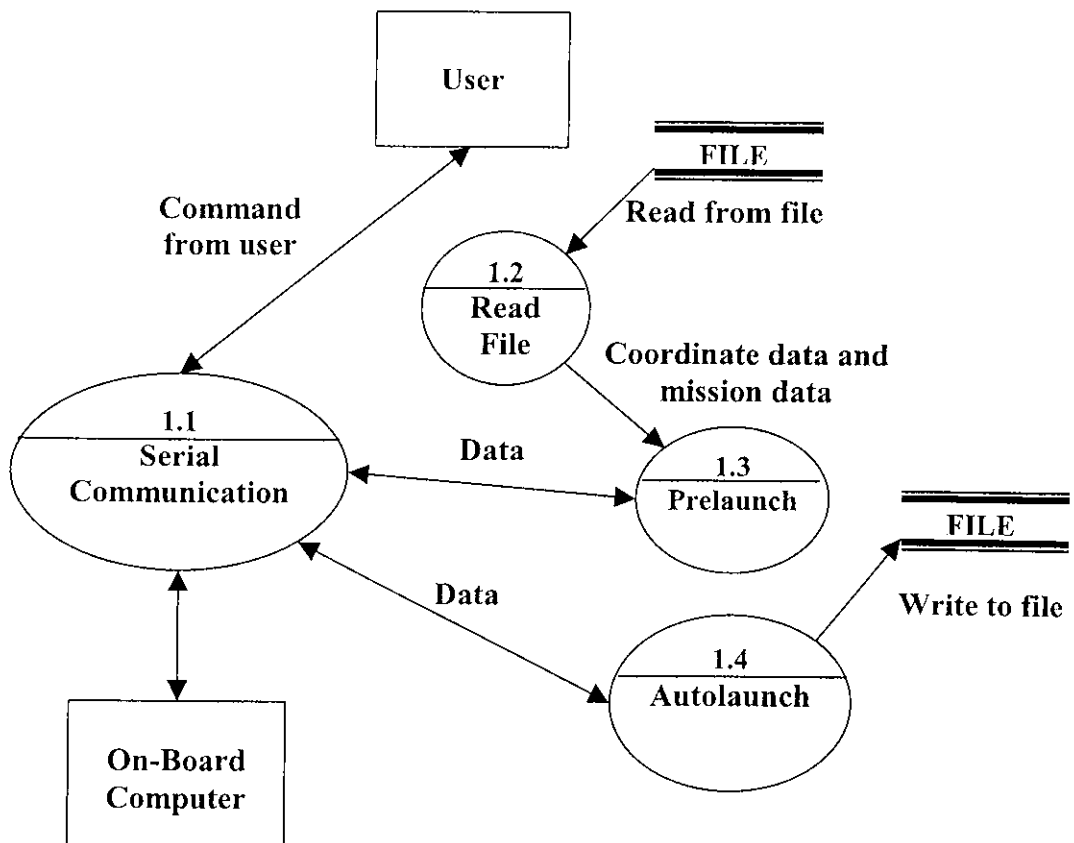


Fig.3.2 1-Level Data Flow Diagram

