“ETHERNET AUTOMATION”

A DISSERTATION

Submitted in partial fulfillment of the requirement for the award of degree of

MASTER OF ENGINEERING

In

CONTROL & INSTRUMENTATION

JUNE 2009

Submitted by

G.S.PREETHI

University Roll No: 12238

Guided by

Dr. PARMOD KUMAR

Professor & Head

Department of Electrical Engineering
Delhi College of Engineering
University of Delhi
Delhi-110042

CERTIFICATE

This is to certify that the dissertation entitled ***“Ethernet Automation”*** has been carried out by ***G.S.Preethi***, University ***Roll No. 12238*** student of Master of Engineering, control & instrumentation, Department of Electrical Engineering, Delhi College of Engineering under my guidance in partial fulfillment of the degree of Master of Engineering in Control & Instrumentation Engineering during the academic session 2009.

This dissertation is a bonafide record of project work carried out by her under my guidance and supervision. Her work is found to be outstanding and has not been done earlier.

I wish her success in all her endeavors.

Dated:

Project Guide

Dr.Parmod Kumar
Professor & Head
Dept. of Electrical Engineering
Delhi College of Engineering

ACKNOWLEDGEMENT

I would like to express my gratitude to ***GOD*** for his inspiration, encouragement & countless blessings and to all who gave me the possibility to complete this thesis.

I would like to thank the ***HED department of STMicroelectronics***, Greater Noida for giving me permission to commence this thesis, to do the necessary research work and to use the departmental data and resources. I also want to thank my manager ***Mr. Manish umar Aggarwal &*** project guide ***Mr. Sudeep umar Srivastava*** for their stimulating support.

I have furthermore to thank my project guide ***Dr.Parmod umar***, Head of Electrical Engineering Department, Delhi College of Engineering who gave permission to carry out the research work in STMicroelectronics and encouraged me to go ahead with my thesis. I express my sincere gratitude to him for his constant support, Guidance and unflinching encouragement throught the development of this thesis.

I am deeply indebted to ***Mr. runal. .Patel*** System Engineer of Captronic Systems for teaching me the basics of Lab View programming & for dedicating his valuable time whenever I needed to discuss project related work.His help, stimulating suggestions and encouragement helped me throughout for this research work.

I am obliged to ***Mr.V.N.D.B.R Prasad Gutti*** for his support, help and valuable hints during my difficult times.

I am deeply indebted to my Father in law ***Mr.C.Gopinath Menon***, Mother in law ***Mrs.Giri a Menon*** & Husband ***Mr.Abhishek Menon*** for their sacrifice, inspiration, patient love and ever encouraging moral support which enabled me to pursue my studies & carry out this work.

G.S.Preethi

ABSTRACT

Automation technology has been characterized in the last few years by the increase in the use of distributed networks based on field bus technology. This in conjunction with the upcoming trend for a single network from the management level to the field level has brought Industrial Ethernet into the focus of industrial communication standards and technologies. Ethernet has become a prominent technology for an IP-based packet centric network infrastructure. Management of Ethernet networks with carrier –class reliability is generally a difficult issue for operators and service providers. Engineers designing or validating the Ethernet need to perform a wide range of tests, quickly, reliably and efficiently.

With rapid adoption of Ethernet standards, industries are facing constant pressure in designing and validating the physical layer of Ethernet devices and improve efficiency Once a design is complete, checking that all is well moves from being a logical activity to a physical one. . So mechanisms for problem and issue management need to be put in place to handle this. There are reasons why and times when a designer may get involved in network troubleshooting. This is when the network is exhibiting unexpected behaviour that cannot be explained by any known failure mechanism. Ethernet validation process involves rigorous testing that ensures validation of MAC layer, physical layer and protocol features of provided IP.

In this thesis an attempt is made to resolve the Ethernet validation challenges by developing an automation software for the data acquisition and analysis of the Ethernet validation system using *Application evelopment Environment* (ADE)- Lab VIEW , to resolve validation challenges quickly and efficiently.

List of Figures

Figure 1.1: Ethernet Logical Relationship to the ISO Reference Model 4

Figure 1.2: MAC and Physical Layer Compatibility Requirements.. 5

Figure 1.3: The Basic IEEE 802.3 MAC Data Frame Format 6

Figure 1.4: ST7xxx MBoard connections with Smart bits 16

Figure 2.1: Block diagram of Set Top Box Chip STi7xxx 20

Figure 2.2: System-level block diagram of GMAC... 21

Figure 2.3: System-level block diagram of PHY-DP83865 22

Figure 2.4: System-level block diagram of MII interface 24

Figure 2.5: System-level block diagram of RMII interface.. 25

Figure 2.7: Diagram of ST Micro Connect 2... 26

Figure 2.8: ST Micro Connect 2 interfaced with system.. 27

Figure 2.9: Main Frame Display of Logic Analyzer... 30

Figure 2.10: Back Panel of Logic Analyzer... 31

Figure 2.11: Probes connected to validation board 32

Figure 2.12: E5382A (single-ended) flying lead probe sets 32

Figure 2.13: Settings of State/Timing Modules 33

Figure 3.1: Lab VIEW Block Diagram Example 36

Figure 3.2: Represent parallel execution of Lab VIEW 38

Figure 3.4: State Diagram 39

Figure 3.5: Lab VIEW Front Panel 41

Figure 3.6: Lab VIEW Computing Targets 42

Figure 3.7: Lab View’s programming flow 45

Figure 3.8: The flow chart for using an ActiveX container 46

Figure 3.9: The flow chart for setting an ActiveX event sequence in Lab VIEW 47

Figure 4.1: Automation Set Up 51

Figure: 5.1: section1 of Main acquisition.VI 55

Figure: 5.2: Logic Analyzer Display before acquisition 56

Figure: 5.3: Logic Analyzer Display after acquisition 56

Figure: 5.4: Section 2 of Main Acquisition.VI 57

Figure: 5.5: Section 3 of Main Acquisition.VI 58
Figure: 5.6: Section 4 of Main Acquisition.VI 59
Figure: 5.7: Section 5 of Main Acquisition.VI 59
Figure: 5.8: Block Diagram View of Main Acquisition.VI 60
Figure: 5.9 VI Hierarchy 60
Figure: 5.10: Flow Chart of Main Acquisition.VI 61
Figure: 5.11: Initialization block 62
Figure: 5.12: Block Diagram of Sub VI Initial.VI 62
Figure: 5.13: Block Diagram of Description.VI 63
Figure: 5.14: Equivalent flow Chart Of Initial.VI & Description.VI 64
Figure: 5.15: Block Diagram of Trigger.VI 65
Figure: 5.16: Equivalent flow Chart of Trigger.VI........ 65
Figure: 5.17: Block Diagram of Acquire.VI 66
Figure: 5.18: Equivalent flow Chart of Acquire.VI 67
Figure: 5.19: Block Diagram of Path for ideal packet.VI 68
Figure: 5.20: Equivalent flow Chart of Path for Ideal Packet.VI 68
Figure: 5.21: Block Diagram of Byte to Nibbles.VI 69
Figure: 5.22: Equivalent flow Chart of Byte to Nibbles.VI 70
Figure: 5.23: Block Diagram of Extract Packet Re-entrant.VI 71
Figure: 5.24: Equivalent flow Chart of Extract Packet Re-entrant.VI 72
Figure: 5.25: Block Diagram of Extract Packet Trial.VI 73
Figure: 5.26 Equivalent flow Chart of Extract Packet Trial.VI 74
Figure: 5.27: Block Diagram of Extract Packets.VI 75
Figure: 5.28: Equivalent flow Chart of Extract Packets.VI 76
Figure: 5.29: Block Diagram of Comparison Module.VI 77
Figure: 5.30: Equivalent flow Chart of Comparison Module.VI 78
Figure: 5.31: Block Diagram of Save Bit Error.VI 79
Figure: 5.32: Equivalent flow Chart of Save Bit Error.VI 80
Figure 5.33: Block Diagram of Nibble to byte conversion of CRC & Preamble 81
Figure: 5.34: Block Diagram of Nibble to byte conversion of Extracted Packet 81
Figure: 5.35: Equivalent flow Chart of Packet Feature.VI 82

Figure: 5.36: Block Diagram of Save Indexes.VI 83
Figure: 5.37: Equivalent flow Chart of Save Indexes.VI 83
Figure 6.1: Ethernet packet format 87
Figure 6.2: Front Panel View after acquisition 91
Figure 6.3: Transmitted packet /Expected Packet (size 64 bytes) 91
Figure 6.4: Received Packet same as Expected Packet (size 64 bytes) 91
Figure 6.5: Front Panel showing Extracted Packet & Expected Packet 92
Figure 6.6: Bit Error packet of size 64 bytes 92
Figure 6.7: Packet with size mismatch (size 48 bytes) 92
Figure 6.8: Front Panel showing comparison results 93

List of Tables

Table 1.1 **Address Filtering Modes Test & Criteria for passing the test 13**

Table 1.2 **Jumbo Frame Tests & Criteria for passing the test 13**

Table 1.3 **IP Checksum Error Test & Criteria for passing the test.................................. 14**

Table 1.4 **Flow Control Test & Criteria for passing the test 14**

Table 1.5 **System Test & Criteria for passing the test 15**

Table 2.1 **ST Micro Connect 2 supported host connections 28**

Table 3.1 **Windows Requirements for Lab View Version 8.6 .... 42**

Table 5.1 **User Defined Inputs 57**

Table 6.1 **Test results for Packet size: 7968 bytes in State mode of Logic Analyzer........ 88**

Table 6.2 **Test results for Packet size: 64 bytes in State mode of Logic Analyzer 89**

Table 6.3 **Test results for Packet size: 7968 bytes in Timing mode of Logic Analyzer ... 90**

Table 6.4 **Test results for Packet size: 64 bytes in State mode of Logic Analyzer 90**

List of Tables

Table 1.1 **Address Filtering Modes Test & Criteria for passing the test 13**

Table 1.2 **Jumbo Frame Tests & Criteria for passing the test 13**

Table 1.3 **IP Checksum Error Test & Criteria for passing the test................................... 14**

Table 1.4 **Flow Control Test & Criteria for passing the test 14**

Table 1.5 **System Test & Criteria for passing the test 15**

Table 2.1 **ST Micro Connect 2 supported host connections 28**

Table 3.1 **Windows Requirements for Lab View Version 8.6 .... 42**

Table 5.1 **User Defined Inputs 57**

Table 6.1 **Test results for Packet size: 7968 bytes in State mode of Logic Analyzer........ 88**

Table 6.2 **Test results for Packet size: 64 bytes in State mode of Logic Analyzer 89**

Table 6.3 **Test results for Packet size: 7968 bytes in Timing mode of Logic Analyzer ... 90**

Table 6.4 **Test results for Packet size: 64 bytes in State mode of Logic Analyzer 90**

Table of Contents

Certificate i

Acknowledgement ii

Abstract iii

List of Figures iv

List of Tables vii

List of Abbreviations viii

Chapter I: Introduction .... (1-18)

1.1) Introduction 1

1.2) Objective 1

1.3) Motivation 2

1.4) Background of Ethernet 3

1.4.1) General description 3

1.4.2) The Basic Ethernet Frame Format 6

1.4.3) The Ethernet Physical Layers 9

1.5 Previous work 10

1.5.1) Validation Concept 10

1.5.2) Test case formulation 11

1.5.3) Validation Issues & Need for Automation Set Up for Validation 17

1.5.4) Conclusion 17

1.6) Problem formulation 17

1.7) Dissection of Thesis 18

Chapter II: Hardware Modules .... (19-34)

2.1) Introduction 19

2.2) Set Top Box Chip-STi7xxx 19

2.2.1) GMAC/ Ethernet 21
2.2.2) PHY - Ethernet Physical Layer 22
2.2.3) Different modes of MAC interface with PHY 23
2.2.4) PHY Management Control (SMI) 26

2.3) Interfacing module-ST Microconnect-2 27

2.3.1) Description 27
2.3.2) Functionality Features 28

2.4) Logic Analyzer- Agilent 16702B 29

2.4.1) Description 29
2.4.2) Functionality Features 29
2.4.3) Working 29
2.4.4) Main Frame Display 30
2.4.5) Main Frame Back Panel 31
2.4.6) Probing Criteria 32
2.4.7) Data Acquisition and Stimulus: State/Timing Modules 33

2.5 Conclusion 34

Chapter III: Software .... (35-48)

3.1) Introduction 35

3.2 Application Development Environment- Lab View 35

3.2.1) Brief History 35
3.2.2) Introduction 35
3.2.3) Key elements of the Lab VIEW development platform 35
3.2.4) System Requirements 42

3.3 Agilent's connectivity software-Intuilink 43

3.3.1) Description 43
3.3.2) System Requirements 43

3.4 Automation interfacing platform -Active X/COM 44

3.3.1) ActiveX Automation 44
3.3.2) ActiveX Controls and Containers 44
3.3.3) ActiveX Events 45
3.3.4) ActiveX Automation with Lab VIEW 45
3.3.5) Information about OLE Variants and Lab VIEW 47

3.5 Conclusion... 48

Chapter IV: Integration of Hardware modules & software ....(49-52)

4.1) Introduction... 49

4.2) Validation Set Up... 49

4.3) Validation Procedure 49

4.4) Steps for Loading Validation Code 50

4.5) Steps to form Configuration file of the logic analyzer for Test Set Up....................... 50

4.6) Interfacing of Lab VIEW & Logic Analyzer... 50

4.7) Automation Set Up 51

4.8) Working of Automation software integrated with Validation Set Up......................... 52

4.9) Conclusion 52

Chapter V: Program Development In Lab View...(53-85)

5.1) Introduction... 53

5.2) Scope & System Overview... 53

5.2.1) Basic Features 53
5.2.2) Application specific Features.. 53
5.2.3) Design Aspects.. 54

5.3) Software Model... 55

5.3.1) Front Panel Description 55
5.3.2) Block Dia gram Description 60
5.3.3) Sub VI’s Description........ 62
5.3.4) Global Variable Description 84
5.4) Conclusion 85

Chapter VI: Testing & Limitations........ (86-95)
6.1 Introduction........ 86
6.2 Case formulation 86
6.2.1) Format of Packet........ 87
6.2.2) Performance Evaluation in State Mode of Logic Analyzer........ 87
6.2.3) Performance Evaluation in Timing Mode of Logic Analyzer 89
6.3) Screen Shots of Graphic user interface........ 91
6.4) Design Hurdles & Solutions 93
6.5) Limitations........ 95
6.6) Conclusion 95

Chapter VII: Conclusions & Further Scope of Work .... (96-97)
7.1) Introduction........ 96
7.2) Conclusion 96
7.3) Further scope of work........ 97

References........ (98-100)

Curriculum Vitae

CHAPTER – I
Introduction

1.1) Introduction

The importance of automation in the process industries has increased dramatically in recent years. Due to the rising demand in factory automation for integrated communication over a single network the automation sector started to create devices that used Ethernet connections. In digital entertainment devices, set top boxes with networking capabilities are designed and sold. These set top boxes with the progress of time have become highly complex which has resulted in the software quality practices of the industry. With rapid adoption of Ethernet standards the engineers designing and validating the Ethernet MAC layer, physical layer of their devices face constant pressure to reduce the bugs and improve efficiency. Engineers need to perform a wide range of compliance tests quickly and reliably right on their bench. Once a design is complete, checking that all is well moves from being a logical activity to a physical one.. So mechanisms for problem and issue management need to be put in place to handle this. There are reasons why and times when a designer may get involved in network troubleshooting. This is when the network is exhibiting unexpected behaviour that cannot be explained by any known failure mechanism. This project is an attempt to study the design challenges faced during validation & to develop comprehensive, integrated compliance software for automating the data acquisition & analysis of Ethernet validation system using Lab View to resolve validation challenges quickly and efficiently.

1.2) Objective

In the present work *“ **comprehensive, integrated compliance software algorithm is required to be developed on Lab View platform to interface the Logic Analyzer with the Ethernet validation set up for Data acquisition and analysis to resolve the validation challenges efficiently.*** “

- **Functional features** developed in the proposed algorithm:

I.) Control of logic analyzer activation through Lab view.

II.) Design of acquisition module to run logic analyzer in both single and continuous acquisition mode.

III.) Acquire data captured by logic analyzer.

IV.) Design of extraction module to extract the relevant data from the acquired data.

V.) Design of comparison module to check the authenticity of the acquired data.

VI.) To save the Erroneous data & its details for future reference.

1.3) Motivation

Automation technology has been characterized in the last few years by the increase in the use of distributed networks based on field bus technology. This in conjunction with the upcoming trend for a single network from the management level to the field level has brought Industrial Ethernet into the focus of industrial communication standards and technologies. Ethernet provides transmitter resolution and has sufficient bandwidth and excellent expansion potential. Debugging today's digital systems is tougher than ever, increased product requirements, complex software, and innovative hardware technologies make it difficult to meet the market goals In the field of digital entertainment devices, set top boxes with networking capabilities have become highly complex which has resulted in the software quality practices of the industry. With rapid adoption of Ethernet standards the engineers designing and validating the Ethernet MAC layer, physical layer of their devices face constant pressure to reduce the bugs and improve efficiency. During the process of debugging and validating a digital system, a common task is the acquisition of data and its analysis. This project is an attempt to study the design challenges faced during validation by thoroughly studying the industrial validation scenario and developing an automation software using Lab View for data acquisition and analysis of Ethernet validation set up for easing the error tracking operation during validation. This project is fully done in ***ST MICROELECTRONICS, GREATER NOIDA.***

1.4) Theoretical background of Ethernet

1.4.1) General description

Ethernet is a family of frame-based computer networking technologies for local area networks (LANs). The name comes from the physical concept of the ether. It defines a number of wiring and signaling standards for the Physical Layer of the OSI networking model, through means of network access at the Media Access Control (MAC) /Data Link Layer, and a common addressing format. Ethernet is standardized as IEEE 802.3. The combination of the twisted pair versions of Ethernet for connecting end systems to the network, along with the fiber optic versions for site backbones, is the most widespread wired LAN technology. It has been in use from around 1980 to the present, largely replacing competing LAN standards such as token ring, FDDI, and ARCNET

Ethernet was originally based on the idea of computers communicating over a shared coaxial cable acting as a broadcast transmission medium. The methods used show some similarities to radio systems, although there are fundamental differences, such as the fact that it is much easier to detect collisions in a cable broadcast system than a radio broadcast. The common cable providing the communication channel was likened to the ether and it was from this reference that the name "Ethernet" was derived [27].

From this early and comparatively simple concept, Ethernet evolved into the complex networking technology that today underlies most LANs. The coaxial cable was replaced with point-to-point links connected by Ethernet hubs and/or switches to reduce installation costs, increase reliability, and enable point-to-point management and troubleshooting. StarLAN was the first step in the evolution of Ethernet from a coaxial cable bus to a hub-managed, twisted-pair network.

- **Brief History**

In late 1972, Metcalfe and his Xerox PARC colleagues developed the first experimental Ethernet system to interconnect the Xerox Alto, a personal workstation with a graphical user interface. The experimental Ethernet was used to link Altos to one another, and to servers and laser printers. The signal clock for the experimental Ethernet interface was derived from the Alto's system clock, which resulted in a data transmission rate on the experimental Ethernet of 2.94 Mbps.Metcalfe's first experimental network, was called the Alto Aloha Network. In 1973 Metcalfe changed the name to "Ethernet," to make it clear that the system could support any computer--not just Altos--and to point out that his new network mechanisms had evolved well beyond the Aloha system. He chose to base the name on the word "ether" as a way of describing an essential feature of the system: the physical medium (i.e., a cable) carries bits to all stations, much the same way that the old "luminiferous ether" was once thought to propagate electromagnetic waves through space. Thus, Ethernet was born.ò [33].

➢ Ethernet Network Elements

Ethernet LANs consist of network nodes and interconnecting media. The network nodes fall into two major classes:

• **Data terminal equipment (DTE)**ð Devices that are either the source or the destination of data frames. DTEs are typically devices such as PCs, workstations, file servers, or print servers that, as a group, are all often referred to as end stations.

• **Data communication equipment (DCE)**ð Intermediate network devices that receive and forward frames across the network. DCEs may be either standalone devices such as repeaters, network switches, and routers, or communications interface units such as interface cards and modems. The current Ethernet media options include two general types of copper cable: unshielded twisted-pair (UTP) and shielded twisted-pair (STP), plus several types of optical fibre cable.

➢ Ethernet Logical Relationship to the ISO Reference Model

Figure 1.1 shows the IEEE 802.3 logical layers and their relationship to the OSI reference model [30].