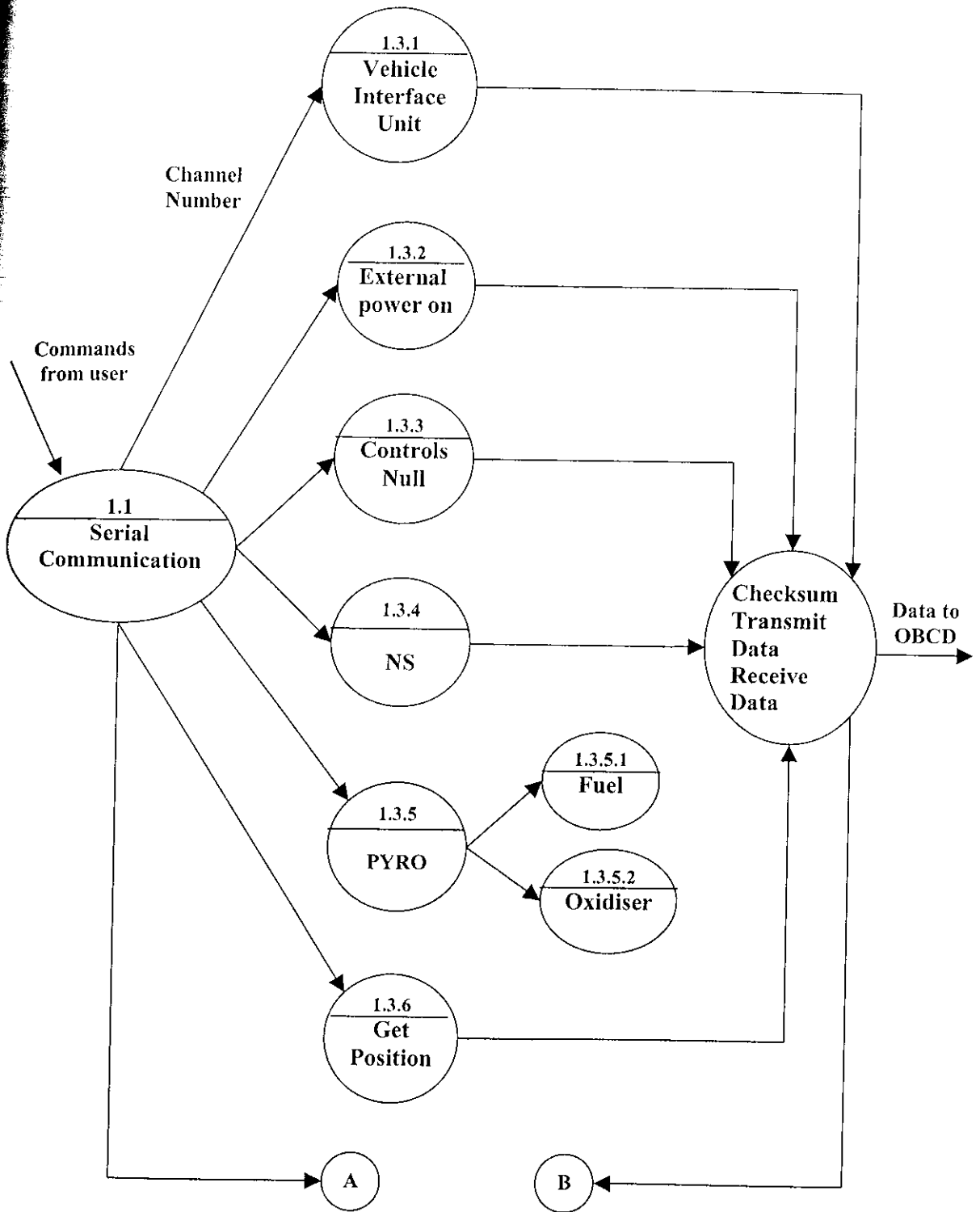
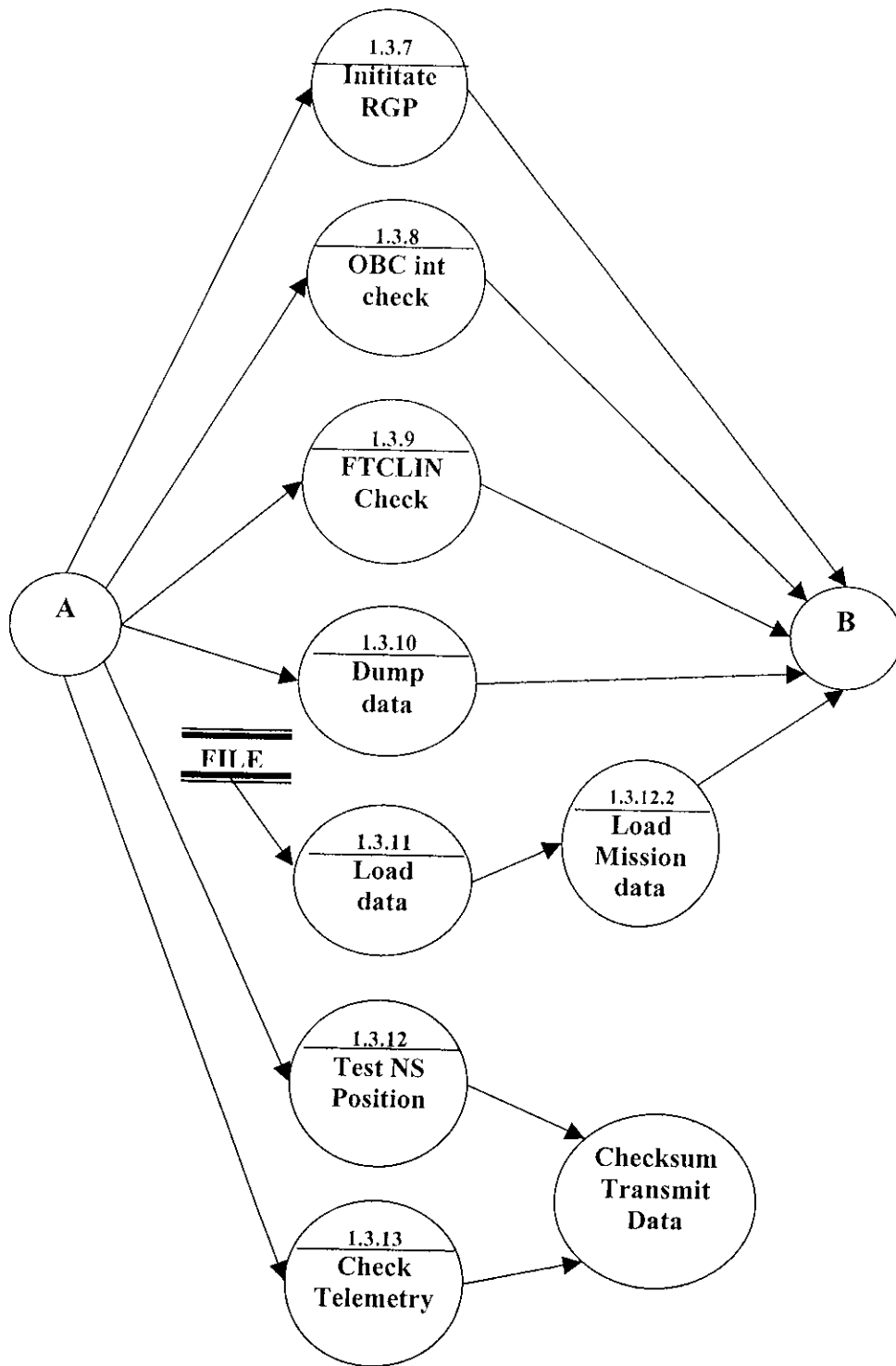


Fig.3.3 2-Level Data Flow Diagram



2- Level Data Flow for PreLaunch-Continuation

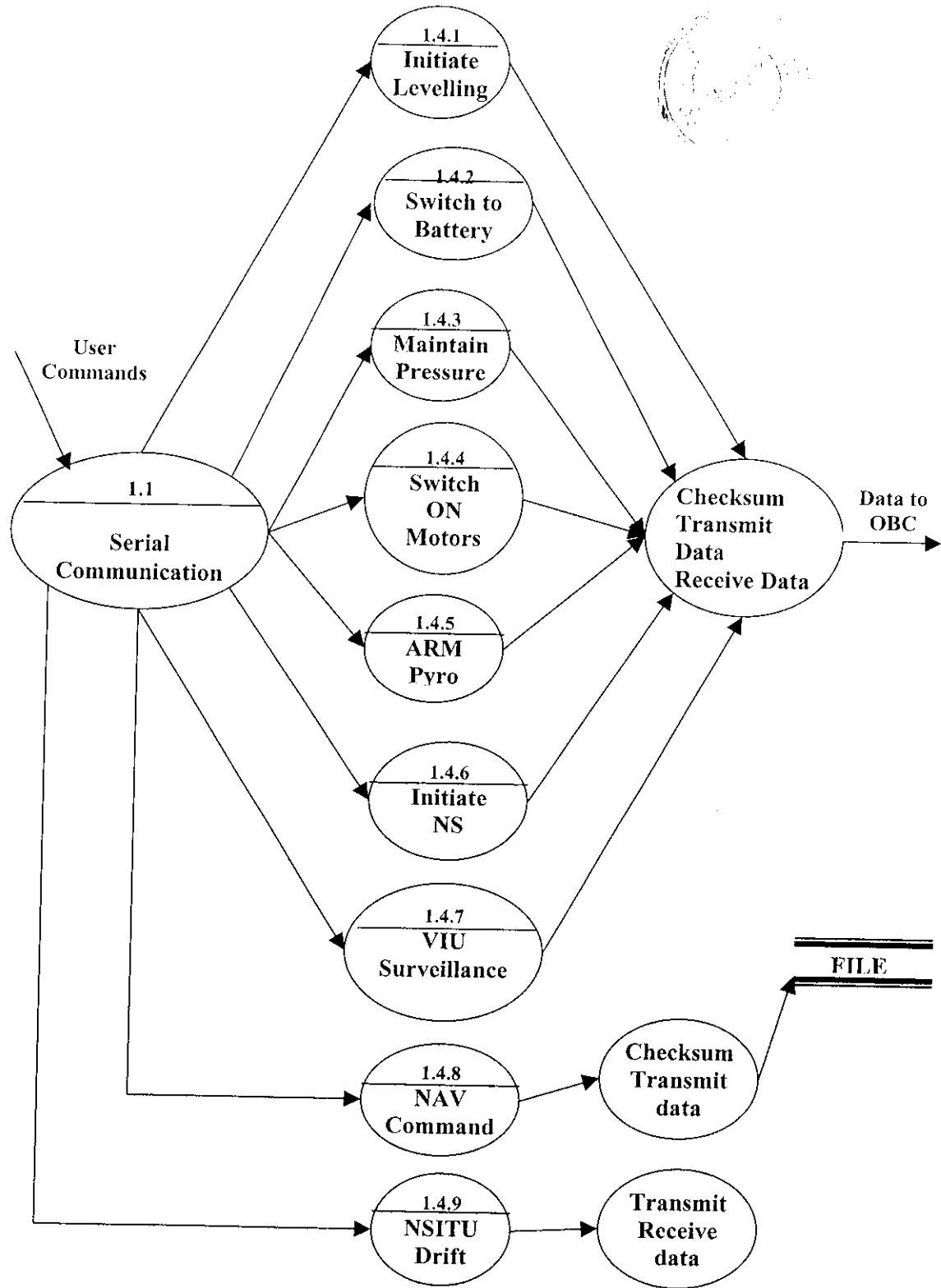


Fig.3.4 2- Level Data Flow for AutoLaunch

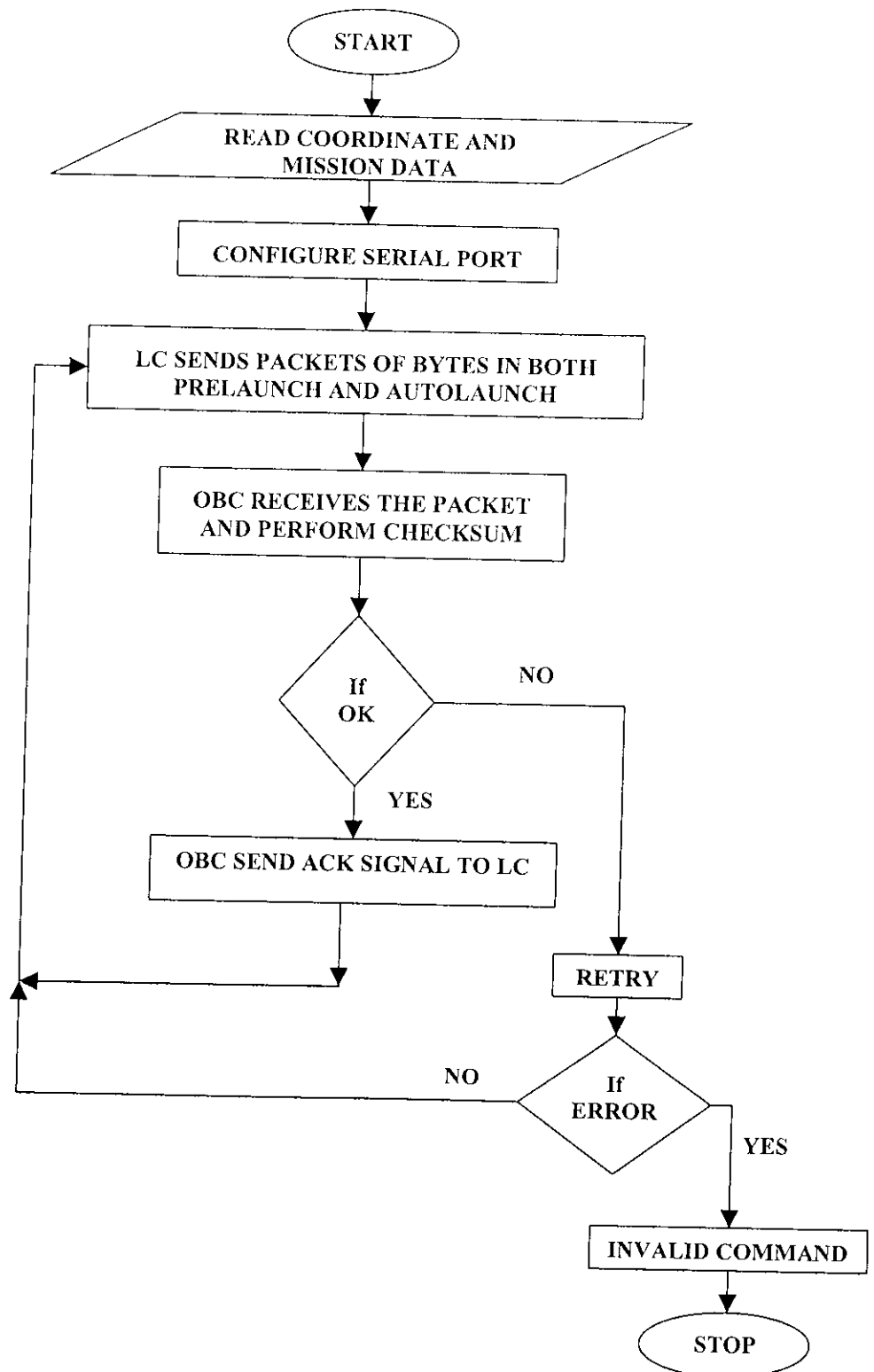


Fig 3.5. LC to OBC data flow

3.2 SOFTWARE DESCRIPTION

3.2.1 System Description:

The objective of the LC Simulation Software is to check the health of all the subsystems in the missile through serial communication. The LC Simulation Software PC is connected to On Board Computer through serial link. The software functions can be divided into the following categories:

1. Prelaunch
2. Autolaunch

Prelaunch:

Prelaunch is also called as checkout phase. In this phase the software has to check the health of the individual subsystems, data loading and switching on and off of the subsystems in accordance with On Board Computer.

During this the LC sends command to On Board Computer and receives an acknowledgement. It also receives the corresponding data and results and displays it. All the process during Prelaunch is done through user intervention.

Autolaunch:

During Autolaunch phase LC sends the command and receives the results without user intervention in real time. In this phase a sequence of events take place with continuous surveillance of various parameters with limited or no human intervention.

Postlaunch:

During post lift off LC- On-Board Computer communication doesn't exist. Navigation command is given through Autolaunch phase.

The LC- On-Board Computer protocol has five bytes of data in which the first four bytes are data and fifth byte is the checksum. The checksum is calculated for the validating the data. In this software the checksum is calculated by performing XOR for a given set of data and storing in a variable. This value is again XOR with new set of data. Many algorithms can also be implemented for calculating the checksum. One of them is the CRC method.

The following are the sequence of operations done by the Launch Computer Software:

- Configures COM port.
- Reads Mission data.
- Performs Prelaunch and Autolaunch operations.
- Receives the response from OBC.

The LC software allows the user to set the COM Port settings which includes setting Baud rate, Parity bit and size of data to be sent. A mission data file which has the coordinate and mission data are read and displayed. The software package is a menu driven, all the options of Prelaunch and Autolaunch are implemented according to the user's choice. The user is given a choice of operating either by clicking the buttons or through the menus.

The commands and data are sent from LC to On-Board Computer. The On-Board Computer receives the data from LC, calculates the checksum for it. If both the checksum matches then it sends back the acknowledgement.

On-Board Computer acts as a bus controller for all the other subsystems shown in fig.8. On-Board Computer communicates with all the other subsystems and monitors the status of all the subsystems. The On-Board Computer is connected to all other subsystems through MIL-STD 1553 link.

The LC communicates via RS-232 link whereas the On-Board Computer communicates via RS-422 link. A converter is needed to convert the RS-232 standard to RS-422 standard. This converter is also called as the Reset Board shown in fig 7.

The LC software deals with only the communication between the LC and the On-Board Computer.