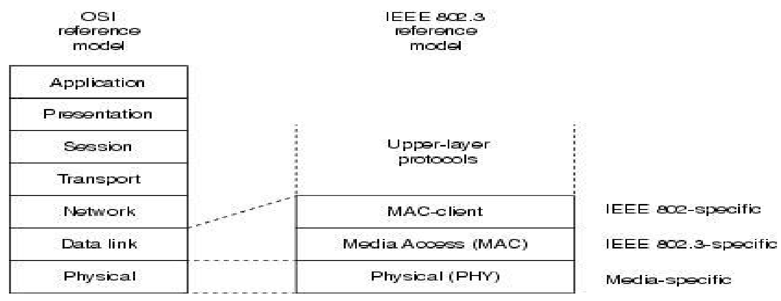


Figure 1.1 Ethernet Logical Relationship to the ISO Reference Model

As with all IEEE 802 protocols, the ISO data link layer is divided into two IEEE 802 sub layers, the Media Access Control (MAC) sub layer and the MAC-client sub layer. The IEEE 802.3 physical layer corresponds to the ISO physical layer.

➢ MAC and Physical Layer Compatibility Requirements for Basic Data Communication

Figure 1.2 shows different compatibility requirements imposed by the MAC and physical levels for basic data communication over an Ethernet link [29].

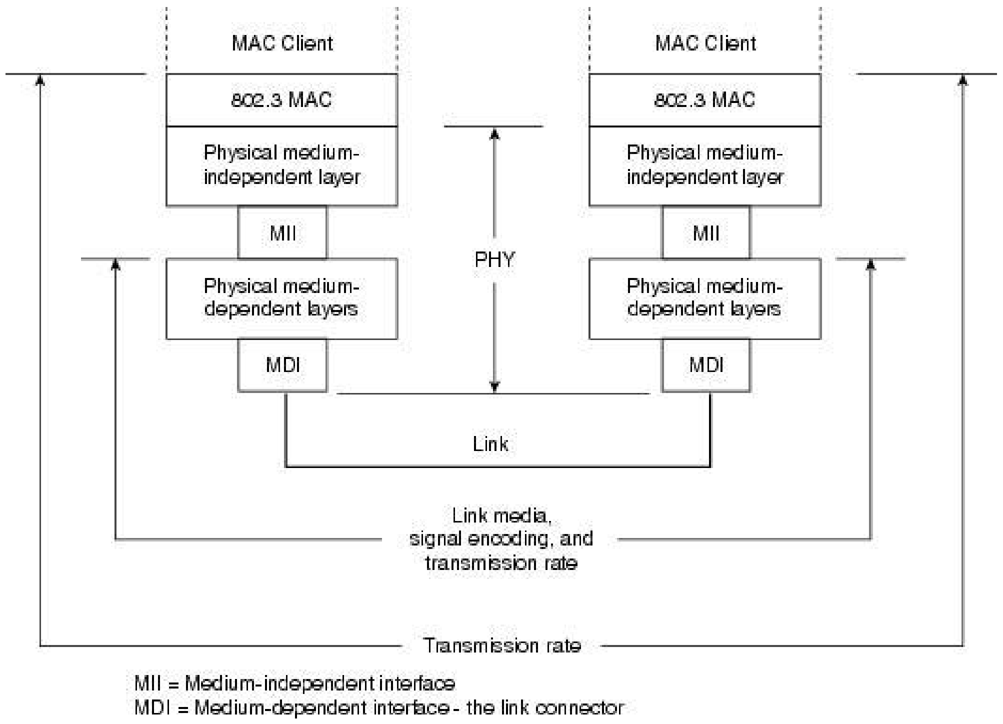

MAC Client
802.3 MAC
Physical medium-independent layer
MII
Physical medium-dependent layers
MDI
MAC Client
802.3 MAC
Physical medium-independent layer
MII
Physical medium-dependent layers
MDI
PHY
Link
Link media, signal encoding, and transmission rate
Transmission rate
MII = Medium-independent interface
MDI = Medium-dependent interface - the link connector

Fig 1.2 MAC and Physical Layer Compatibility Requirements for Basic Data Communication

The MAC layer controls the node's access to the network media and is specific to the individual protocol. All IEEE 802.3 MACs must meet the same basic set of logical requirements, regardless of whether they include one or more of the defined optional protocol extensions. The only requirement for basic communication (communication that does not require optional protocol extensions) between two network nodes is that both MACs must support the same transmission rate. The 802.3 physical layer is specific to the transmission data rate, the signal encoding, and the type of media interconnecting the two nodes. Gigabit Ethernet, for example, is defined to operate over either twisted-pair or optical fiber cable, but each specific type of cable or signal-encoding procedure requires a different physical layer implementation [29].

➢ <u>The Ethernet MAC Sub layer</u>

The MAC sub layer has two primary responsibilities:

Å Data encapsulation, including frame assembly before transmission, and frame parsing/error detection during and after reception

Å Media access control, including initiation of frame transmission and recovery from transmission failure

1.4.2) The Basic Ethernet Frame Format

The IEEE 802.3 standard defines a basic data frame format that is required for all MAC implementations, plus several additional optional formats that are used to extend the protocol's basic capability. The basic data frame format contains the seven fields shown in Figure 1.3 [35].

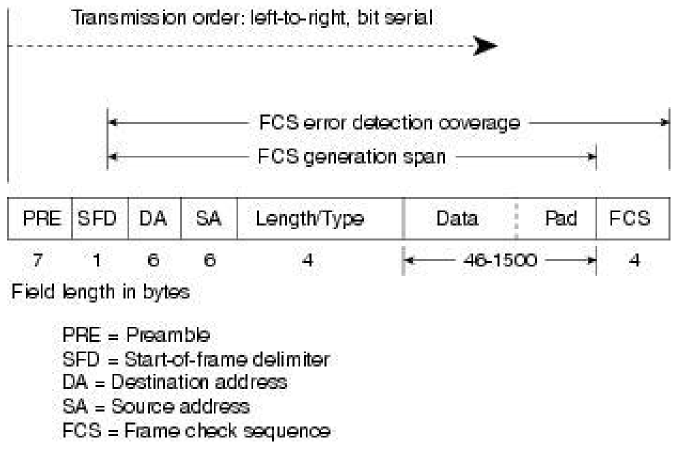


Figure 1.3 The Basic IEEE 802.3 MAC Data Frame Format

Å **Preamble (PRE)**ð Consists of 7 bytes. The PRE is an alternating pattern of ones and zeros that tells receiving stations that a frame is coming, and that provides a means to synchronize the frame-reception portions of receiving physical layers with the incoming bit stream.

Å **Start-of-frame delimiter (SOF)**ð Consists of 1 byte. The SOF is an alternating pattern of ones and zeros, ending with two consecutive 1-bits indicating that the next bit is the left-most bit in the left-most byte of the destination address.

Å **Destination address (DA)**ð Consists of 6 bytes. The DA field identifies which station(s) should receive the frame. The left-most bit in the DA field indicates whether the address is an individual address (indicated by a 0) or a group address (indicated by a 1). The second bit from the left indicates whether the DA is globally administered (indicated by a 0) or locally administered (indicated by a 1). The remaining 46 bits are a uniquely assigned value that identifies a single station, a defined group of stations, or all stations on the network.

Å **Source addresses (SA)**ð Consists of 6 bytes. The SA field identifies the sending station. The SA is always an individual address and the left-most bit in the SA field is always 0.

Å **Length/Type**ð consists of 2 bytes. This field indicates either the number of MAC-client data bytes that are contained in the data field of the frame, or the frame type ID if the frame is assembled using an optional format. If the Length/Type field value is less than or equal to 1500, the number of LLC bytes in the Data field is equal to the Length/Type field value. If the Length/Type field value is greater than 1536, the frame is an optional type frame, and the Length/Type field value identifies the particular type of frame being sent or received.

Å **Data**ð Is a sequence of *n* bytes of any value, where *n* is less than or equal to 1500. If the length of the Data field is less than 46, the Data field must be extended by adding a filler (a pad) sufficient to bring the Data field length to 46 bytes.

Å **Frame check sequence (FCS)**—Consists of 4 bytes. This sequence contains a 32-bit cyclic redundancy check (CRC) value, which is created by the sending MAC and is recalculated by the receiving MAC to check for damaged frames. The FCS is generated over the DA, SA, Length/Type, and Data fields.

➢ <u>Frame Transmission</u>

Whenever an end station MAC receives a transmit-frame request with the accompanying address and data information from the LLC sub layer, the MAC begins the transmission sequence by transferring the LLC information into the MAC frame buffer.

Å The preamble and start-of-frame delimiter are inserted in the PRE and SOF fields.

Å The destination and source addresses are inserted into the address fields.

Å The LLC data bytes are counted, and the number of bytes is inserted into the Length/Type field.

Å The LLC data bytes are inserted into the Data field. If the number of LLC data bytes is less than 46, a pad is added to bring the Data field length up to 46.

Å An FCS value is generated over the DA, SA, Length/Type, and Data fields and is appended to the end of the Data field.

After the frame is assembled, actual frame transmission will depend on whether the MAC is operating in half-duplex or full-duplex mode[40].

The IEEE 802.3 standard currently requires that all Ethernet MACs support half-duplex operation, in which the MAC can be either transmitting or receiving a frame, but it cannot be doing both simultaneously. Full-duplex operation is an optional MAC capability that allows the MAC to transmit and receive frames simultaneously.

A) Half-Duplex Transmission—The CSMA/CD Access Method

The CSMA/CD protocol was originally developed as a means by which two or more stations could share a common media in a switch-less environment when the protocol does not require central arbitration, access tokens, or assigned time slots to indicate when a station will be allowed to transmit. Each Ethernet MAC determines for itself when it will be allowed to send a frame [28].

The CSMA/CD access rules are summarized by the protocol's acronym:

Å **Carrier sense**ð Each station continuously listens for traffic on the medium to determine when gaps between frame transmissions occur.

Å **Multiple access**ð Stations may begin transmitting any time they detect that the network is quiet (there is no traffic).

Å **Collision detect**ð If two or more stations in the same CSMA/CD network (collision domain) begin transmitting at approximately the same time, the bit streams from the transmitting stations will interfere (collide) with each other, and both transmissions will be unreadable. If that happens, each transmitting station must be capable of detecting that a collision has occurred before it has finished sending its frame. Each must stop transmitting as soon as it has detected the collision and then must wait a quasirandom length of time (determined by a back-off algorithm) before attempting to retransmit the frame.

B) Full-Duplex Transmission

Full-duplex operation is an optional MAC capability that allows simultaneous two-way transmission over point-to-point links. Full duplex transmission is functionally much simpler than half-duplex transmission because it involves no media contention, no collisions, no need to schedule retransmissions, and no need for extension bits on the end of short frames. The result is not only more time available for transmission, but also an effective doubling of the link bandwidth because each link can now support [28].

- **Frame Reception**

Frame reception is essentially the same for both half-duplex and full-duplex operations, except that full-duplex MACs must have separate frame buffers and data paths to allow for simultaneous frame transmission and reception [36].

Frame reception is the reverse of frame transmission. The destination address of the received frame is checked and matched against the station's address list (its MAC address, its group addresses, and the broadcast address) to determine whether the frame is destined for that station. If an address match is found, the frame length is checked and the received FCS is compared to the FCS that was generated during frame reception. If the frame length is okay and there is an FCS match, the frame type is determined by the contents of the Length/Type field. The frame is then parsed and forwarded to the appropriate upper layer.

1.4.3) The Ethernet Physical Layers

Because Ethernet devices implement only the bottom two layers of the OSI protocol stack, they are typically implemented as network interface cards (NICs) that plug into the host deviceȏs motherboard. The different NICs are identified by a three-part product name that is based on the physical layer attributes. The naming convention is a concatenation of three terms indicating the transmission rate, the transmission method, and the media type/signal encoding [30].

For example, consider this:

- 10Base-T = 10 Mbps, baseband, over two twisted-pair cables
- 100Base-T2 = 100 Mbps, baseband, over two twisted-pair cables
- 100Base-T4 = 100 Mbps, baseband, over four-twisted pair cables
- 1000Base-LX = 100 Mbps, baseband, long wavelength over optical fiber cable

Currently four data rates are defined for operation over optical fiber and twisted-pair cables for Ethernet networks:10 Mbps ï 10Base-***T Ethernet (802.3)*** ,100 Mbps ï ***Fast Ethernet (802.3u)*** ,1000 Mbps ï ***Gigabit Ethernet (802.3z)*** ,10 Gigabit Ethernet ï IEEE 802.3ae

1.5) Previous Work

1.5.1) Validation Concept

As part of a quality management system, validation confirms that the needs of an *external* customer or user of a product, service, or system are met. Validation is ensuring "you built the right product" and verification is ensuring ñyou built the product rightò.

➢ **<u>Overview of Ethernet Validation</u>**

In Ethernet validation various tests are conducted that ensure validation of MAC layer, physical layer and protocol features of provided Ethernet IP. More emphasis is given to validation of user oriented features, apart from validation sometimes design issues are also dealt if user oriented features are unsatisfactory. The rigorous testing involves transmission and reception of data ranging from small frames to jumbo frames. During the process of debugging and validating common task is the data acquisition and analysis of frames transmitted & received through Ethernet [38]. Logic analyzer is used during validation for capturing the response of system for use in debugging and verifying its operation.

➢ **<u>About Ethernet IP- GMAC-UNIV</u>**

The Ethernet IP developed by Synopsysôs Design Ware Cores is referred as Ethernet MAC Universal in short as GMAC-UNIV .It enables a host to transmit and receive data over Ethernet in compliance with the IEEE 802.3 specification & provides an optimized, configurable, flexible product to meet the needs of various applications and customers, and supports a multitude of industry standard interfaces to the PHY, in addition to the default Gigabit Media Independent Interface (GMII)/ Media Independent Interface (MII) defined in the IEEE 802.3 specifications. The GMAC-UNIV can be used in number of applications such as switches, network interface cards, etc [22].

1.5.2) Test case formulation

Ethernet validation mainly focuses on verifying the different features of Ethernet module-GMAC IP of Set Top Box Chip-STI 7xxx, different test scenario is prepared based on IP specific features provided by the GMAC-IP.Validation tests are divided mainly into three categories based on modes of interfacing between MAC & PHY, IP Features & interoperability with different PHY.

I) Different modes of interfacing between MAC & PHY :

♦ The **Media Independent Interface** (**MII**) is a standard interface used to connect a Fast Ethernet (i.e. 100Mb/s) MAC-block to a PHY. The MII transfers data using 4-bit words (nibble) in each direction it uses 4 data lines, clocked at 25 MHz & 2.5MHz to achieve 100 Mbit/s speed & 10 Mbit/s speed respectively[22].

♦ **Reduced Media Independent Interface (RMII)**

It is a standard that addresses the connection of Ethernet physical layer transceivers (PHY) to Ethernet switches. It reduces the number of signals/pins required for connecting to the PHY from 16 (for an MII-compliant interface) to between 6 and 10. RMII transfers data using 2-bit words in each direction it uses 2 data lines, clocked at 50 MHz & 5MHz to achieve 100 Mbit/s speed & 10 Mbit/s speed respectively [22].

♦ **Gigabit Media Independent Interface (GMII)**

The GMII transfers data using 8-bit words (byte) in each direction, it uses 8 data lines clocked at 125 MHz to achieve 1000 Mbit/s speed

♦ **Turbo Media Independent Interface (TMII)**

TMII interface is same as MII but the clock is doubled .It is also known as over clocked MII.The TMII transfers data using 4-bit words (nibble) in each direction it uses 4 data lines, clocked at 50 MHz & 5 MHz to achieve 200 Mbit/s speed & 20 Mbit/s speed respectively.

♦ **Reverse MII Interface**

The main purpose of Reverse MII interface (RevMII) is to provide peer-to-peer Communication between two devices using MII interface without need of Ethernet cable and medium access mechanism [22].

II) IP Features

IP features can be classified into two categories namely features specified by IEEE Std. and features specified by Vendor.***IEEE Std features*** are flow control ,Address filtering, pause frames, Wake up on LAN,VLAN Tagging etc.***Vendor Specific Features*** are interrupts,PBL value, Frame length etc

III) Interoperability with different PHY

Compatibility check or interoperability of MAC with different PHY is essential to validate the performance of provided IP; it ensures the flexibility of the IP with different PHY

➢ **Validation Tests & Criteria.**

a) Ethernet Initialization Test: It is also known as IP initialization test, In this IP is initialized for different modes of interfacing namely MII, RMII, TMII, GMII, RevMII. This test Enables the ethernet interface, writes to the DMA registers and reads them back and compares with the written values.

Criteria for passing the test: Should be able to get proper clocks

b) IP Feature Test

- **Address Filtering Modes Test:** Different address filtering tests are performed based on provided features. Table below list the different tests & its criteria.

Table 1.1 Address Filtering Modes Test & Criteria for passing the test

Sl NO.	Test	Criteria for passing the test
1	Perfect source address filter test	Frames with source address matching should be filtered.
2	Perfect source inverse address filter test	Frames with source address matching should be passed.
3	Destination inverse address filter test	Frames with destination address matching should be passed
4	Receive all frames	All frames irrespective of whether they pass filtering should be received

- **Jumbo Frame Tests:** This test checks the reception & transmission of Jumbo frames which are large IP frames used in high-performance networks. Anything over the IP MTU (*Maximum Transmission Unit*) which is 1500 bytes on an Ethernet is referred jumbo frame.

Table 1.2 Jumbo Frame Tests & Criteria for passing the test

Sl NO	Test	Criteria for passing the test
1	Jumbo Frame External Loopback	Configure the IP in MII mode, speed100Mbps with Jumbo frame enable and run the external loopback. Transmit the 8100 bytes data and receive the data properly.
2	Jumbo frame transmit and Receive test	Configure the GMAC IP in GMII Mode. Transmit the Jumbo frame from N2X protocol analyzer, receive at GMAC and then again retransmit the same packet from the GMAC IP to N2X.

- **IP Checksum Error Test:**

Table 1.3 IP Checksum Error Test & Criteria for passing the test

Sl NO	Test	Criteria for passing the test
1	IP checksum test	Configure the GMAC IP for Checksum offload. Transmit the IP packets from Smart bits and Check the IPC checksum error at GMAC IP in RDES0(receive descriptor 0).This bit should reset when good frames are transmitted and should set for IP frames with incorrect checksum in the header.

- **Flow Control Test:**

Table 1.4 Flow Control Test & Criteria for passing the test

Sl NO	Test	Criteria for passing the test
1	Hardware Flow control test- transmit flow control by setting the bit	Pause frame should be generated with the desired pause time.
2	Hardware Flow control test - receive flow control by setting the bit	Pause frame transmitted from N2X with specific pause time, and captured at N2X; received data should show the time difference between the received frames equal to the pause time.
3	Flow control Busy/backpressure In Half duplex mode	Set the FCB bit in Flow control & transmit some pattern to occur collision, as a result other side should stop transmitting data.

c) System Test

Table 1.5 System Test & Criteria for passing the test

TEST ID	**TEST**	**CRITERIA FOR PASSING THE TEST**
1	IP in power down mode	In power down mode, it should not receive any packet
2	Wake up BY magic packet with broadcast packet	IP should come out of power down mode and start receiving with broadcast magic packet
3	Wake up by magic packet with Unicast packet	IP should come out of power down mode and start receiving with broadcast magic packet
4	Run GMAC with full throughput.	GMAC should run continuously without BAD frames
5.	Configure the PHY in HDX mode for MII@10/100 both the speeds	Frames transmitted & received should not contain any BAD frames.

d) Debug Test: Loopback Tests

It is done to test the deviceȏs transmit & receive components /modules. Loopback testing typically involves routing of predetermined communications eg. test patterns. Such testing can determine whether the components are working correctly, without actually attempting to transmit across a network or other external communication link. Diagnostic tools must have information regarding a device loopback testing capabilities before they can invoke those capabilities. Device loopback capabilities may indicate the location modules or protocol layers within the device at which the loopback testing can be preformed [41]. Basically four loopback tests are performed namely *GMAC Loopback Test, GMAC PHY Loopback Test, and GMAC External loopback*

Criteria for passing the test: Data transmitted and received should be same with no loss of packets.

➢ Test setup Tools/Equipments

♦ Hardware Tools

a.) STi7xxx MBoard: This is validation purpose Mother Board comprising the Set Top Box chip STi7xxx and related circuitry .GMAC is inside the chip

b.) ST Micro Connect 2: It is used to connect the validation Board with the PC to load the code.

c.) Smart bits 200/2000 with Two ML-7710 Cards support 10/100Mbps speed: It is a protocol analyzer which can receive and transmit Ethernet frames. Applications provided are as follows: Smart application Utility for throughput test & Smart Window Utility for testing transmission, reception, supports many IP feature test like CRC checking by generating error, pause frame generation, Configuration of interframe gap etc.

♦ Software Tools

Validation code is written in C language .Following softwareŵs are loaded in the PC to communicate with the hardware set up.

a.) ST micro toolset 4.2.0 version

b.) STŵs Multisim

c.) Perl Scripting software

♦ Test Set Up

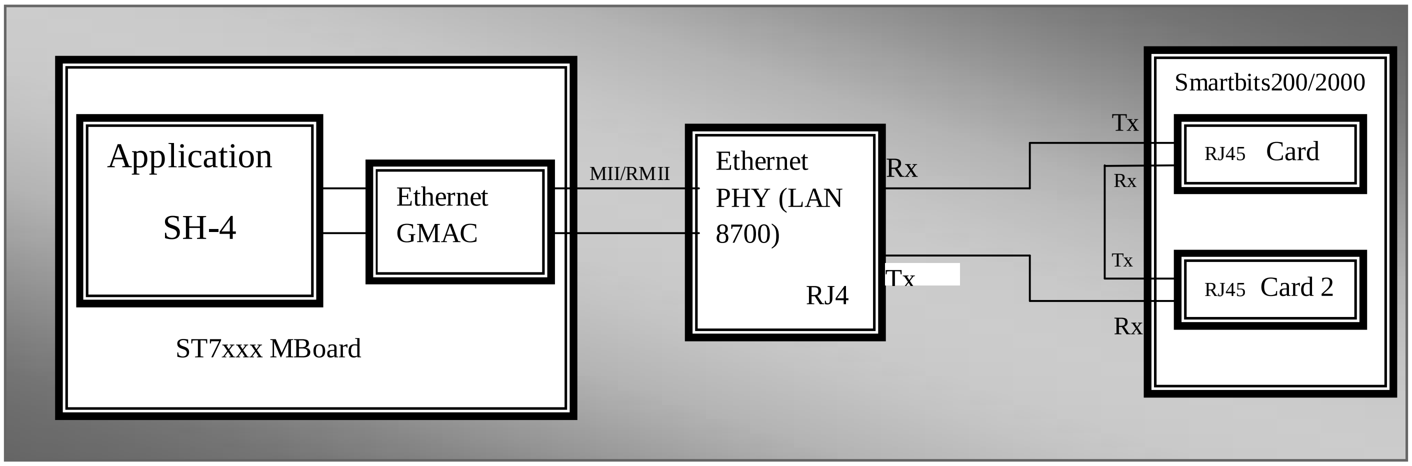


Figure 1.4 STi7xxx MBoard connections with Smart bits

1.5.3) Validation Issues & Need for Automation Set Up for Validation

Debugging digital systems is tougher than ever, increased product requirements, complex software, and innovative hardware technologies make it difficult to meet the market goals .Set top boxes with networking capabilities are highly complex engineers designing and validating the Ethernet MAC layer, physical layer of their devices face constant pressure to reduce the bugs and improve efficiency. During the process of debugging and validating a digital system, a common task is the acquisition of data and its analysis. when corrupted frames are received validating the same becomes a tedious process and for overnight test debugging is even more difficult as logic analyzer used to capture the frames displays the last result only and in such scenario error tracking consumes a lot of time.

- **Need for Automation Set Up for Validation**

Validation process is real complex to conduct wide range of tests quickly and efficiently an automation software is need of the hour it will not only speed up the data acquisition and analysis of the received frame but will also help to track the bugs in minimum possible time.

1.5.9) Conclusion

This chapter highlighted the various aspects which motivated to carry out the research work. It defined the objective of this thesis and discussed the need for automating the Ethernet validation procedures to carry out the validation efficiently.

1.6) Problem Formulation

Testing reliability aspects are important for end using the product of the system, as part of a quality management system; validation confirms that the needs of an *external* customer or user of a product, service, or system are met. Validation is ensuring "you built the right product" and verification is ensuring ñyou built the product rightò.

Ethernet has become a prominent product of an IP-based packet centric network infrastructure .Management of Ethernet networks with carrier ï class reliability is generally a difficult issue for operators and service providers. Engineers designing or validating the Ethernet need to perform a wide range of tests, quickly, reliably and efficiently. In Ethernet validation various tests are conducted that ensure validation of MAC layer, physical layer and protocol features of provided IP. Validation process involves rigorous testing which involves data ranging from small frames to jumbo frames. Testing involves data acquisition and analysis of these frames and when corrupted frames are received validating the same becomes a tedious process and for overnight test debugging is even more difficult as logic analyzer used to capture the frames displays the last result and in such scenario error tracking consumes lot of time.

This project is a step towards the automation of such testing set up where the data acquisition and analysis will go in parallel, indicating not only the type of error but the position where data got corrupted. Automation is done through Lab View software; the developed module not only controls the Logic Analyzer activation but also acquires the data captured by the Analyzer, checks its authenticity and stores the erroneous data for future reference.

1.7) Dissection of Thesis

A variety of background information is presented first; including a cursory survey of previous work in Ethernet Validation done in ST Microelectronics is discussed.

Chapter II describes the different Hardware Modules, its main components, characteristics, & relevant functionalities.

Chapter III describes the different Software❜s used its main functionalities and Characteristics.

In Chapter IV describes the Integration of hardware modules and software❜s for the development of automation set up

Chapter V highlights the Software perspective, the scope of developed software & the logical flow of the program in the form of flowchart

Chapter VI presents the different measurements conducted to test the performance of the developed software. It features the design process used in implementing software model, followed by case studies .It addresses the challenge faced during designing & outlines the limitations.

Finally, in **Chapter VII** the conclusions of this research are presented along with Some recommendations for future work.

CHAPTER - II
Hardware Modules

2.1) Introduction

This chapter describes the different types of hardware used. The main components are the Ethernet Validation Board-STi7xxx-MBoard, ST Micro Connect 2, and Agilentôs Logic Analyzer- 16702B Model & PC. **STi7xxx-MBoard** is a set-top box decoder developed in-house by STMicroelectronics especially for validation of Set Top Box Chip STi7xxx which comprises the Ethernet module-GMAC. **ST Micro Connect 2** is a host-target interface which provides fast access to host services. It can be used with an extensive range of STMicroelectronicsôs cores and enables any of the hosts to connect to a target development board with debug support. **Agilent's Logic Analyzer- 16702B Model** is used during validation for capturing the response of validation system for debugging and verifying its operation. This chapter briefs out the important aspects of each hardware component.

2.2) Set Top Box Chip-STi7xxx

- **General Overview of Set Top Box**

A **set-top box** (STB) or **set-top unit** (STU) is a device that connects to a television and an external source of signal, turning the signal into content which is then displayed on the television screen. The signal source might be an Ethernet cable, a satellite dish, a coaxial cable (see cable television), a telephone line. It enables a television set to become a user interface to the Internet and also enables a television set to receive and decode digital television broadcasts. DTV set-top boxes are sometimes called receivers. A set-top box is necessary to television viewers who wish to use their current analog television sets to receive digital broadcasts. a typical digital set-top box contains one or more microprocessors for running the operating system, possibly Linux or Windows CE, and for parsing the MPEG transport stream. A set-top box also includes RAM, an MPEG decoder chip, and more chips for audio decoding and processing. The contents of a set-top box depend on the DTV standard used.. More sophisticated set-top boxes contain a hard drive for storing recorded television broadcasts, for downloaded software, and for other applications provided by DTV service provider.

- **About-STi7xxx**

The STi7xxx is a new Omega2 set-top box decoder IC targeted at low bit-rate HDTV market for home networking applications. **STi7xxx** integrates a Media Access controller unit, a highly integrated and optimized MAC hardware unit compliant with DOCSIS/Euro DOCSIS 2.0 standard. The MAC supports up to 4 simultaneous downstream TS inputs chosen from the 3 on chip QAM demodulators and the extra TS input. The MAC is supported by a dedicated ST40-300 RISC CPU, plus a dedicated Flexible DMA engine (FDMA).It is supported by the integration of an Ethernet controller with integrated MAC and GMII/MII/RMII interfaces for glue less connection to an external PHY[18,19 &20].

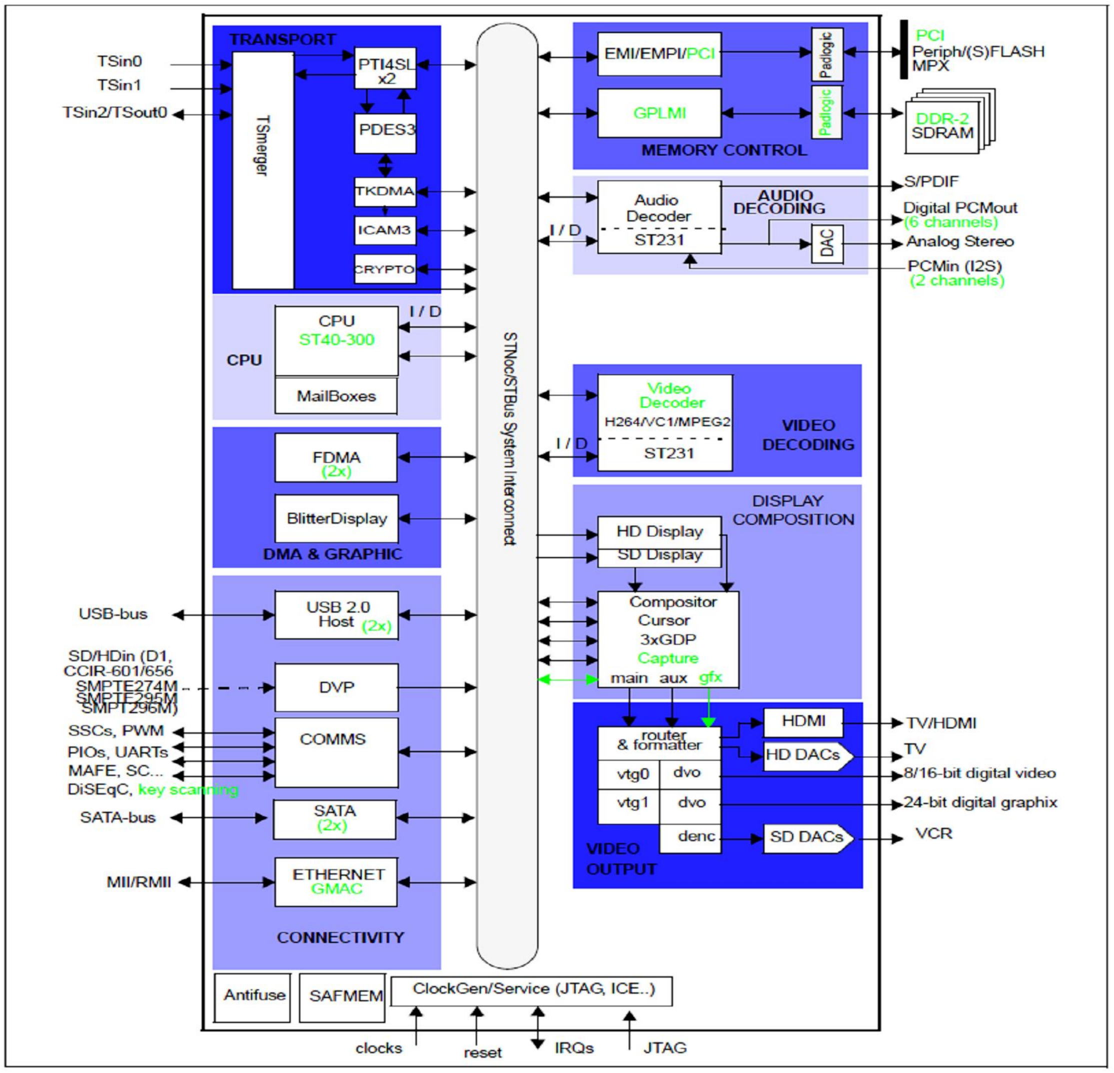


Fig 2.1 Block diagram of Set Top Box Chip STi7xxx

2.2.1) GMAC/ Ethernet

The GMAC-UNIV provides an optimized configurable, flexible product to meet the needs of various applications and customers, and supports a multitude of industry standard interfaces to the PHY. GMAC-UNIV can be used in number of applications such as switches, network interface cards, etc. The GMAC-AHB is designed to interface to the industry standard AMBA High-Performance Bus (AHB) on the application side[22].

- **Overview of GMAC/Ethernet IP**

A system-level block diagram of GMAC is shown in Figure 2.2 .The function of GMAC is to do data transfer over Ethernet network. It can support both full-duplex and half-duplex mode of operation depending on the media and application. It has an optional function to generate pause frame/collision automatically when the host processor is busy for a longer duration of time. It can detect and generate VLAN frames. The in-built power management feature allows the IP to conserve power. Software can enable the checking of IP header Checksum module on-line.

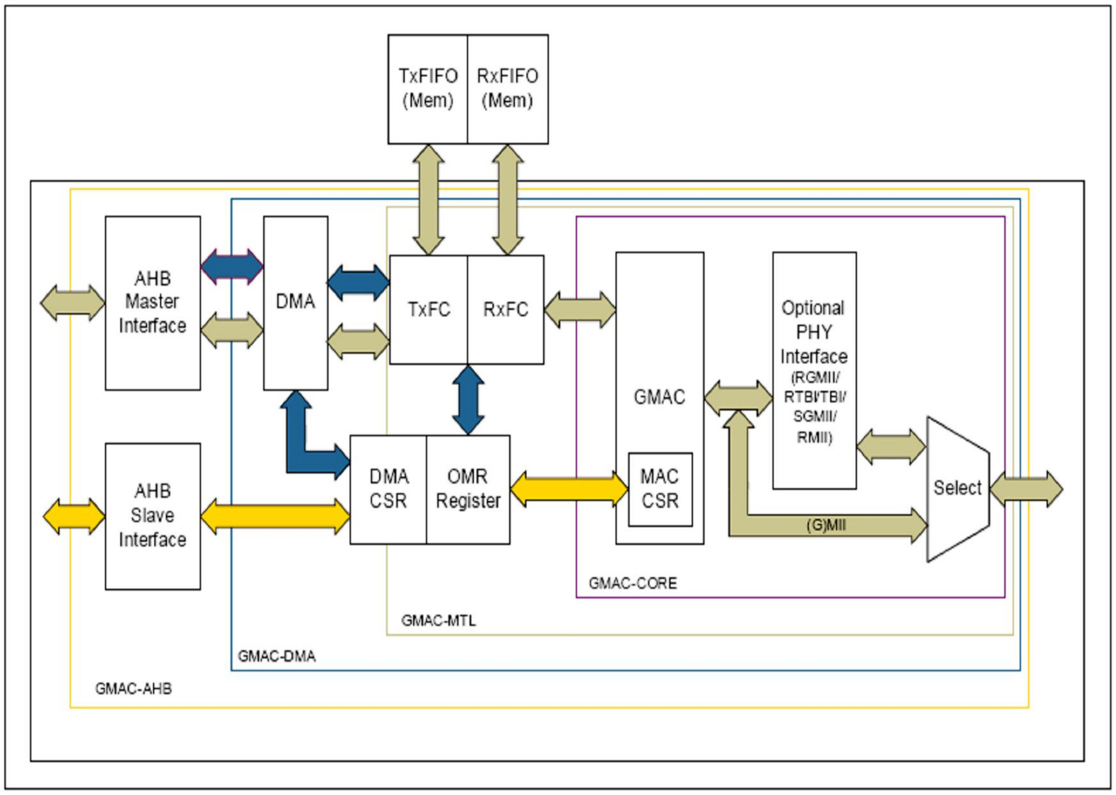


Fig: 2.2 System-level block diagram of GMAC

2.2.2) PHY - Ethernet Physical Layer

The DP83865 is a fully featured Physical Layer transceiver with integrated sub layers to support 10BASE-T, 100BASE-TX and 1000BASE-T Ethernet protocols. The DP83865 is designed for easy implementation of 10/100/1000 Mb/s Ethernet LANs. It interfaces directly to Twisted Pair media via an external transformer. This device interfaces directly to the MAC layer through the IEEE802.3u Standard Media Independent Interface (MII), the IEEE 802.3z Gigabit Media Independent Interface (GMII),or Reduced GMII (RGMII) [23].

➢ **MAC Interface of DP83865**

The DP83865 MAC interface can be configured to one ofthe following different modes:

- MII Mode: Supports 10/100 Mbps MACs.
- GMII Mode: Supports 802.3z compliant 1000 Mbps MACs.
- RGMII Mode: Supports RGMII version 1.3.

Only one mode is used at a time.

The DP83865 supports six different Ethernet protocols: 10BASE-T Full Duplex and Half Duplex, 100BASE-TX Full Duplex and Half Duplex, 1000BASE-T Full Duplex and Half Duplex. There are three ways to select the speed and duplex modes, i.e. manual configuration with external strapping options or through management register write and Auto-Negotiation.

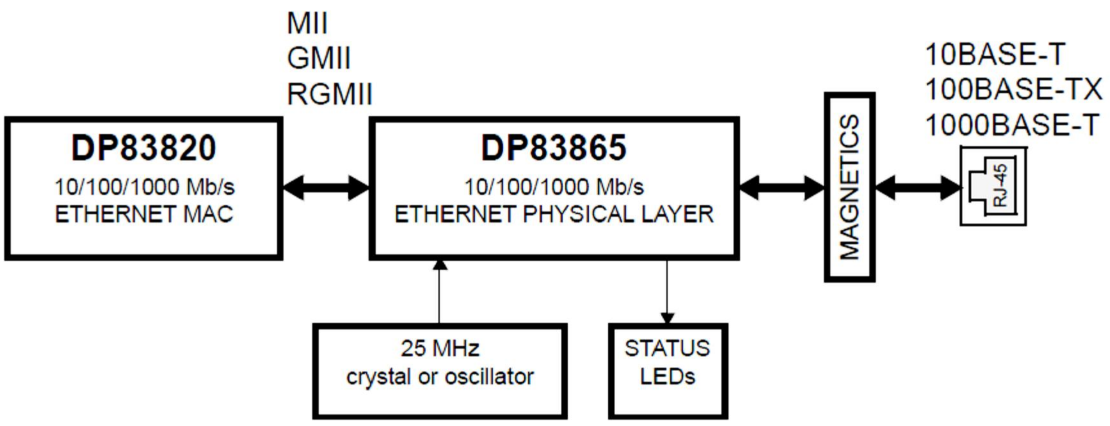


Fig: 2.3 System-level block diagram of PHY-DP83865

2.2.3) Different modes of MAC interface with PHY

An Ethernet interface normally consists of 4 major parts: The MAC (Media Access Controller), the PHY (Physical Interface or transceiver), the magnetics, and the connector. Connectors with integrated magnetics are available. The MAC handles the high level portions of the Ethernet protocol (framing, error detection, when to transmit, etc) and the PHY handles the low level logic (4B/5B encoding/decoding, SERDES (serialization/deserialization), and NRZI encoding/decoding) and analog portions. Different possible interfaces between MAC & PHY are as follows [23]:

a) Media Independent Interface (MII)

The **Media Independent Interface** (**MII**) is a standard interface used to connect a Fast Ethernet (i.e. 100Mb/s) MAC-block to a PHY. The MII may connect to an external transceiver device via a pluggable connector (see photo) or simply connect two chips on the same printed circuit board. Being media independent means that any of several different types of PHY devices can be used without redesigning or replacing the MAC hardware. The equivalents of MII for other speeds are AUI (for 10 megabit Ethernet), GMII (for gigabit Ethernet), and XAUI (for 10 gigabit Ethernet).

➢ **MII bus**

The MII bus (standardized by IEEE 802.3u) is a generic bus that connects **different types of PHYs** to the same network controller (MAC). The network controller may interact with any PHY using the same hardware interface, independent of the media the PHYs are connected to. The MII transfers data using 4-bit words (nibble) in each direction, clocked at 25 MHz to achieve 100 Mbit/s speed. On a PC the CNR connector Type B carries MII bus interface signals. Serial Management Interface (SMI) (see MDIO) is used to transfer management information between MAC and PHY. **MII signals** include: Transmit Data, transmit strobe, transmit clock, transmit error, receive data, receive strobe, receive clock, receive error, collision indication, carrier sense.