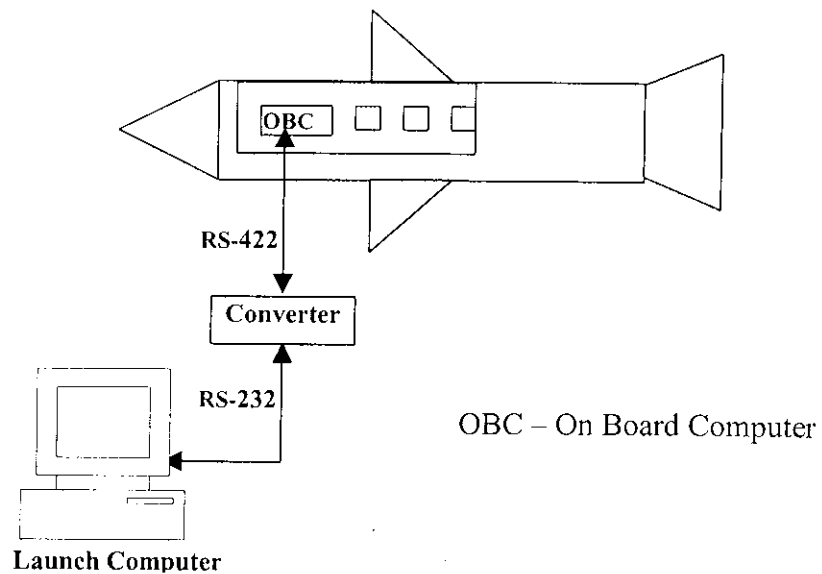


Fig.3.6 Block Diagram

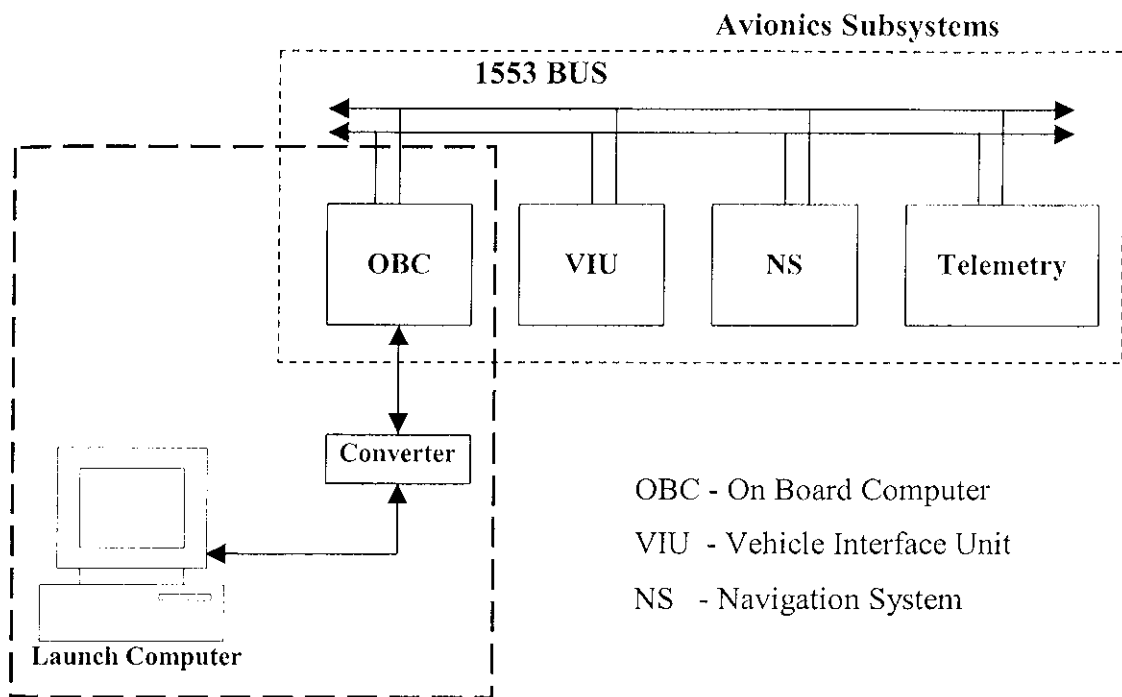


Fig.3.7 Detailed Block Diagram

Launch Computer:

Launch Computer is an industrial Pentium PC. It is connected on RS422 serial link with the On Board Computer. It performs the functions listed above. It has an efficient user interface and multitasking environment to execute concurrent activities in order to restrict the time response to the order of milliseconds. The launch computer is connected via RS422 link to the On Board Computer in Avionics System. The LC issues command to the Launcher

interface Unit for powering on the Avionics System. The Launcher Interface Unit which is near the Avionics System is connected by RS422link to the LC.

User Interface:

This is the interactive module to establish man machine interface. It is a pull down menu driven software that interacts with all functional modules. It consists of the following components:

1. Menu bar (horizontal/vertical)
2. Windows to display information
3. dialog boxes
4. data entry program
5. menu selection
6. hotkeys
7. print utility
8. Graphics utility to plot data in realtime/non realtime as per the user choice for linear bar/pie charts

The user interface has been designed based on experience from other projects and understanding the requirements of the user. It is very interactive and user friendly with most of the operations being self explanatory in their use.

3.2.2 Software Modules:

Software modules are developed in C language in Linux and the GUI is developed in GTK+ using Glade. The LC software has five major modules. The user is allowed to select any of the choice which is required. These modules are interactively invoked by the operator to perform various functions. These modules generate specific commands to interact with the

communication module and get data about the subsystems, perform processing and display in the desired format.

1. Read file: This module reads the coordinates and the mission data from the mission data file and displays the data file.
2. Serial Communication: In this module the PC COM port is configured by the user by selecting the options of COM port, Baud rate, Parity bit and size of the data to be sent. When the serial link is established the LC is ready to send its command and data to the On Board Computer.
3. Prelaunch: In the Prelaunch module the channel numbers and voltage value are selected by the user by forming five byte data which is sent to the On Board Computer. Also other operations like telemetry check, position check, controls checks, powering ON and OFF of the subsystems etc are performed.
4. Autolaunch: In this module the packetisation of five byte data is similar to the prelaunch module. The previous modules works under operator command, whereas this module once initiated by the operator, runs as per the external/internal clock. It generates the events that otherwise was generated by the functional module in case of operator initiated command.
5. Checksum: In the five byte data the fifth byte corresponds to the checksum. This checksum is calculated by performing XOR with the next data. The checksum is used for correctness of the data.