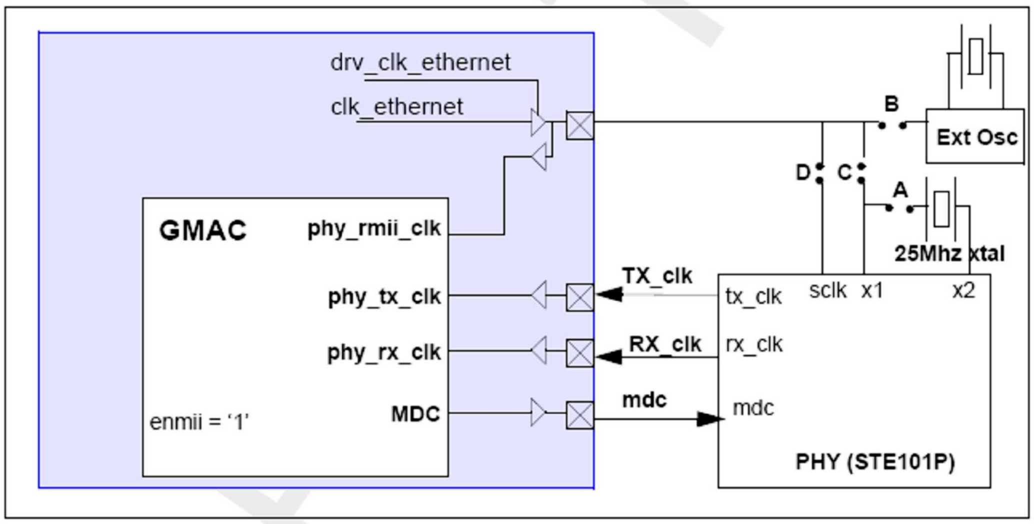


Fig: 2.4 System-level block diagram of MII interface

b) Reduced Media Independent Interface (RMII)

It is a standard that addresses the connection of Ethernet physical layer transceivers (PHY) to Ethernet switches. It reduces the number of signals/pins required for connecting to the PHY from 16 (for an MII-compliant interface) to between 6 and 10. RMII is capable of supporting 10 and 100 Mbit/s; gigabit interfaces need a wider interface. By comparison, the MII interface requires two additional data lines in each direction, RX_DV and CRS are separate rather than multiplexed, a separate TX_CLK and RX_CLK are used instead of a shared reference clock, and a collision signal is added for a total of 7 additional lines. RMII signals include-TXD0 Transmit data bit 0 (MAC to PHY) (transmitted first) ,TXD1 Transmit data bit 1 (MAC to PHY) ,TX_EN When high, clock data on TXD0 and TXD1 to the transmitter (MAC to PHY) ,RXD0 Receive data bit 0 (PHY to MAC) (received first),RXD1 Receive data bit 1 (PHY to MAC) ,CRS_DV, Carrier Sense (CRS)/RX_Data Valid(RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles. (PHY to MAC) ,RX_ER Receive Error (optional on switches) (PHY to MAC) ,REF_CLK Continuous 50 MHz Reference Clock (may be shared among interfaces). Reference clock may be an input on both devices or may be driven from MAC to PHY,MDIO Management data I/O line,MDC Management data clock line (bidirectional but MAC to PHY in practice. MDC and MDIO can in some cases be shared among multiple PHYs and with other devices.

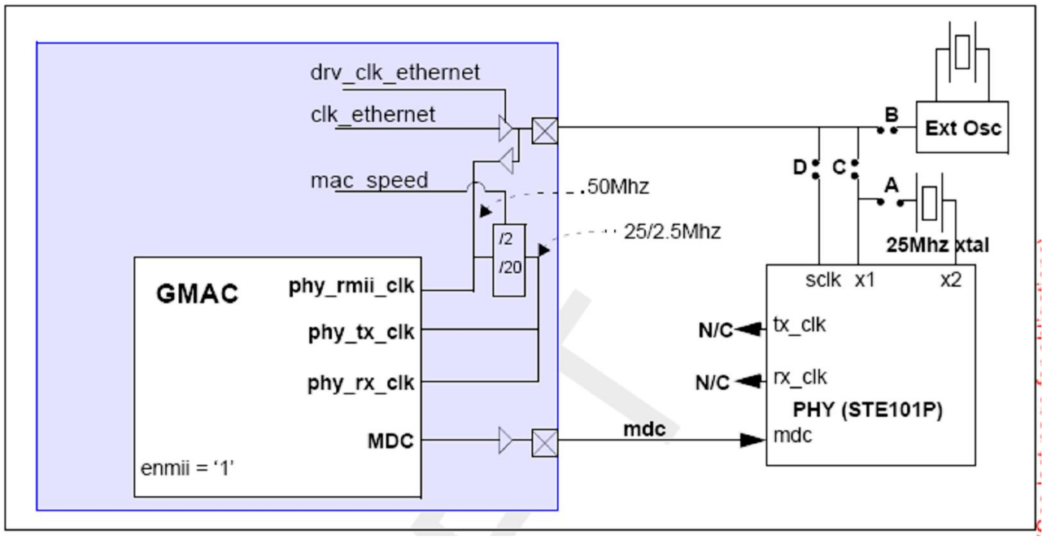


Fig: 2.5 System-level block diagram of RMII interface

c) Reverse MII Interface

The main purpose of Reverse MII interface (RevMII) is to provide peer-to-peer Communication between two devices using MII interface without need of Ethernet cable and medium access mechanism. RevMII also contains all the multiplexers and control logic (both for data and management signals) to connect external Ethernet PHY module seamlessly. The interface is fully IEEE 802.3(u) [1] compliant and allows communicating in Half- and Full-duplex modes with up to 100Mbps maximum data rate. The controlling of the interface is done via standard Serial Management Interface (SMI, defined in IEEE Std. 802.3). The SMI is fully compliant and supports basic control and status register set.

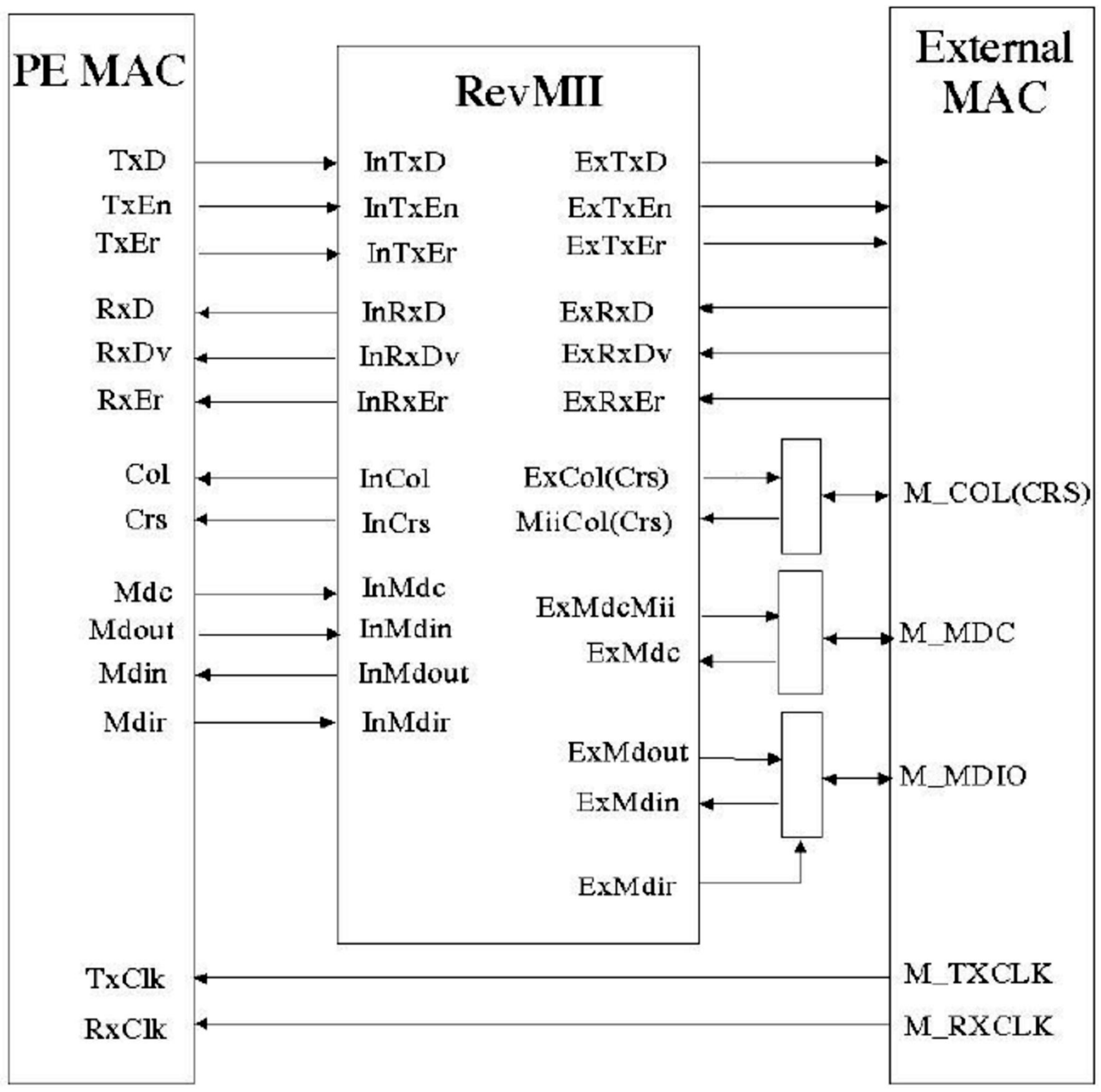


Fig: 2.6 System-level block diagram of Reduced MII interface

2.2.4) PHY Management Control (SMI)

The serial management interface is used to control PHY and obtain its status. At the system level there are two signals-**MDIO** is bi directional open drain & **MDC** is the clock.MDC is a periodic clock provided by station management controller.MDIO signal receives serial data command from the controller SMC & sends serial data status to the SMC.The minimum time between edges of the MDC is 160ns.There is no maximum time between edges. The data on the MDIO line is latched on the rising edge of the MDC [23].

2.3) Interfacing module-ST Microconnect-2

Fig: 2.7 Diagram of ST Micro Connect 2

2.3.1) Description

The ST Micro Connect 2 is the new generation host-target interface for single and multi-core (SoCs), providing fast access to host services. It can be used with an extensive range of STMicroelectronicsȏs cores and enables any of the hosts to connect to a target development board with debug support. The ST Micro Connect 2 provides support for multiple debug-links multiplexed onto a single JTAG line using the TAPMux protocol.Figure 2.8 shows how the ST Micro Connect 2 is used in a system [21].

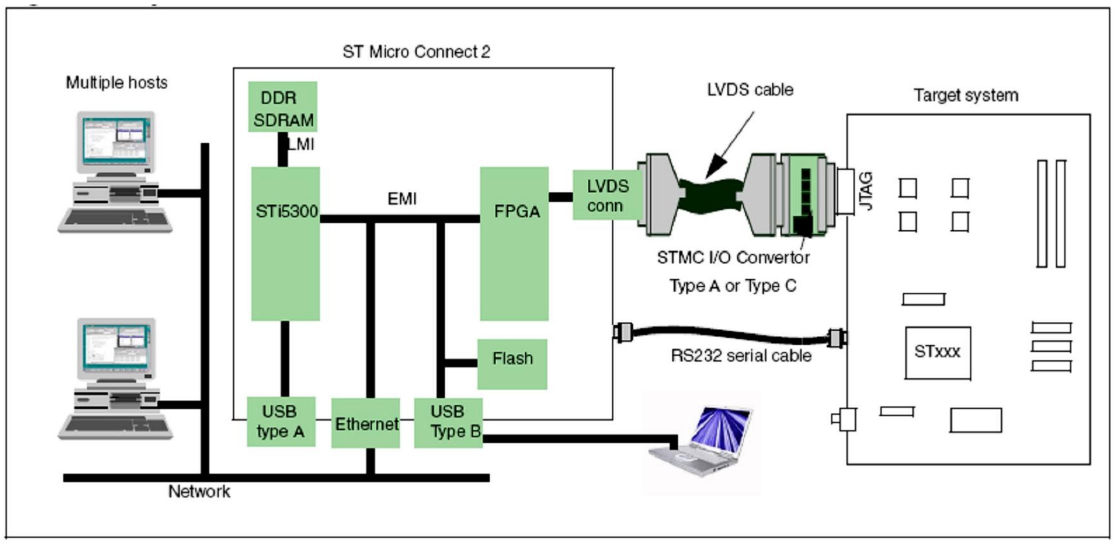
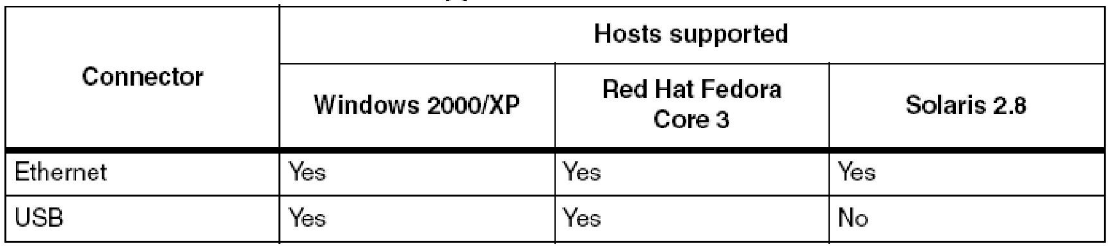


Fig: 2.8 ST Micro Connect 2 interfaced with system

ST Micro Connect 2 is connected to the host by plugging into an **Ethernet** network or host **USB** port, when connected to the Ethernet, ST Micro Connect 2 is a network device that provides a fast, flexible interface between the host system and a target development system. The Advantage of using ST Micro Connect 2 in this mode is that it enables the target system to be remote and to be easily shared among users. This datasheet assumes that the user is reasonably familiar with the concepts and terms involved with TCP/IP networking.
Configuration is made easy by using an LCD interface, which is located on the top surface of the unit. The LCD is operated by two push button switches on the front panel that provide **Select** and **Next** menu navigation

Table 2.1 ST Micro Connect 2 supported host connections



Connector	Hosts supported		
	Windows 2000/XP	Red Hat Fedora Core 3	Solaris 2.8
Ethernet	Yes	Yes	Yes
USB	Yes	Yes	No

.

2.3.2) Functionality Features

The following functionality is provided:

ŏ Status and configuration information display

ŏ User message display

ŏ Network configuration

ŏ Firmware downgrade

ŏ ST Micro Connect 2 reboot

Connection to the development target is through a VHDCI LVDS connector, and a shielded 68-wire SCSI-V cable. A separate LVDS to TTL convertor (the STMC I/O convertor Type A) is provided as connection between the SCSI-V cable and the target interface. This converts LVDS to the standard TTL IDC header, used by some ST boards for JTAG interfacing.
describes the STMC I/O convertor Type A.An RS232 serial port connector is provided for connection to the target. This enables serial data to be relayed between the target and the host through the ST Micro Connect 2. A terminal emulation program can be run on the host to receive serial data and for the user toenter serial data

2.4) Logic Analyzer-Agilent 16702B

2.4.1) Description

Logic analyzer is often used during validation for capturing the response of system for use in debugging and verifying its operation. The Agilent Technologies 16700 Series logic analysis systems provide the power to conquer complex systems by combining state/timing analysis, oscilloscopes, pattern generators, post-processing tool sets, and emulation in one integrated system [25]. Agilent offers a wide variety of state/timing modules for a range of applications, from high-speed glitch capture to multi-channel bus analysis

2.4.2) Functionality Features

The Agilent 16702B mainframe supports a large, 12.1 inch LCD touch screen and redesigned front panel controls for an easy-to-operate, self-contained unit requiring minimal bench space and offering simple portability simultaneously acquire data up to 4 GHz timing and 600 MHz state through the same connection. Timing Zoom is available across all channels, all the time. Remote Programming with Microsoftôs COM/ Microsoft Visual Basic perform pass/fail analysis, stimulus response tests, data acquisition for offline analysis, and system verification and characterization tests.. Rapidly consolidate large amounts of data into displays that provide insight into systemôs behaviour [25].

2.4.3) Working

Triggering the logic analyzer memory system is similar to a circular buffer. When the acquisition is started, the analyzer continuously gathers data samples and stores them in memory. When memory becomes full, it simply wraps around and stores each new sample in the place of the sample that has been in memory the longest. This process will continue until the logic analyzer finds the trigger point. The logic analyzer trigger stops the acquisition at the point you specify and provides a view into the system under test. The primary responsibility of the trigger is to stop the acquisition, but it can also be used to control the selective storage of data. Consider a logic analyzer with the trigger resources you need to quickly set up your measurements. Memory depth & Triggering is the most important aspect of a logic analyzer to consider. On the one hand, powerful triggering resources and algorithms will allow you to focus on potential problem sources without using up valuable memory. On the other hand, to be useful, the trigger must be easy to set up

2.4.4) Main Frame Display

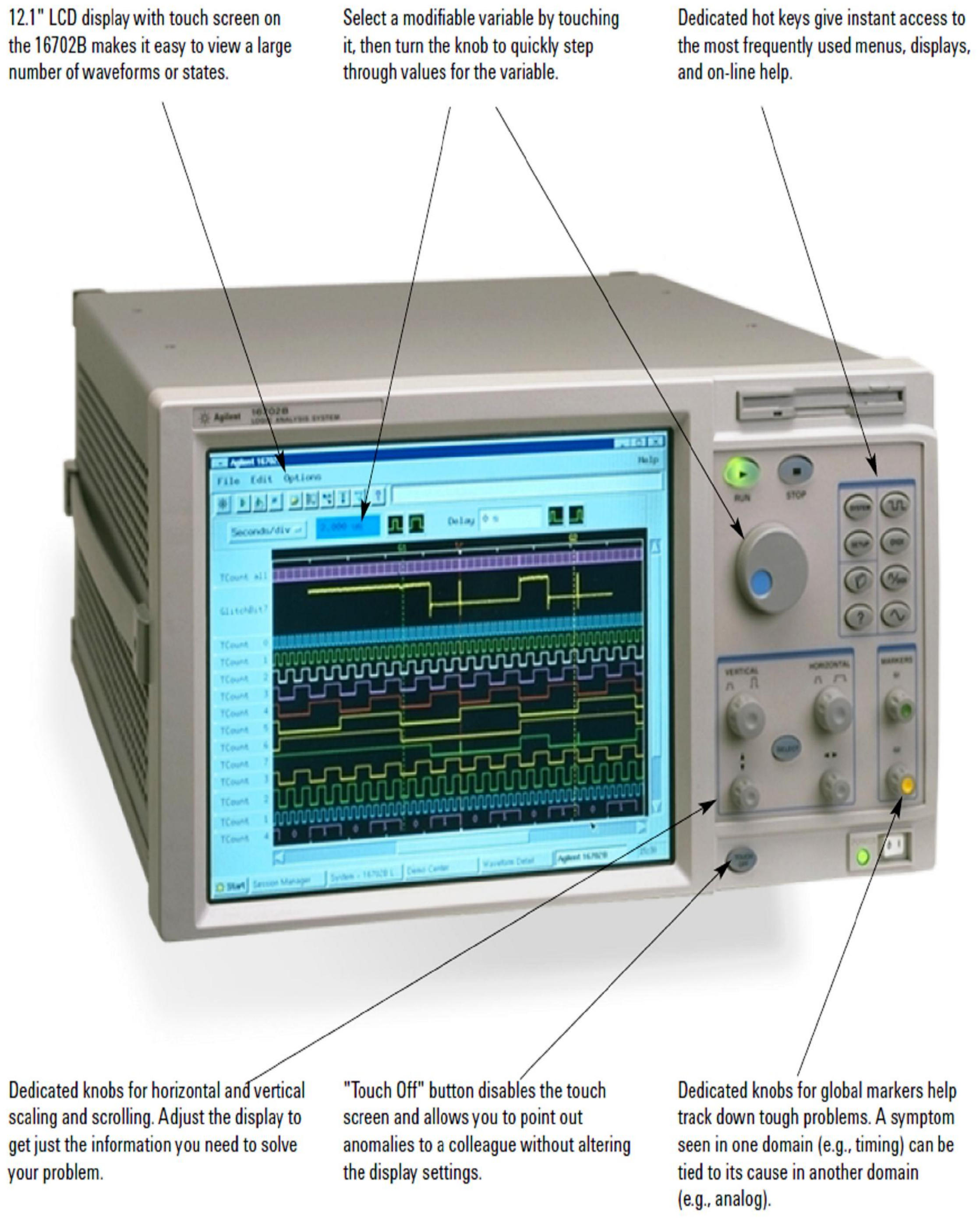

12.1" LCD display with touch screen on the 16702B makes it easy to view a large number of waveforms or states.
Select a modifiable variable by touching it, then turn the knob to quickly step through values for the variable.
Dedicated hot keys give instant access to the most frequently used menus, displays, and on-line help.
Dedicated knobs for horizontal and vertical scaling and scrolling. Adjust the display to get just the information you need to solve your problem.
"Touch Off" button disables the touch screen and allows you to point out anomalies to a colleague without altering the display settings.
Dedicated knobs for global markers help track down tough problems. A symptom seen in one domain (e.g., timing) can be tied to its cause in another domain (e.g., analog).

Fig: 2.9 Main Frame Display of Logic Analyzer

2.4.5) Main Frame Back Panel

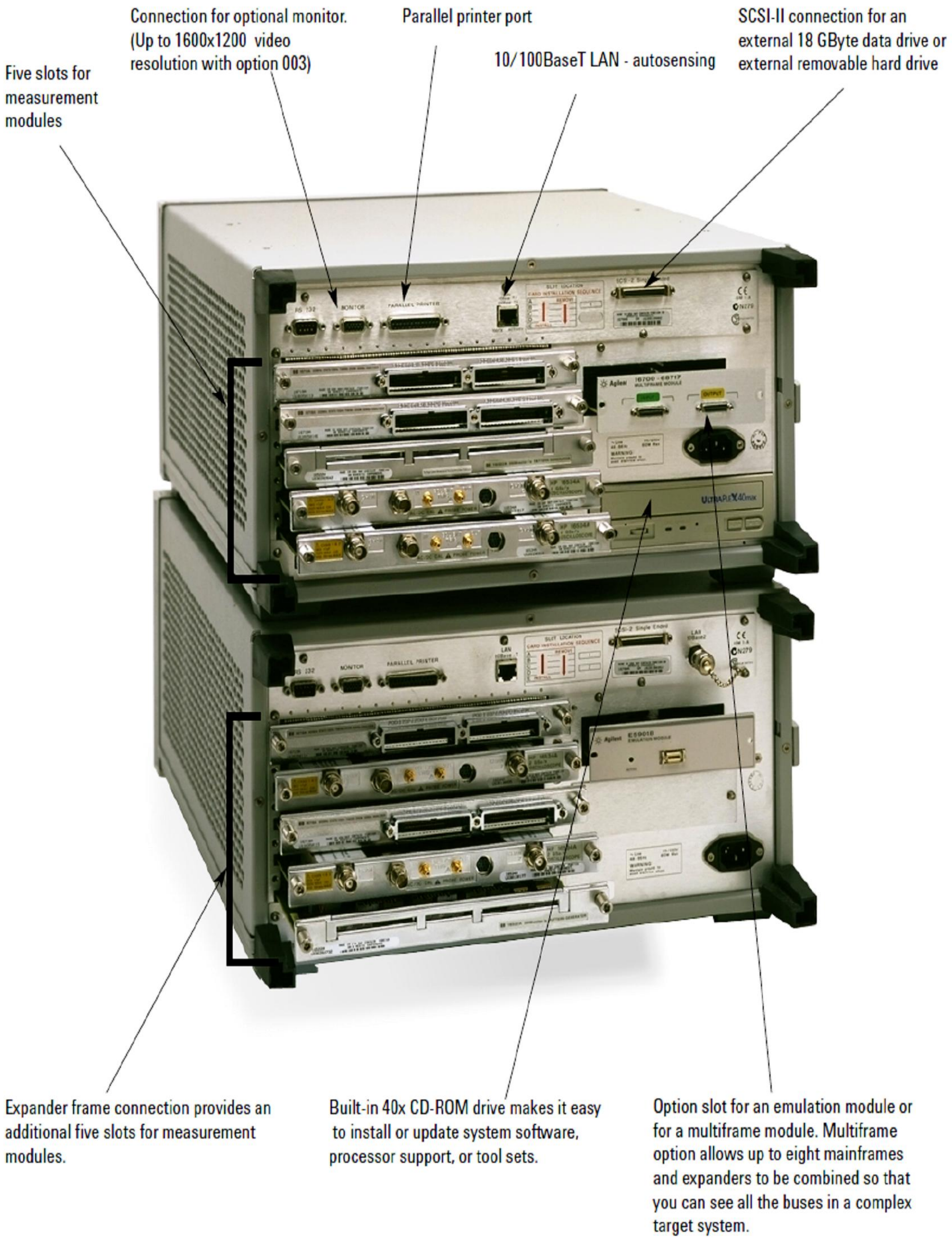

Connection for optional monitor. (Up to 1600x1200 video resolution with option 003)
Parallel printer port
SCSI-II connection for an external 18 GByte data drive or external removable hard drive
10/100BaseT LAN - autosensing
Five slots for measurement modules
Expander frame connection provides an additional five slots for measurement modules.
Built-in 40x CD-ROM drive makes it easy to install or update system software, processor support, or tool sets.
Option slot for an emulation module or for a multiframe module. Multiframe option allows up to eight mainframes and expanders to be combined so that you can see all the buses in a complex target system.

Fig: 2.10 Back Panel of Logic Analyzer

2.4.6) Probing Criteria

Debugging tools perform three important tasks: probing target system, acquiring data, and analyzing data. Data acquisition and analysis tools are only as effective as the physical interface to target system. For probing determine system requirements, the number of signals to be probed , design probing connectors on the target PC board itself, determine Mechanical probing clearance requirements & Signal loading effects [25].

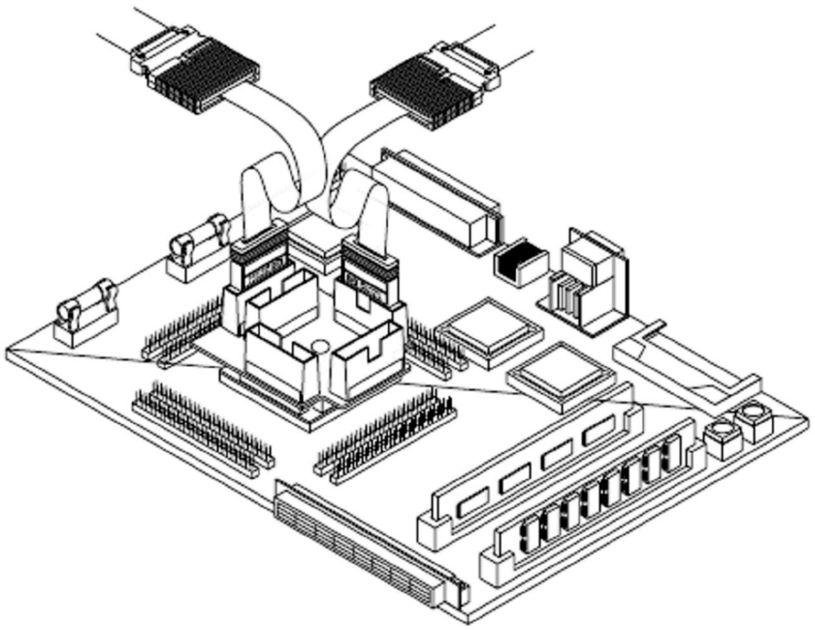


Fig: 2.11 Probes connected to validation board

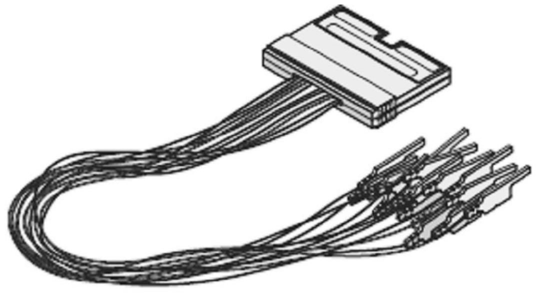
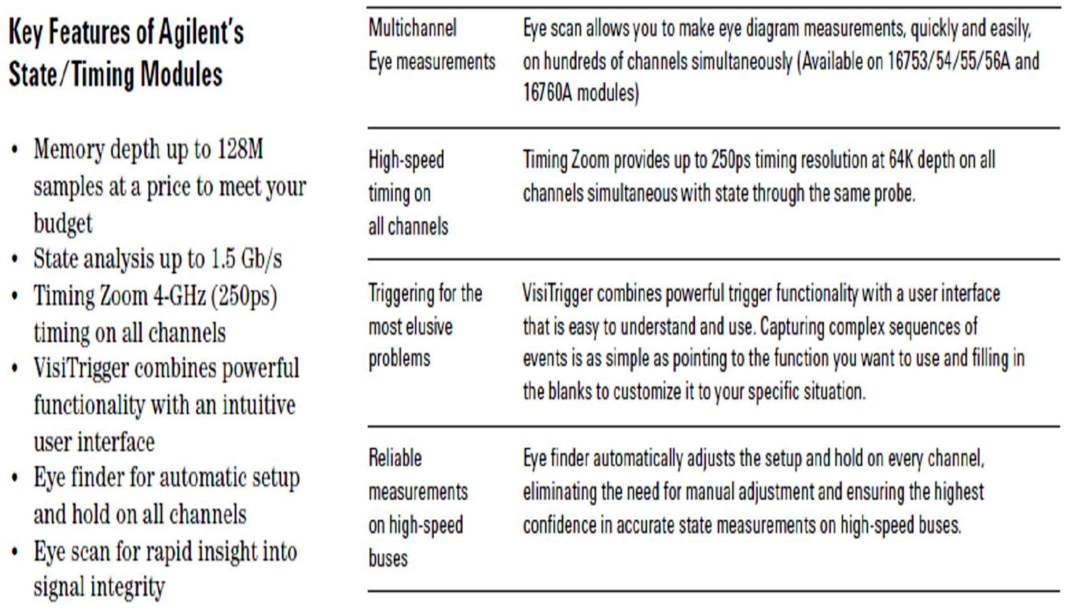


Fig: 2.12 E5382A (single-ended) flying lead probe sets

2.4.7) Data Acquisition and Stimulus: State/Timing Modules

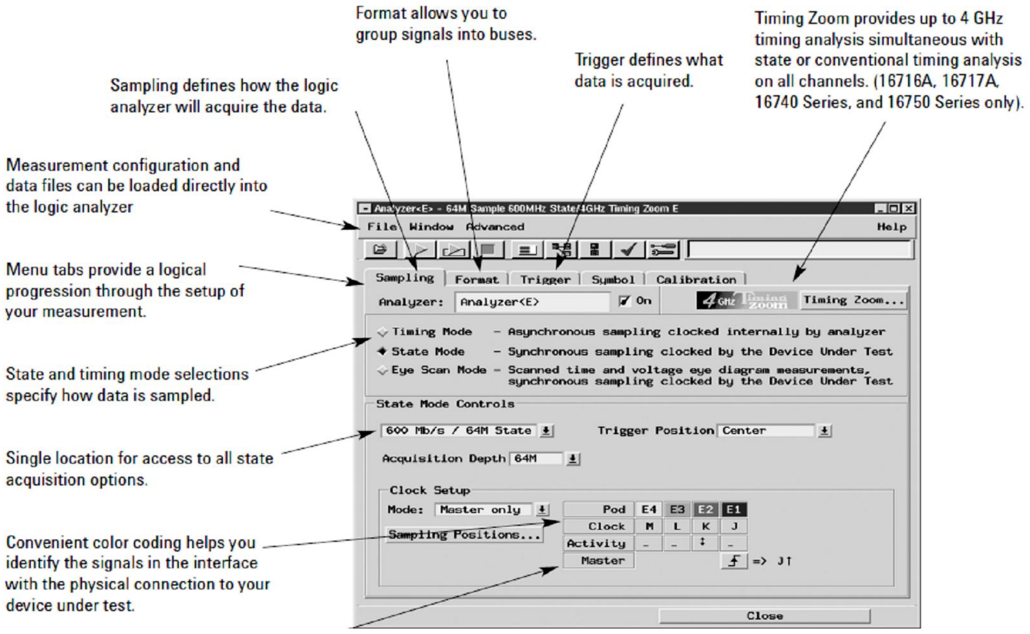


Key Features of Agilent's State/Timing Modules

- Memory depth up to 128M samples at a price to meet your budget
- State analysis up to 1.5 Gb/s
- Timing Zoom 4-GHz (250ps) timing on all channels
- VisiTrigger combines powerful functionality with an intuitive user interface
- Eye finder for automatic setup and hold on all channels
- Eye scan for rapid insight into signal integrity

Multichannel Eye measurements	Eye scan allows you to make eye diagram measurements, quickly and easily, on hundreds of channels simultaneously (Available on 16753/54/55/56A and 16760A modules)
High-speed timing on all channels	Timing Zoom provides up to 250ps timing resolution at 64K depth on all channels simultaneous with state through the same probe.
Triggering for the most elusive problems	VisiTrigger combines powerful trigger functionality with a user interface that is easy to understand and use. Capturing complex sequences of events is as simple as pointing to the function you want to use and filling in the blanks to customize it to your specific situation.
Reliable measurements on high-speed buses	Eye finder automatically adjusts the setup and hold on every channel, eliminating the need for manual adjustment and ensuring the highest confidence in accurate state measurements on high-speed buses.

➢ **Settings of State/Timing Modules**



Format allows you to group signals into buses.

Timing Zoom provides up to 4 GHz timing analysis simultaneous with state or conventional timing analysis on all channels. (16716A, 16717A, 16740 Series, and 16750 Series only).

Trigger defines what data is acquired.

Sampling defines how the logic analyzer will acquire the data.

Measurement configuration and data files can be loaded directly into the logic analyzer

Menu tabs provide a logical progression through the setup of your measurement.

State and timing mode selections specify how data is sampled.

Single location for access to all state acquisition options.

Convenient color coding helps you identify the signals in the interface with the physical connection to your device under test.

Analyzer<E> - 64M Sample 600MHz State/4GHz Timing Zoom E

Analyzer: Analyzer<E> ✓ On

◇ Timing Mode – Asynchronous sampling clocked internally by analyzer
◆ State Mode – Synchronous sampling clocked by the Device Under Test
◇ Eye Scan Mode – Scanned time and voltage eye diagram measurements, synchronous sampling clocked by the Device Under Test

State Mode Controls

600 Mb/s / 64M State
Trigger Position: Center
Acquisition Depth: 64M

Clock Setup
Mode: Master only

Pod	E4	E3	E2	E1
Clock	M	L	K	J
Activity	–	–	↕	–
Master				⨍ => J↑

Fig: 2.13 Settings of State/Timing Modules

2.5) Conclusion

This chapter described the different hardware components of Ethernet validation system. A detailed description of features of Set Top Box chip **STi7xxx, Ethernet IP-GMAC,** & different interfacing modes of MAC & PHY were presented. In addition, it featured the **ST Micro Connect 2** functions and probing and triggering concepts of the logic analyzer.

CHAPTER III
Software

3.1) Introduction

This chapter features the softwareôs employed for developing the Automation software model for the Validation set up. ***Lab View-8.6 versions & Agilent's free connectivity software – Intuilink*** are the main softwareôs which are used for this purpose. Communications between these two softwareôs are facilitated by ***Automation platform -COM /Active X***.

3.2 Application Development Environment- Lab View

3.2.1) Brief History

First introduced in 1986, Lab VIEW was initially intended to enable engineers with little programming experience to rapidly integrate data acquisition and testing equipment in the laboratory [11]. National Instruments sought to revolutionize instrumentation by divorcing the processing software from the acquisition hardware in laboratory instruments. Harnessing the power of the personal computer, the idea of virtual instrumentation enabled engineers
to conduct multiple tests with the same acquisition hardware. Since then, Lab VIEW has evolved into a fully functional programming language, and today, engineers use Lab VIEW to automate manufacturing and inspection in factories, acquire test data for the Xbox video game controller, and control lasers to perform eye surgery [12].

3.2.2) Introduction

Lab VIEW is an open development platform that has gained acceptance in many different application areas and industries. Why? At its core, the Lab VIEW graphical programming language has enabled thousands of scientists and engineers to develop complex measurement and control applications very quickly and easily. However, the language is only one element of the platform that has powered its acceptance over the past 20 years [13&15].

3.2.3) Key elements of the Lab VIEW development platform

I) Intuitive graphical programming language

II) High-level application-specific tools

III) Integrated measurement and control-specific capabilities

IV) Multiple computing targets

I) Intuitive Graphical Programming Language

The Lab VIEW language has a tremendous collection of libraries and structures that have been introduced and improved over the past 20 years.

a)Dataflow

Lab VIEW is a development environment based on a graphical programming language. This approach to developing applications significantly reduces the learning curve because graphical representations are a more natural design notation for engineers and scientists than text-based code. the tools and functions can be accessed through interactive palettes, dialogs, menus, and hundreds of function blocks, known as VIs (virtual instruments). These VIs can be dragged and dropped onto a diagram to define the behavior of required applications. This point-and-click approach significantly reduces the time it takes to get from initial setup to a final solution. The flow of data and the execution of the application is defined through a concept known as dataflow programming. Data is passed from one VI to the next, eventually defining the execution order and functionality of the entire application. Dataflow is comparable in nature to reading a flow chart. Block diagrams consist of functions, which are represented by icons, wires that connect these icons, and structures that control execution logic. Data flows from one function to the next, and the functions and VIs do not execute until all terminals or wire connections have data available for processing.

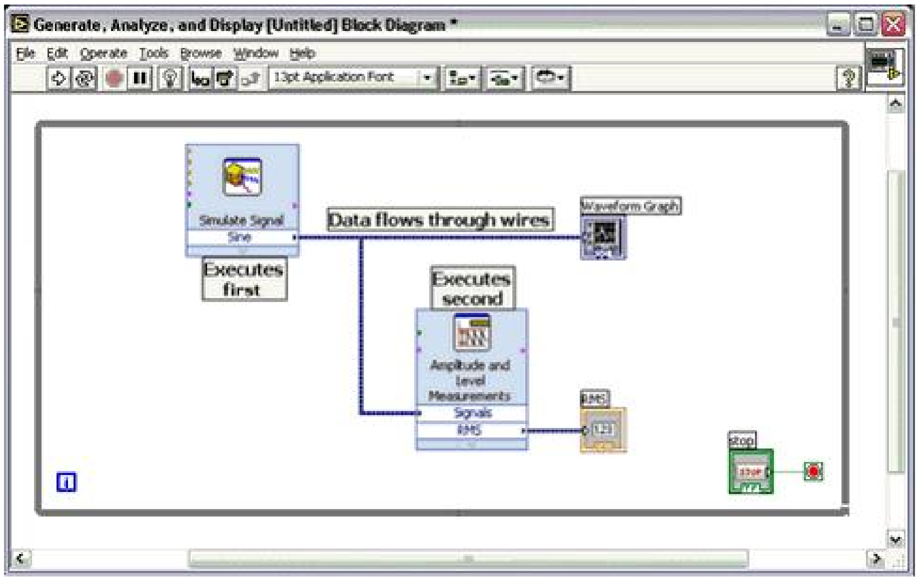

Generate, Analyze, and Display [Untitled] Block Diagram *
File Edit Operate Tools Browse Window Help
13pt Application Font
Simulate Signal
Sine
Executes first
Data flows through wires
Waveform Graph
Executes second
Amplitude and Level Measurements
Signals
RMS
RMS
stop
STOP

Figure 3.1 Lab VIEW Block Diagram Example

b)Modularity

Lab VIEW naturally encourages modularity and reuse of code. Users create VIs, or code modules, with a graphical front panel that displays the inputs and outputs of the functional code as graphical controls and indicators. The graphical controls and indicators (knobs, meters, gauges, graph displays, strip charts, etc) represent data types for the data passing into and out of the functions. Users can easily plug these VIs into other VIs, allowing for modular, hierarchical code that enables users to gradually build up complex systems one component at a time and reuse common operations as subVIs along the way. There is no limit to the number of layers or subVIs used in an application, so the language scales with the complexity required for the application.

c)Multithreading and Parallelism

Lab VIEW eliminates much of the tedious low-level coding required by traditional languages, such as memory management (variable declarations, etc). Lab VIEW also has intuitive graphical structures for common programming structures in text-based languages. For example, while loops and for loops are represented as a box ï the code residing graphically within the box is code executed by the loop iterations. Looking one level deeper into the language, Lab VIEW is designed as a parallel language, which means that the graphical language constructs naturally represent the simple concept of parallel execution. This simple concept, however, can be very difficult to implement in text-based languages because they traditionally execute sequentially (line by line). With Lab VIEW, users can develop parallel-executing applications simply by placing multiple loop structures into their code. A graphical representation of two independent loops, as shown in Figure 3.2, executes independently in parallel as well. This feature is an incredibly simple way to represent a very difficult coding challenge. Parallel execution can be critical in automated test systems, where multiple units under test (UUTs) may be tested, in real-time control systems, where time-critical loops are acquiring data and controlling outputs while data is communicated to the host at the same time, or in embedded applications, where multiple types of inputs must be responded to in a deterministic fashion. When developing parallel-execution applications, the programmer must have tools for setting the priority of different operations. For example, the I/O portion of the program many times is more critical than the user interface. With Lab VIEW, users can configure thread priorities at the OS level using intuitive dialogs and settings.