3.2.3 GUI Windows:

The LC software consists of five windows. All the windows in this software are menu and command driven. This software is secured, where

Unauthorized users cannot enter. Only valid users are allowed to use into this software. When the software is executed, it asks for the username and password. Valid users are allowed to enter whereas others are prompted for valid user and password. The following are the five windows involved in this software.

1. Main Window: The valid users are allowed to access the main window. This window provides the users with various options. The user can select any of the option either through menus or through the commands. When an option is selected, the user is linked to another window. The user is also allowed to return back to the main window from any of the window.
2. COM Port setting window: In this window the user is allowed to configure the COM port, Baud rate, Parity bit and size of the data. There are two options for the user after selecting the settings. The user can either apply the selected settings or set it to the default settings.
3. Read file window: In this window the user can view the coordinate and the mission data from the mission.dat.
4. Prelaunch window: The Prelaunch menu has many functions such as telemetry check, data loading and dumping, checking for pyro etc. The user can select any of the function required. After entering the function an acknowledgement from the OBC is received and the corresponding data are displayed in the textview.
5. Autolaunch window: The Autolaunch menu has functions like surveillance, switching off external power, start leveling, navigation command etc.

3.2.4 Results:

When the project is build *.c file are generated by glade. Only the `callbacks.c` file is modified and the logic is implemented. To compile the code the following commands are executed in sequence.

```
# ./autogen.sh  
# make
```

The LINUX based C code is compiled through the `make` command it automatically generates an executable file. This executable file can be run through,

```
./lcsim
```

where `lcsim` is the exe file.

When this command is issued first the ENTRY form is displayed. The ENTRY form is depicted in Fig.

The valid users are allowed to access the Main window which is depicted in Fig.

From the main window the user selects any of the choice and the corresponding window appears. The other windows are depicted in the Fig.

CHAPTER 4

SYSTEM CONFIGURATION

4.1 Hardware:

- Processor : Pentium III
- RAM : 64 MB
- Harddisk : 10 GB

4.2 Software:

- Operating System : Linux 8.0
- GUI development : GTK+ using Glade
- Programming Language : C

CHAPTER 5

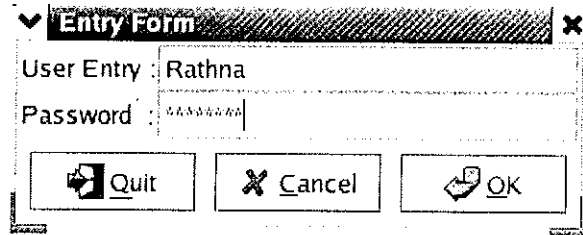
CONCLUSION

The Simulation Software is developed on Linux platform in C language with GTK+ using Glade as GUI. This GUI is user friendly menu driven software which facilitates the user to select the options of choice to run the program in Prelaunch and Autolaunch modes. Also the user can set the PC COM port to his choice of COM port, Baudrate, parity and other options. This is a general application and can be used for a specific missile by changing some of the modules or by adding some modules in the system.

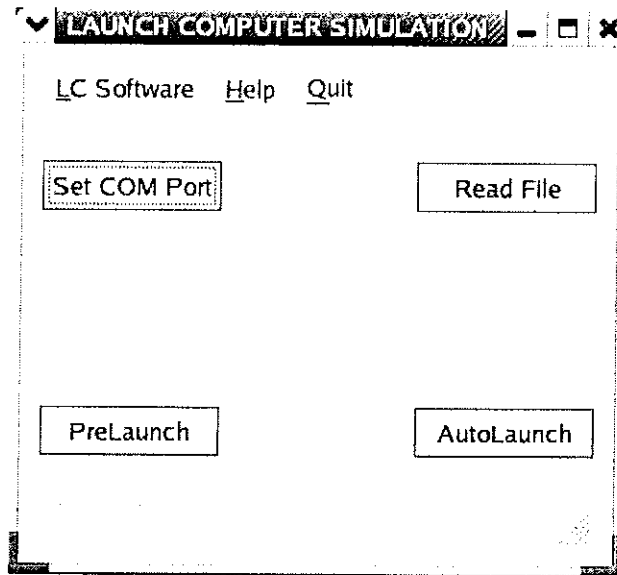
Improvements:

- i) This application can also be developed through MIL-STD-1553B serial link.
- ii) The GUI can be developed using Qt in Linux.