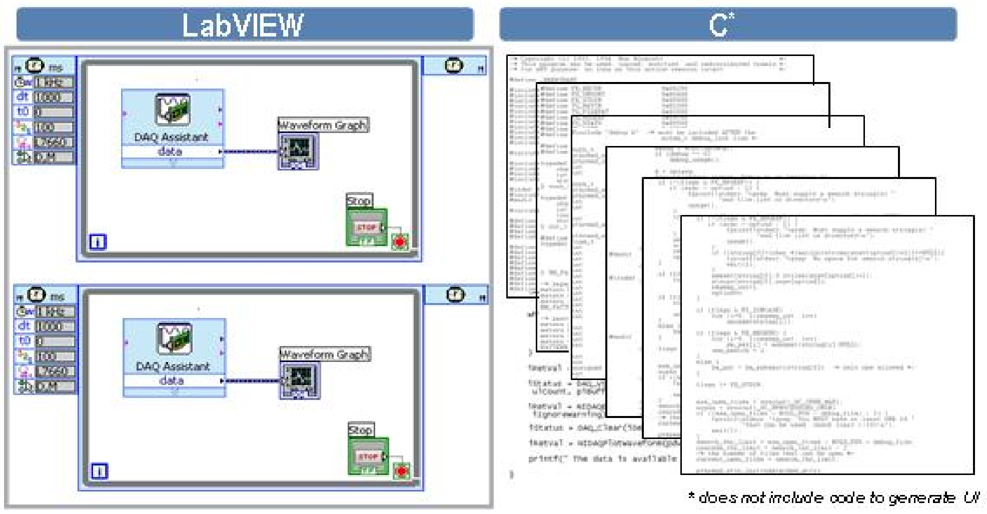


Figure 3.2 The Lab VIEW graphical programming language is designed to represent parallel execution much more intuitively than a sequential text-based language.

d)Interactive Execution and Debugging

The Lab VIEW language is interactive as well, which means users can easily experiment with different functions in the libraries during development, which is particularly important when programming I/O resources. For example, when configuring a data acquisition (DAQ) operation, users can simply select an acquisition function from the built-in DAQ library and run it independently. This operation will actually retrieve data from the DAQ board in the computer, so the user can inspect the data to see if the operation is appropriate for the program. If so, simply drop the VI into the program and continue. If not, try another VI in the library until you find the right one.

Debugging in Lab VIEW is also interactive, featuring all of the common capabilities of traditional programming tools, such as breakpoints, step over/into/out of, and so on. A unique debugging capability of Lab VIEW is the ability to visualize data anywhere within the algorithms you develop without degrading the performance of the algorithm or requiring complex programming. For example, while developing a complex signal processing algorithm in Lab VIEW, easily graph controls can be dropped on the front panel and can be wired to the data path to view the data at that point in the algorithm. Or, can connect a control, such as a knob or slide control, to vary input parameter values of the algorithm. This ability to interactively peek and poke at data and parameter values makes debugging much faster and more intuitive in Lab VIEW.

II) High-Level Application-Specific Development Tools

The Lab VIEW graphical language is an intuitive way for scientists and engineers to develop their measurement and control applications. In addition to being easy to learn and use, the language also delivers the performance needed for advanced applications. The compiled language executes at speeds comparable to traditional compiled text languages.

However, for many applications, there may be higher-level ways to represent a solution (or part of the solution) than using low-level code. Lab VIEW has a growing collection of higher-level tools targeted at solving particular types of structures or constructs much faster. With most of these tools, the user can work at a higher conceptual level to develop a solution, which is then converted into the low-level Lab VIEW code to deliver all of the openness, flexibility, and performance of the compiled Lab VIEW language. These development tools include[16]:

- Control block diagrams ï for designing linear, nonlinear, discrete, and continuous control systems. Users can develop them using traditional control concepts such as transfer function blocks, integrators, differentiators, and feedback loops

- State diagram ï for defining multiple states and transition logic between them using a graphical state diagram representation

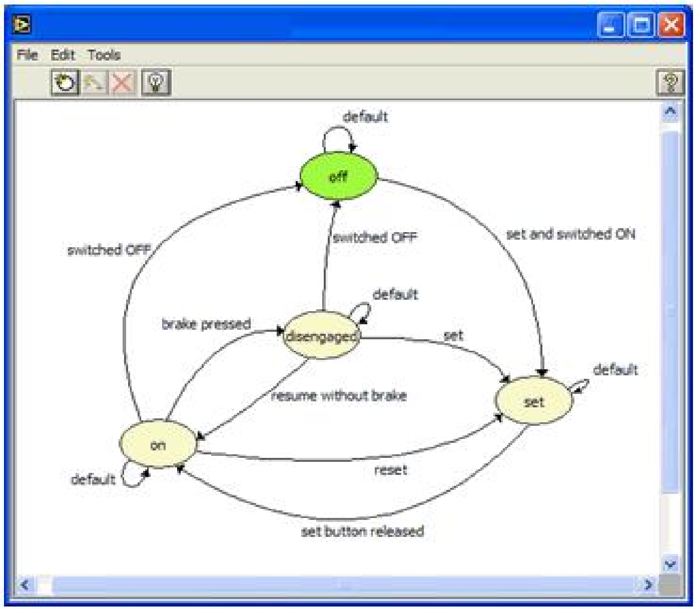


Figure 3.4. State Diagram

- Formula/script nodes ï for implementing complex formulas in text or importing algorithms defined in traditional math tools such as The Math Works MATLAB® or Matrix.
- User interface programming ï for managing very complex user interfaces in graphical code using the event structure in Lab VIEW. The structure receives information about each user event that can be processed in different panes of the event structure.

By combining these high-level concepts to build specific applications with the flexibility of the Lab VIEW language, users get the best of both approaches in one platform.

III) Integrated I/O Capabilities

Lab VIEW is best known as a data acquisition and instrument control tool. These capabilities are built into the language and are pervasive throughout the environment. The language itself naturally manages continuous, looping data acquisition operations, and delivers significant time savings to developers simply because the tool provides functionality throughout with an engineering and scientific perspective in the areas listed below:

➢ **I/O Libraries:**

- Plug-in data acquisition devices
- Modular instruments
- Stand-alone instruments (GPIB, RS232, etc)
- Vision/image acquisition
- Motion control

➢ **Analysis**

- Signal processing
- Sound and vibration
- Order analysis (rotational machinery analysis)
- Spectral measurements and modulation

➢ **Display**

- Graphs, strip charts
- Knobs, meters, gauges
- Pumps, valves, pipes
- Thermometers, tanks

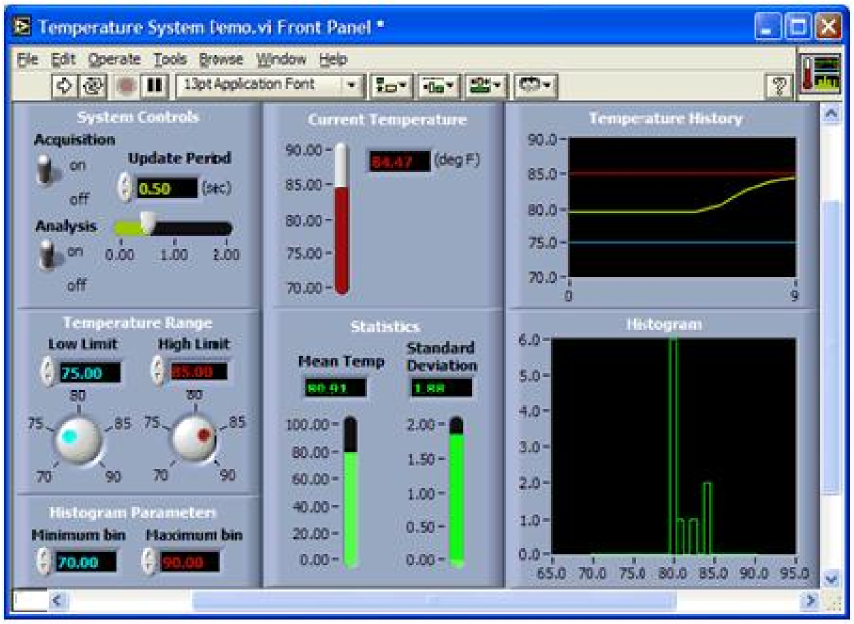


Figure 3.5. Lab VIEW Front Panel

The out-of-the-box integration of all of these different types of engineering-specific controls and libraries cannot be underestimated.

IV) Multiple Computing Targets

Another advantage to the Lab VIEW platform is its open back end that can target a wide variety of computing platforms. The native Lab VIEW compiler runs on all popular desktop OSs, such as Windows, Mac OS X, and Linux. Lab VIEW also runs on industrial real-time platforms, for applications that require determinism or additional reliability. Lab VIEW programs also can be targeted to handheld devices running Windows Mobile, Windows CE, or Palm OS. In addition to the obvious handheld PDA or smart phone devices, these technologies are often found on flat panel displays used in machines or industrial systems. And finally, the Lab VIEW embedded family of products converts Lab VIEW diagrams into C code for execution on 32-bit microprocessors.

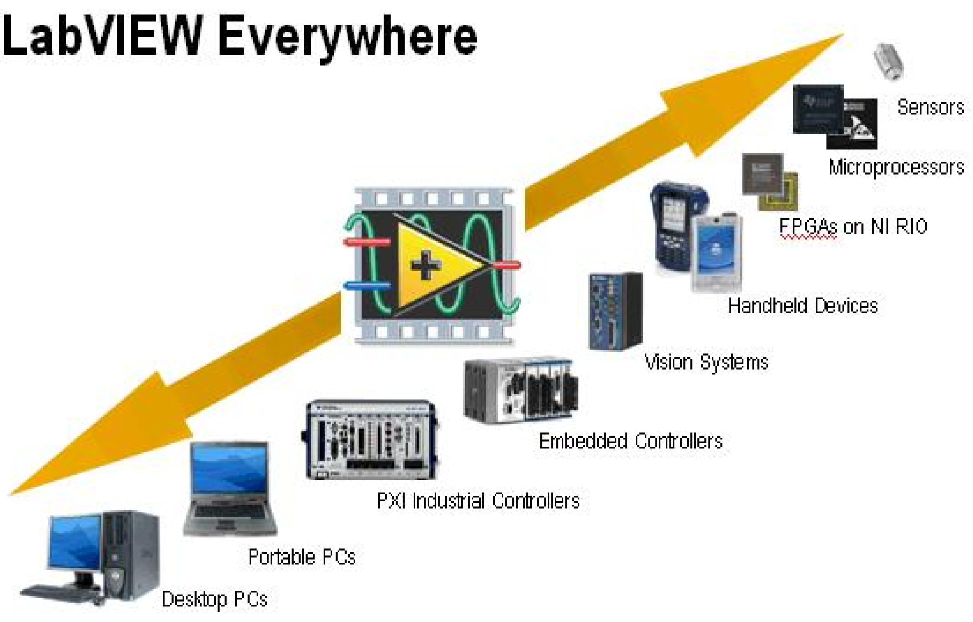


Figure 3.6. Lab VIEW Computing Targets

With this wide array of computing targets, Lab VIEW users can choose the right run-time environment for their application, as well as scale up or down as their requirements change

3.2.4) System Requirements

Table 3.1 Windows Requirements for Lab View Version 8.6

Windows Platform	**Minimum**	**Recommended**
Processor	Pentium III/Celeron 866 MHz or equivalent	Pentium 4/M or equivalent
RAM	256 MB	1 GB
Screen Resolution	1024 x 768 pixels	1024 x 768 pixels
Operating System	Windows Vista/XP/2000	Windows Vista/XP

3.3) Agilent's connectivity software- Intuilink

Agilent Intuilink is free connectivity software that links the test and measurement data to PC applications. IntuiLink is the bridge that links PCs and instruments. It allows retrieving images and data generated by test and measurement instrument from within PC application, without forcing to leave that application or learn new software. It allows to control instruments from PC, providing built-in routines for simple test system automation that streamline repetitive or complex tasks. IntuiLink allow working in familiar PC environment, but use integrated, intrinsic tools to simplify the way of work. IntuiLink functions as part of PC application. Data can be moved from an instrument to a PC with zero programming skills, without forcing to learn new development tools. IntuiLink gives high-level instrument control with no learning curve [25]. Agilent Technologies' IntuiLink software provides drivers and programming samples for interfacing the Agilent 16700 series of logic analyzers with the following PC based programs: ***Microsoft Excel, Microsoft Visual Basic, Microsoft Visual C++, National Instruments Lab VIEW, HP-VEE***

3.3.1) Description

IntuiLink 16700 is a software package containing two software utilities:

- A tool-bar add-in for Microsoft Excel, and
- A Remote Programming Interface (RPI).

In addition, some programming examples are provided for Lab VIEW, HP VEE, Visual Basic and Visual C++. IntuiLink can transfer data from logic analyzer into an Excel spreadsheet of PC, control logic analyzer using custom Visual Basic or Visual C++ programs, or can be used to write programs for Lab VIEW or HP VEE.

3.3.2) System Requirements

PC with a Pentium or higher processor running Microsoft Windows 5/98/NT 4.0 with service pack 3 or higher, ***Microsoft Excel 97 or higher*** to use the Excel Add ï in, ***Microsoft Visual Basic 5.0 or Visual C++ 5.0 or higher*** to use the Programming Samples and ActiveX/COM Automation Server, ***HP VEE 5.0 or Lab VIEW 5.1 or higher*** if it is used in the Programming Samples and ActiveX/COM Automation Server [25].

3.3) Automation Platform-ActiveX/COM

ActiveX is the general name for a set of Microsoft Technologies that allows you to reuse code and link individual programs together to suit your computing needs. Based on COM (Component Object Model) technologies, ActiveX is an extension of a previous technology called OLE (Object Linking and Embedding). Each program does not need to regenerate components, but rather, reuse components to give you the power to combine applications together [9]. Lab VIEW offers support for ActiveX automation as a server as well as support for ActiveX Containers, and ActiveX Events.

3.3.1) ActiveX Automation

ActiveX/COM refers to the process of controlling one program from another via ActiveX. Like networking, one program acts as the client and the other as the server. Lab VIEW supports automation both as the client and the server. Both programs, client and server, exist independent of each other but are able to share information. The client communicates with the ActiveX objects that the server opens to allow the sharing of information. The automation client can access the object's properties and methods. Properties are attributes of an object. Another program can set or retrieve an object's attributes. Similarly, methods are functions that perform an operation on objects. Other applications can invoke methods. An example of an ActiveX property is the program name, height or width. An example of an ActiveX method is the save or print method.

3.3.2) ActiveX Controls and Containers

The most common usage of ActiveX is via ActiveX controls, which are embeddable components that exist inside ActiveX containers. Any program that can function as an ActiveX container allows you to drop ActiveX controls into the program as a container. From these containers, the ActiveX controls have their own functionality and properties. Lab VIEW is an ActiveX container and can house ActiveX controls. Again, properties and methods manipulate the embedded control.

3.3.3) ActiveX Events

ActiveX is an event-driven technology. Programs report when certain events occur so that other programs, or you, can react accordingly. Lab VIEW supports the processing of ActiveX events via both automation and embedding of ActiveX controls.

3.3.4) ActiveX Automation with Lab VIEW

Lab VIEW as an ActiveX server or ActiveX client can interface with other programs from the Lab VIEW programming interface. In this case, Lab VIEW acts as the automation client and requests information of the automation server, or other program. Likewise, other ActiveX automation clients can interface with the Lab VIEW ActiveX automation server. Common programs used are Microsoft Visual Basic and Microsoft Visual C++.

- ***Lab VIEW as an Automation Client***

Lab VIEW provides functions in its API that allow Lab VIEW to act as an automation client with any automation server. The diagram below shows Lab Viewês programming flow, and gives the associated functions with each block.

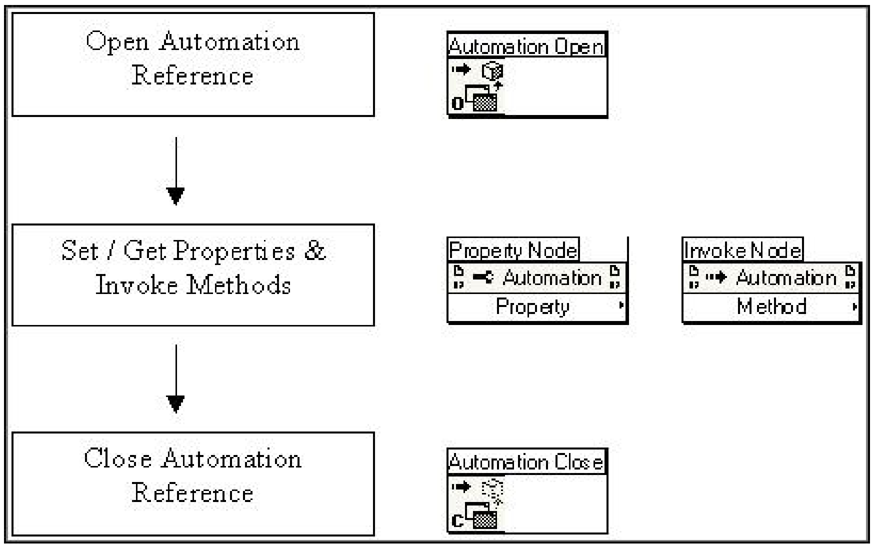


Figure 3.7 Lab View's programming flow

In general, information about a programôs ActiveX automation server can be obtained from the programôs documentation or by browsing the programôs type library. Often, Lab VIEW is used as an automation client for Microsoft Office programs and their object models are available online from Microsoft.

- ***Lab VIEW as an Automation Server***

Other programs can interface with the Lab VIEW automation server using ActiveX automation. Using an automation client, it is possible to programmatically launch Lab VIEW, open and run VIs, and pass their data back to the calling program. The automation client interfaces with the Lab VIEW type library which is located in the \lab view\resource directory. By browsing this type library, information about the classes that Lab VIEW exports is available. In general, Lab VIEW exports a creatable class, Application, and a dispatch class, Virtual Instrument. Additionally, Lab VIEW executables can be ActiveX automation servers. Automation clients such as Lab VIEW or another Lab VIEW executable can access ActiveX automation servers.

- ***Lab VIEW as an ActiveX Container***

In general, Lab VIEW can embed any ActiveX control. Using the controlôs properties and methods, Lab VIEW can programmatically interact with the control. The flow chart for using an ActiveX container is shown below:

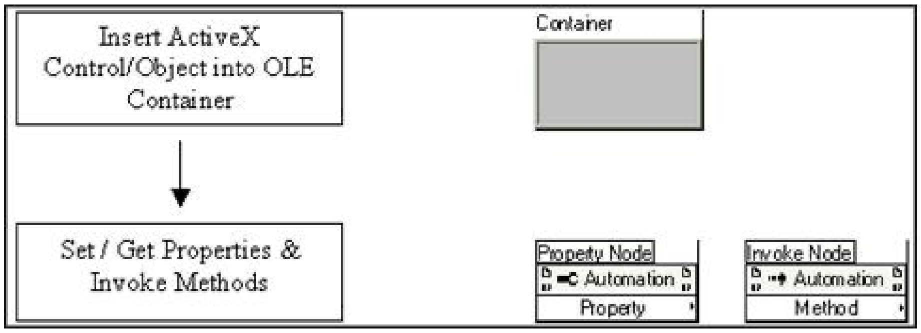

Insert ActiveX Control/Object into OLE Container
Set / Get Properties & Invoke Methods
Container
Property Node
Automation
Property
Invoke Node
Automation
Method

Figure 3.8 The flow chart for using an ActiveX container

the properties and methods available can be explored using the property nodes and invoke nodes by wiring a reference to the nodes.

➢ ***ActiveX Events and Lab VIEW***

Lab VIEW supports ActiveX events via automation and ActiveX controls embedded in containers. ActiveX events allow programmers to receive notification of a specific occurrence and then act accordingly. Commonly, programs wait until an event has fired and then after the event fires, continue with program execution dependent on what event fired. The flow chart for setting an ActiveX event sequence in Lab VIEW is shown below with the corresponding Lab VIEW functions.

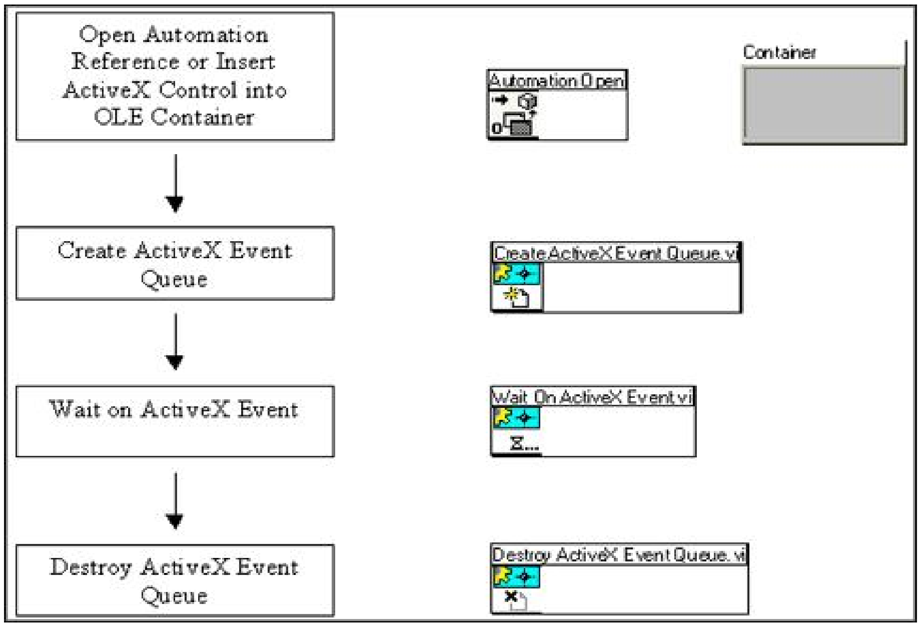

Open Automation Reference or Insert ActiveX Control into OLE Container
Automation Open
Container
Create ActiveX Event Queue
Create ActiveX Event Queue.vi
Wait on ActiveX Event
Wait On ActiveX Event.vi
Destroy ActiveX Event Queue
Destroy ActiveX Event Queue.vi

Figure 3.9 The flow chart for setting an ActiveX event sequence in Lab VIEW

3.3.5) Information about OLE Variants and Lab VIEW

The Variant data type in ActiveX/OLE is used to pass data between programs. A variant can be *anything*; it can represent any data type. It is necessary to use this data type when passing data between programs because common data types in each program can be represented differently [6]. For example, an array in LabVIEW is represented differently than an array in Visual Basic. Thus, by passing data as variant we can pass that information from one ActiveX component to another. Convert the variant data sent from one program to a useable data type when the other program receives it. The To G Data function allows you to convert a variant to a LabVIEW data type

3.5) Conclusion

This chapter described how the plethora of available drivers, combined with the intuitive graphical environment of Lab VIEW, enable novice programmers to develop complex applications with a very short learning curve. It featured its advanced programming concepts such as parallel processing, dataflow programming, real-time execution & FIFO queues. In addition to Lab view it featured the automation platform ActiveX and the connectivity software Intuilink which facilitates the communication between the developed model & the Logic Analyzer.

CHAPTER IV

Integration of Hardware modules & software

4.1) Introduction

This chapter describes the automation set up and its working. It includes the relevant characteristics of the hardware and software employed. Figure 4.1 illustrates a schematic representation of the physical set up, emphasizing the connection of main components.

4.2) Validation set up

The test set up includes the PC connected to Micro connect through LAN ,Micro connect is connected to the validation board through JTAG ,it loads the validation code to the testing board STi7xxxMBoard , Agilentôs logical Analyzer is connected to the testing board through probes. Probing is done between MAC and PHY to capture the data flowing from MAC to PHY or vice versa.

4.3) Validation Procedure

Ethernet validation mainly focuses on verifying the different features of Ethernet IP - GMAC of Set Top Box chip STi7xxx, different test scenario is prepared based on IP specific features provided by the GMAC-IP. Validation coding is done in óCô language. It is done to tests deviceôs various features like transmit & receive modules, filtering options, interrupts etc. Typically testing involves routing of predetermined communications eg. test patterns to determine whether the components are working correctly & transmitting the signals across a network or other external communication link properly or not. Diagnostic tools must have information regarding testing capabilities before they can invoke those capabilities. Mostly in all tests specific patterns are transmitted & its authenticity is checked at the receiving end (protocol analyzer) and similarly patterns transmitted from protocol analyzer are checked for authenticity at the receiving end (PC). This procedure is repeated for all modes of interfacing.

4.4) Steps for Loading Validation Code

1) Power up PC, Micro connect, STi7xxx MBoard, Protocol Analyzer.
2) Load the path of validation code in command window.
3) Input the IP Address of the Target Micro connect & itôs type.
4) Load the Main program.
5) Run the Main program.
6) Once the Target host connection established instruction for transmit test or receive test is given.
7) Validation code is fired on STi7xxx M Board through Micro connect.
8) For transmit test, pattern received in protocol analyzer is compared with the one sent to validation board through PC.
9) For receiving test packets received in PC is compared with packets sent by protocol analyzer.
10) Logic Analyzer used for capturing data is probed between MAC and PHY.It stores the data once triggering occurs, which is set based on some condition.

4.5) Steps to form Configuration file of the logic analyzer for Test Set Up:

1) Choose the Analyzer & slot.
2) Set the acquisition Depth.
3) Choose the mode of analysis -timing mode/state mode.
4) Set the triggering condition & format the data to be acquired from the test set up, give appropriate label and specify the probe number to which it is connected.
5) Finally save the configuration file in a directory.

4.6) Interfacing of Lab VIEW & Logic Analyzer

To automate the Logic Analyzer PC installed with Lab View 8.6 & Agilentôs connectivity software Intuilink is connected to the Logic Analyzer through LAN which is connected to the validation set up through pods. PC, Logic Analyzer & Microconnnect are under same network. The automation software developed through Lab View controls the Logic Analyzer only, loading of validation code & data to be transmitted or received from the validation set up is independent of the automation software.

4.7) Automation Set Up

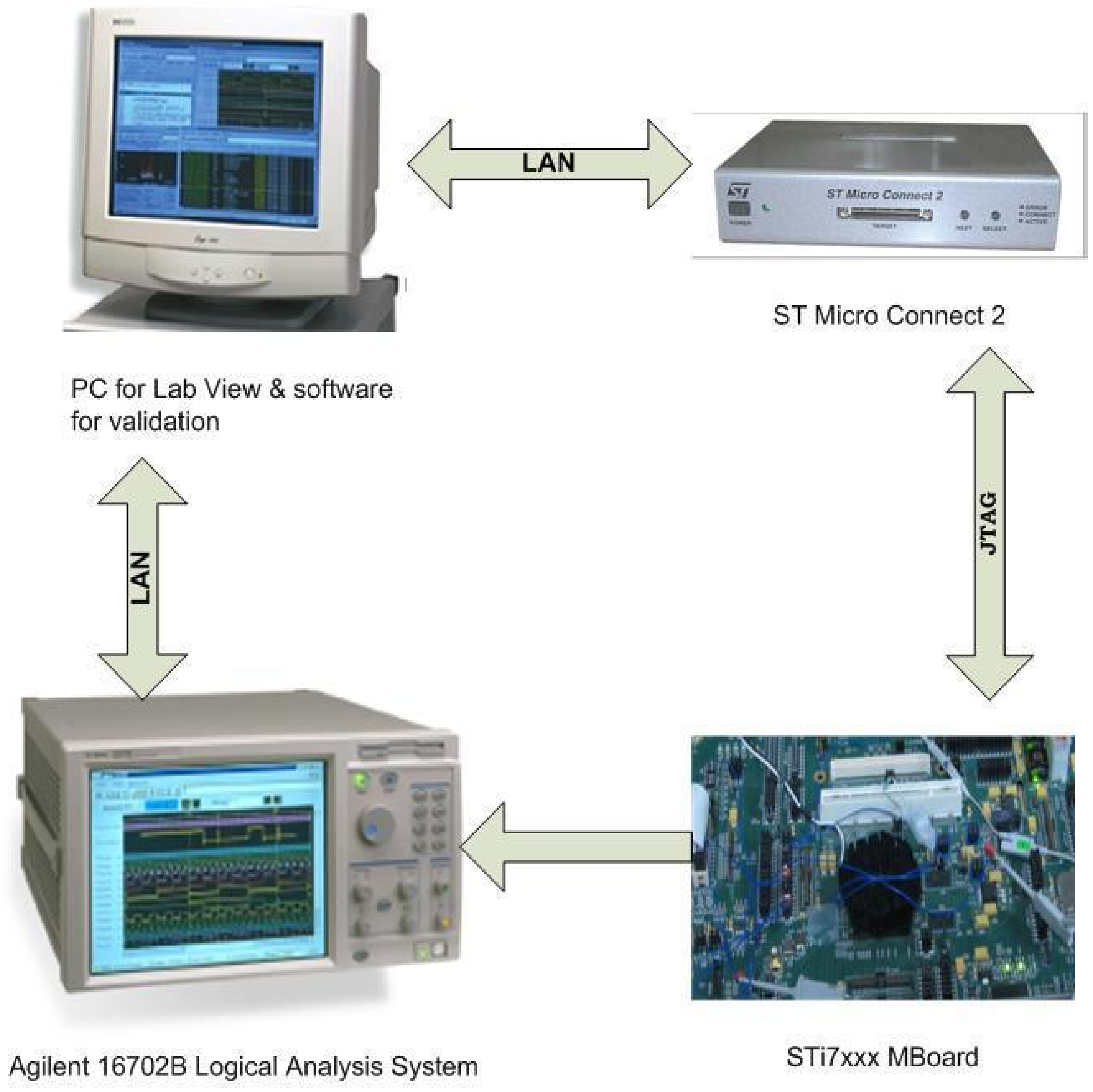

LAN
ST Micro Connect 2
PC for Lab View & software for validation
LAN
JTAG
Agilent 16702B Logical Analysis System
STi7xxx MBoard

Figure 4.1 Automation Set Up

4.8) Working of Automation software integrated with Validation Set Up

Automation software developed not only controls the Logic Analyzer activation but also acquires the data captured by the Logic Analyzer, checks its authenticity and stores the erroneous data for future reference. Steps to start-up the software model are as follows:

1) Power up PC, Micro connect, STi7xxx MBoard ,Protocol Analyzer & Logic Analyzer
2) Load the basic information like Logic Analyzerôs IP address, configuration file of LA, No of samples to be acquired, Path of expected packet etc in the user interface of developed module in Lab View
3) Load the Validation code through Micro Connect on STi7xxx M Board
4) Select the stop button to run the module in either single acquisition mode or continuous acquisition mode.
5) Start the acquisition using Run command.
6) Lab view through LAN will load the Logic Analyzerôs configuration file.
7) Based on triggering condition Logic Analyzer will capture the data and once capturing is over data will be transferred to PC within the Lab view program.
8) Analysis for authentification of captured data & tracing of error will be done in comparison module of application.
9) Last Packet acquired & comparison result relevant to last packet will be in display.
10) Erroraneous packet if any will be stored in PC.

4.9) Conclusion

This chapter described the validation set up and its working. It featured the steps followed to connect the automation software to the validation set up comprising the hardware components. It highlighted the overall working of hardware interfaced with the developed software module.

CHAPTER: V

Program Development In Lab VIEW

5.1) Introduction

This chapter highlights the Software perspective, the scope of developed software & the logical flow of the program in the form of flowchart. It details the design process used in implementation. It features the front panel organization & the block diagram structure of the developed module. This section presents a cursory introduction to the developed VIô's that will help in understanding the software model.

5.2) Scope & System Overview

This software is developed mainly keeping in mind the needs of Ethernet validation system. The functions possible with this software are as follows:

5.2.1) Basic Features:

- control of logic analyzer activation
- Running of logic analyzer in both single and continuous acquisition mode.
- Data Acquisition from Logic Analyzer

5.2.2) Application specific Features

- As per validation requirement extraction of the relevant data from the acquired data
- Comparison module to check the authenticity of the acquired data.
- Saving the Erroneous data & its details for future reference
- Up to 64Kbytes data can be acquired without compromising time

5.2.3) Design Aspects

This application is specifically designed for MII interfacing mode .The MII transfers data using 4-bit words (nibble) in each direction it uses 4 data lines, clocked at 25 MHz to achieve 100 Mbit/s speed .

- **MII signals** include:

- Transmit Data (4 data lines): TXD0, TXD1, TXD2, TXD3
- Transmit strobe/ Transmit enable: TXEN
- Transmit clock: TXCLK
- , Transmit error: TXER,
- Receive data(4 data lines): RXD0,RXD1,RXD2,RXD3,
- Receive Strobe/Receive Data Valid :RXDV
- Receive clock: RXCLK,
- Receive Error: RXER,
- Collision indication, Carrier sense.

Ethernet frame captured by the Logic Analyzer will include Preamble, CRC Interframe Gap. During transmission of packet TXCLK become high, transmit enable signal (TXEN) validates the data if it is high data is said to be valid ,time frame between two packets is known as interframe gap at which TXEN signal will be low. During reception of packet RXCLK will be high , Receive data valid signal (RXDV) validates the received data if it is high data is said to be valid & at interframe gap RXDV signal will be low.

Due to MII Interfacing 4 bit of data will be transmitted/received per clock cycle i.e. data transmitted & received will be in nibble form. Pattern of data is in byte format. To receive/transmit a byte system will take two clock cycles. Logic Analyzer connected to the validation board will capture the data based on interfacing mode, for MII mode it will capture the data in Nibble form. Since data is in Nibble form to get a complete packet No. of samples to be acquired from Logic Analyzer through Lab view must be in accordance with the given relation:

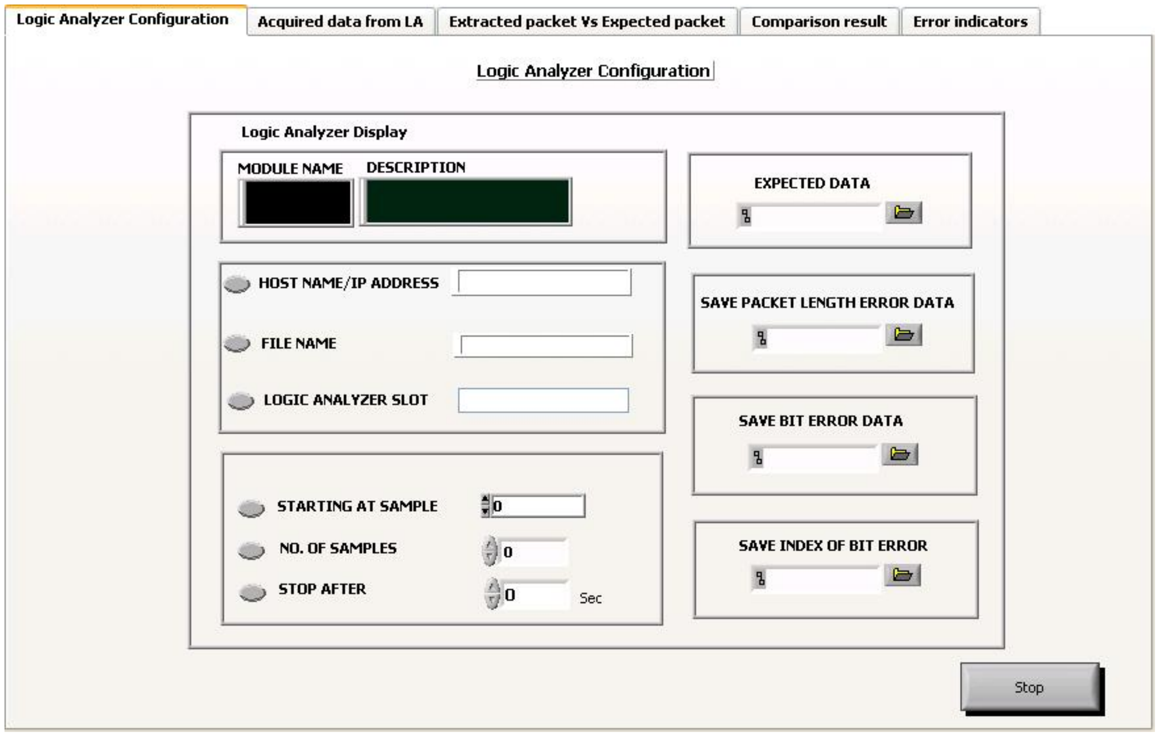
No of samples > 2 times [(Expected Packet+Preamble+Crc) bytes]

♦ **Logic Analyzer's configuration file format**

It is configured in State Mode, Signal probed to capture either the transmitted data or the received data are TXD0, TXD1, TXD2, TXD3 , TXEN & RXD0,RXD1,RXD2,RXD3 respectively, Acquisition Depth is set based on Packet Size & analysis requirements.

5.3) Software Model

5.3.1) Front Panel Description



STARTING AT SAMPLE 0
NO. OF SAMPLES 0
STOP AFTER 0 Sec

Figure: 5.1 Program graphic user interface representing section1 of Main acquisition.VI

This is the main window of the developed application. It is constructed in sequences. A detail of this sequence is given below. User window is tabbed into five sections each section represents the relevant analysed data. The five main sections are listed below:

a) ***Section 1: Logic Analyzer Configuration:***

b) ***Section 2: Acquired data from LA***

c) ***Section 3: Extracted Packet Vs Expected Packet***

d) ***Section 4 :Comparison Result***

e) ***Section 5 :Error Indicators***

a) *Section 1: Logic Analyzer Configuration:*

This window mainly represents the user defined controls and it indicates whether the connection with Logic Analyzer is established or not. The major user defined inputs are listed below:

- **Logic Analyzer Display**

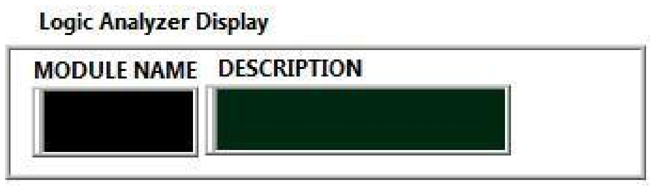


Figure: 5.2 Logic Analyzer Display before acquisition

Once lab view establishes its connection with the Logic Analyzer this screen indicate the Logic Analyzerôs mode of operation sampling frequency, slot name etc figure 5.2 represent logic analyzer display in acquisition mode

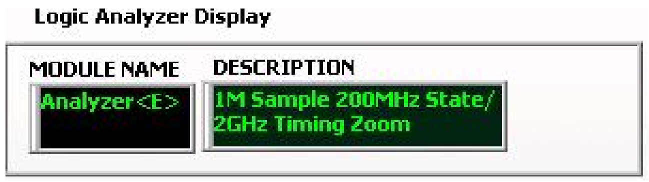


Figure: 5.3 Logic Analyzer Display after acquisition

- **The major user defined inputs are listed below**:

Sl No.	Control Name	Function
a)	**Host Name /IP Address**	Input IP address of Logic Analyzer
b)	**File Name**	Input configuration file to be loaded from Logic Analyzer
c)	**Logic Analyzer slot**	Input logic Analyzer slot name
d)	**Starting at sample**	Input offset of samples to be acquired
e)	**No. of samples**	Input no of samples to be acquired from logic Analyzer
f)	**Stop After**	Input timeout in case logic Analyzer never triggers. Acquisition will stop after this time out.

g)	**Expected Data**	Input path of Expected data file
h)	**Save bit error data**	Input path to save Bit error data
i)	**Save index of bit error**	Input path to save indexes of bit error data
J)	**Save packet length error data**	Input path to save packet length error data
k)	**Labels**	Input the. Data labels required to be acquired from the LA (It is in section 2: acquired data)
L)		**Single acquisition mode** This button must be kept in true (pressed) state. **Continuous acquisition mode** this button must be kept in false state. Press it to stop acquisition when the module is running.

Table 5.1 User Defined Inputs

b) ***Section 2: Acquired data from LA***

This section displays the data acquired from logic analyzer once the acquisition is complete. The index represents the no. of samples acquired from Logic Analyzer.