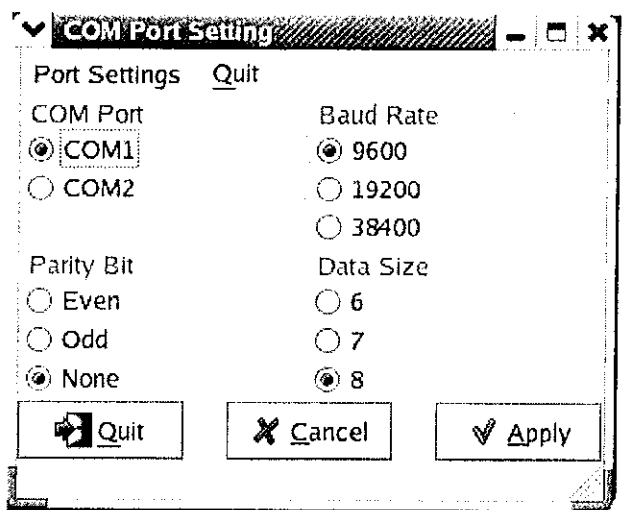
APPENDIX 1



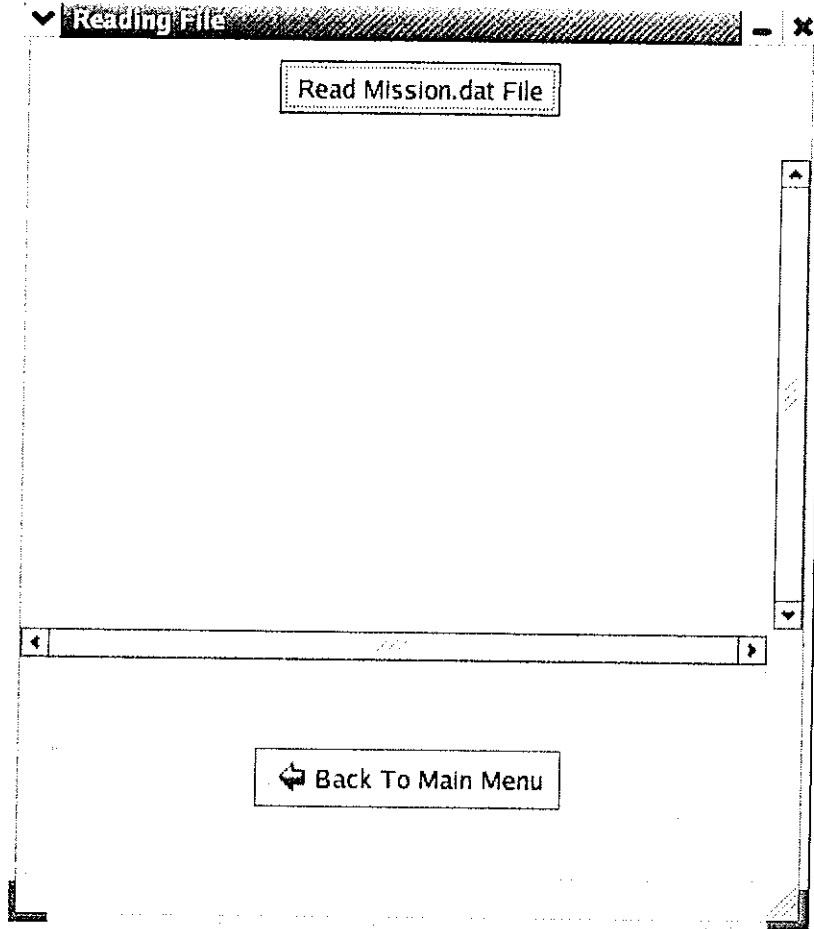
ENTRY FORM



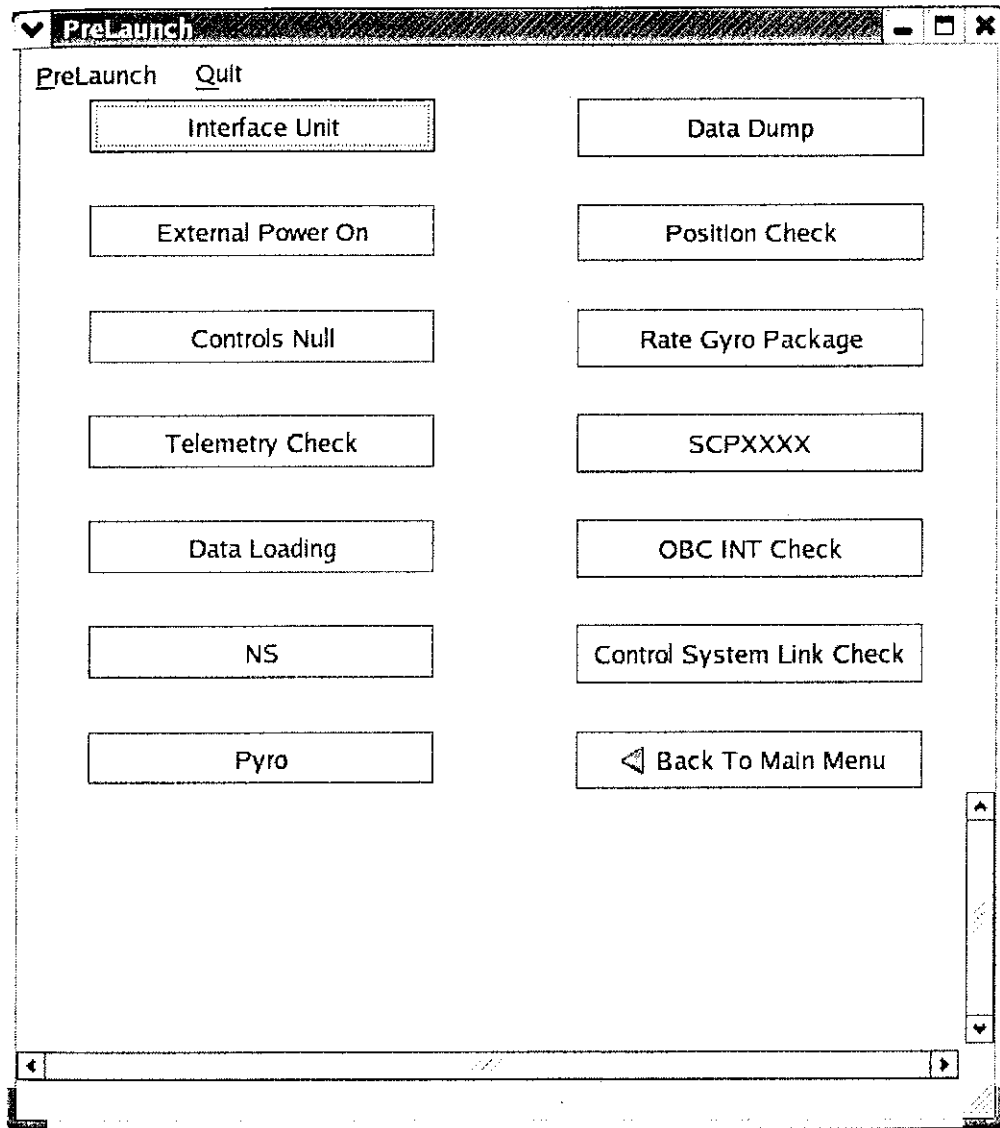
MAIN WINDOW



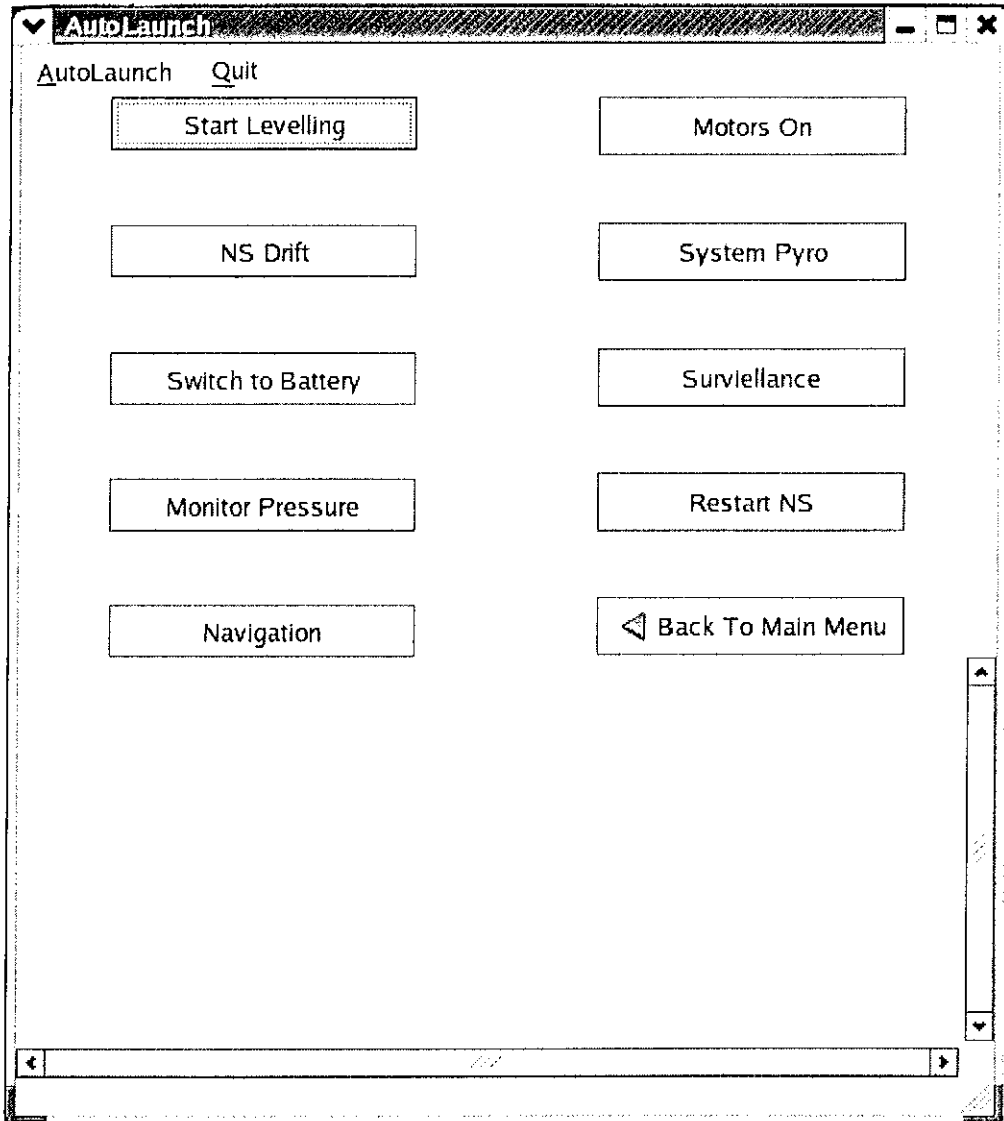
COM PORT SETTING WINDOW



READ FILE WINDOW



PRELAUNCH WINDOW



AUTOLAUNCH WINDOW

REFERENCES

1. *Denim Mac man* (1999) "Linux Networking" The Red hat Official Press
2. *Kallis Thomson* (1993) "Shell Programming in Linux " Tata McGraw Hill
3. *Thomas Kurt's* (1995) "Red hat Linux Networking Administration Tool"
The Red hat Official Press
4. *Yashwant Kanetkar* (1990) "The Mastering C Programming Language "
BPB Publications