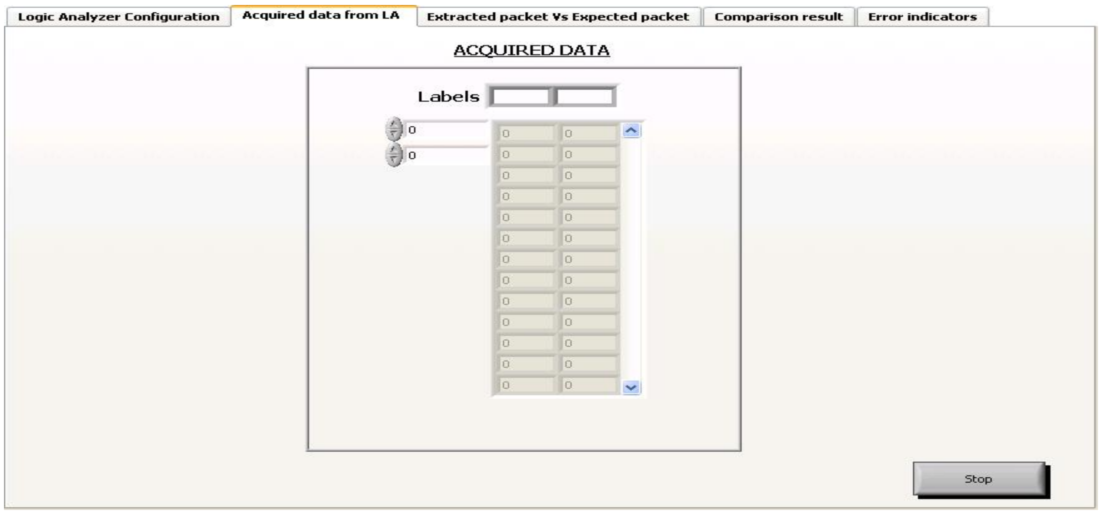


Figure: 5.4 Section 2 of Main Acquisition.VI

C) ***Section 3: Extracted Packet Vs Expected Packet***

This section displays the result of extraction of relevant data from the data acquired from Logic Analyzer. It displays the different features like ***Source Address***, ***Destination Address***, ***Ether Type***, ***CRC*** ***and*** ***Preamble*** of received data & expected data.

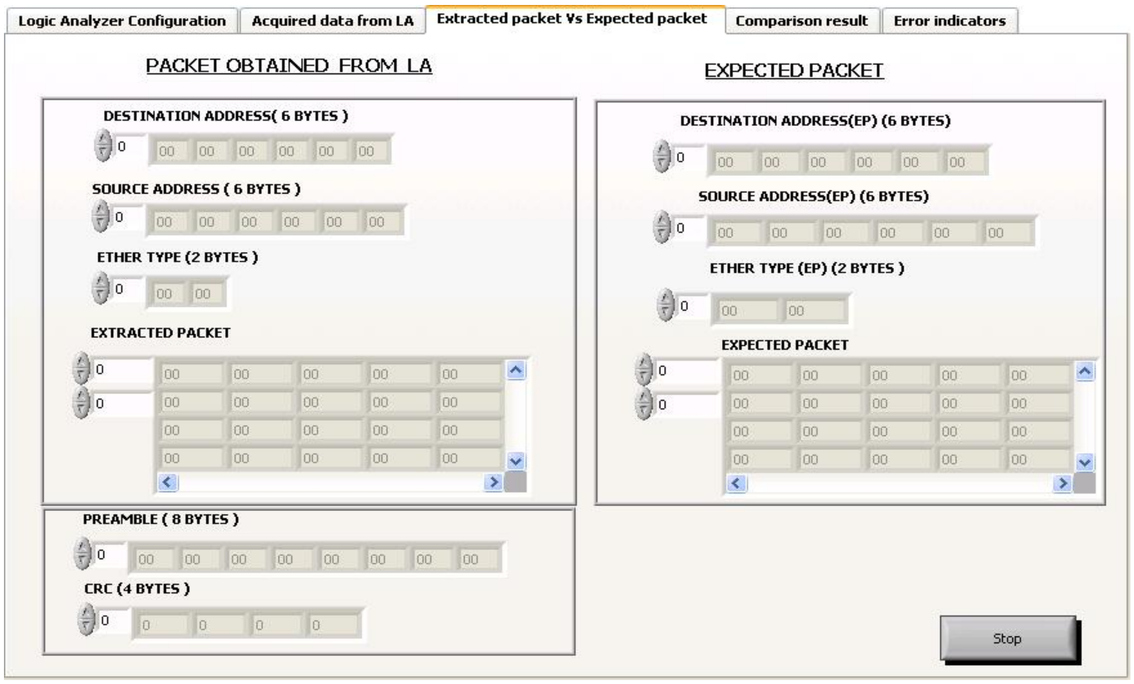

Logic Analyzer Configuration
Acquired data from LA
Extracted packet Vs Expected packet
Comparison result
Error indicators
PACKET OBTAINED FROM LA
DESTINATION ADDRESS(6 BYTES)
SOURCE ADDRESS (6 BYTES)
ETHER TYPE (2 BYTES)
EXTRACTED PACKET
PREAMBLE (8 BYTES)
CRC (4 BYTES)
EXPECTED PACKET
DESTINATION ADDRESS(EP) (6 BYTES)
SOURCE ADDRESS(EP) (6 BYTES)
ETHER TYPE (EP) (2 BYTES)
EXPECTED PACKET
Stop

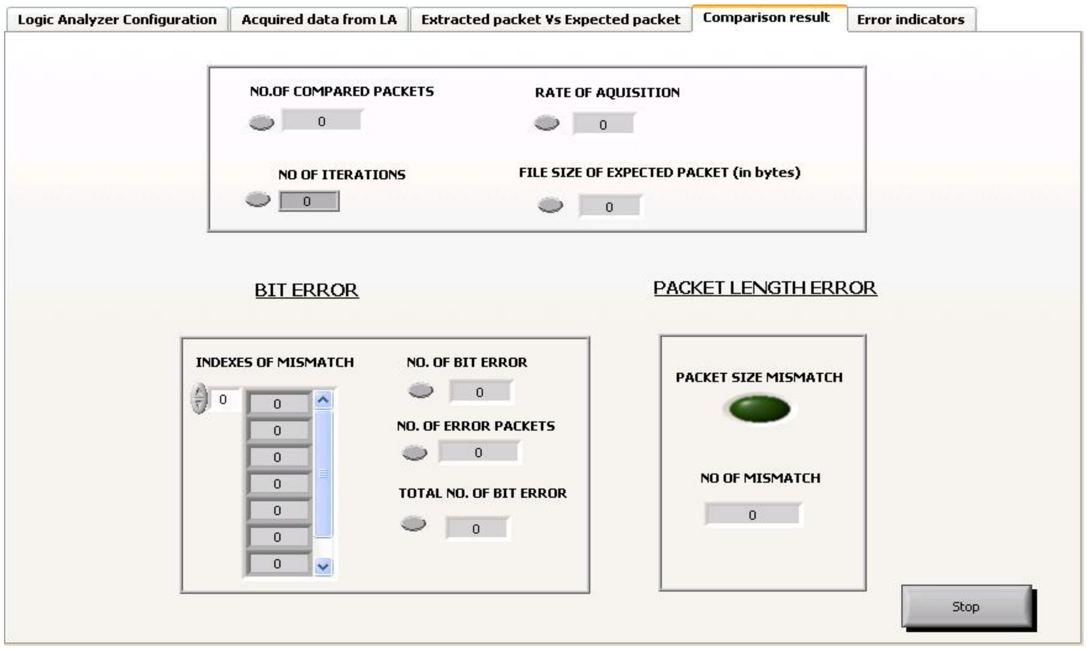
Figure: 5.5 Section 3 of Main Acquisition.VI

D) ***Section 4: Comparison Result***

This section displays the result of comparison of Extracted Packet with Expected Packet. It features the ***no of compared packets***, ***Size of Expected Packet***, ***rate of acquisition*** which indicate the time taken by lab view to acquire the specified no of samples & ***No. of iterations*** in case the module is running in continuous acquisition mode.

Further it features the **Bit error data** in which ***No. of bit error*** represents the bit error of each packet**,** ***index of mismatch*** indicate the indexes of each packet at which bit error occurred ***, total bit error*** indicate the total bit errors of all the compared packets & ***No. of error packets*** indicate the total bit error packets received .

In **Packet Length Error Data** indicator indicates the size mismatch it turns red when a mismatch occur and No. of Mismatch indicate the total packet length error data received.


Logic Analyzer Configuration
Acquired data from LA
Extracted packet Vs Expected packet
Comparison result
Error indicators
Stop

NO.OF COMPARED PACKETS 0
RATE OF AQUISITION 0
NO OF ITERATIONS 0
FILE SIZE OF EXPECTED PACKET (in bytes) 0

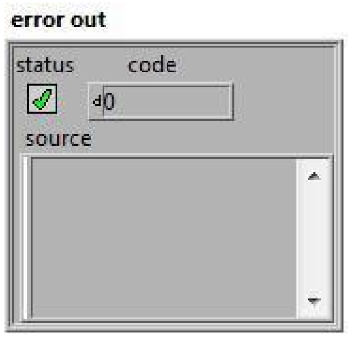
BIT ERROR
INDEXES OF MISMATCH 0
0 0 0 0 0 0 0
NO. OF BIT ERROR 0
NO. OF ERROR PACKETS 0
TOTAL NO. OF BIT ERROR 0

PACKET LENGTH ERROR
PACKET SIZE MISMATCH
NO OF MISMATCH 0

Figure: 5.6 Section 4 of Main Acquisition.VI

E) ***Section 5: Error Indicators***

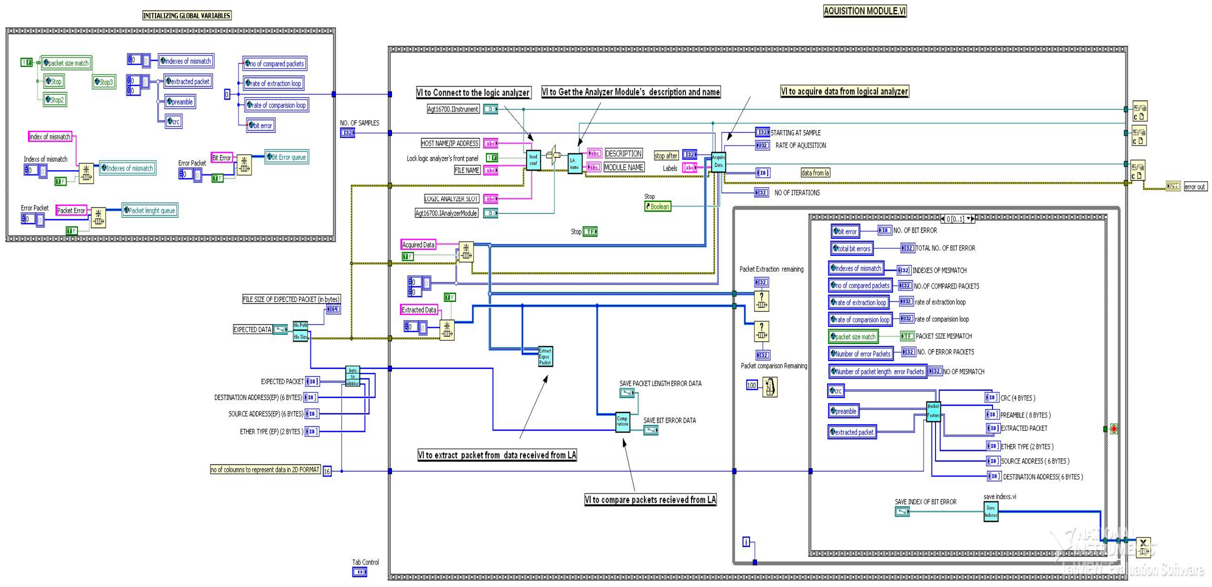
This section displays all sorts of error if acquisition fails.



error out
status ✓
code d0
source

Figure: 5.7 Section 5 of Main Acquisition.VI

5.3.2) Block Dia gram Description of Main Acquisition.VI



INITIALIZING GLOBAL VARIABLES
AQUISITION MODULE.VI
VI to Connect to the logic analyzer
VI to Get the Analyzer Module's description and name
VI to acquire data from logical analyzer
VI to extract packet from data received from LA
VI to compare packets recieved from LA

Figure: 5.8 Block Diagram View of Main Acquisition.VI

Figure 5.8 shows the block dia gram of the developed software model. It is the Main VI and consists of sixteen Sub VIôs. Figure 5.9 represents the **VI hierarchy** which depicts the relation of each Sub VI

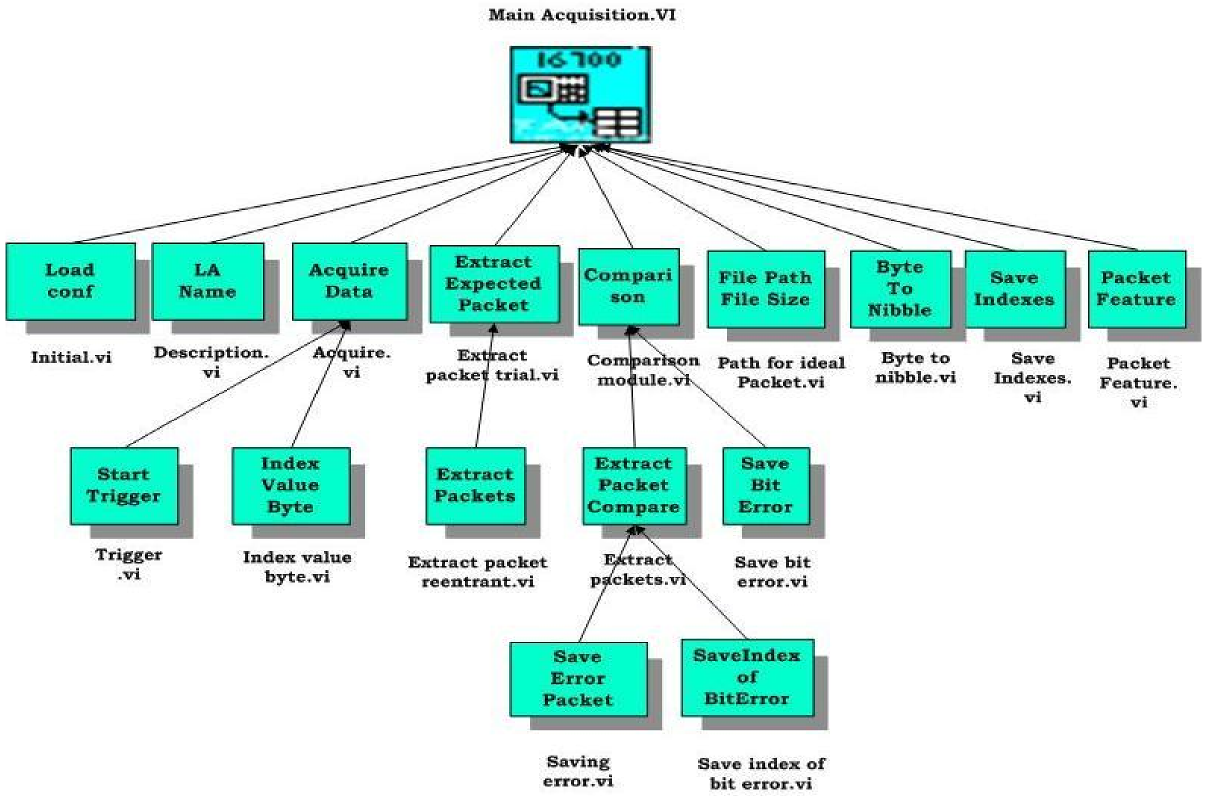
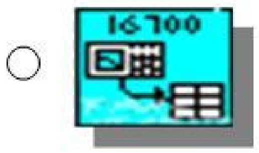


Figure: 5.9 VI Hierarchy

➢ **Sequence of Main acquisition.VI**

Main acquisition.VI



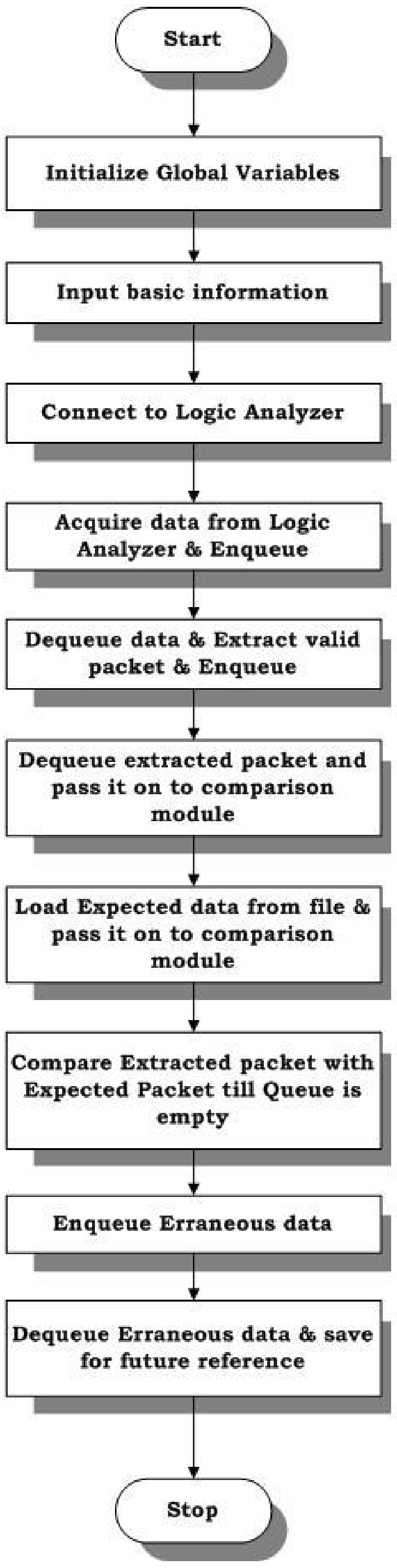

Start
Initialize Global Variables
Input basic information
Connect to Logic Analyzer
Acquire data from Logic Analyzer & Enqueue
Dequeue data & Extract valid packet & Enqueue
Dequeue extracted packet and pass it on to comparison module
Load Expected data from file & pass it on to comparison module
Compare Extracted packet with Expected Packet till Queue is empty
Enqueue Erraneous data
Dequeue Erraneous data & save for future reference
Stop

Figure: 5.10 Flow Chart of Main Acquisition.VI

➢ **Initialization block**

♦ **Description:** This block initializes the entire global variables.

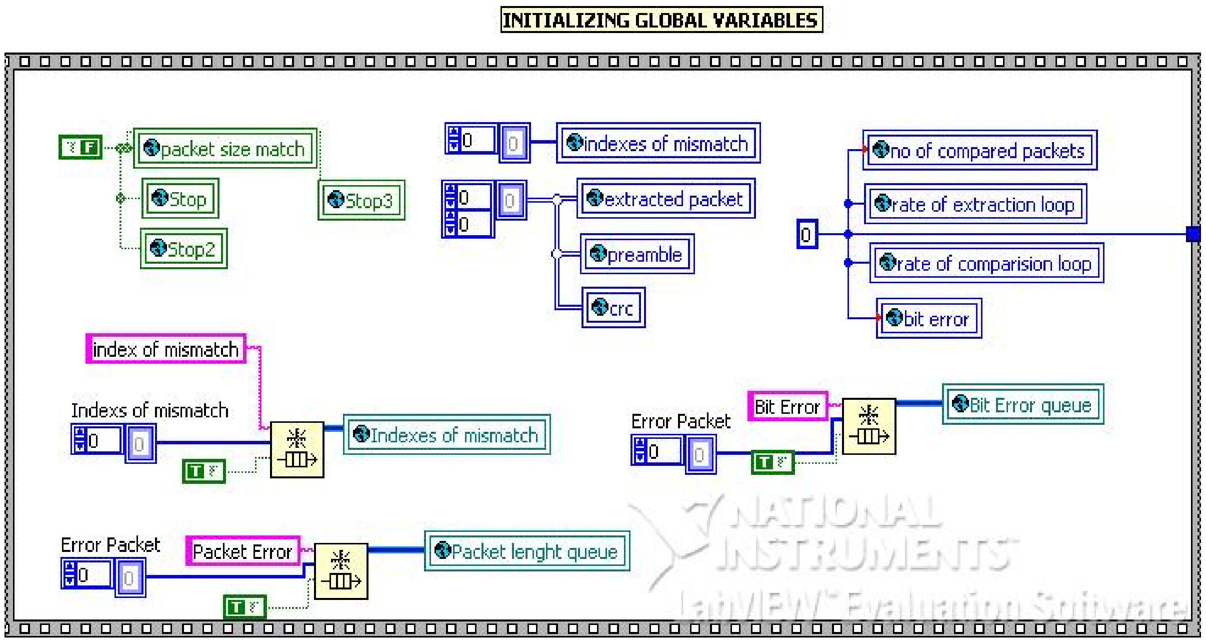


Figure: 5.11 Initialization block

5.3.3) Sub VI's Description

A VI within another VI is called a sub VI. A sub VI corresponds to a subroutine in text ï based programming languages. Here a total of 16 sub VIôs are used in Main Acquisition VI.

Following sections will elaborate the working of each sub VI.

A) Initial.VI

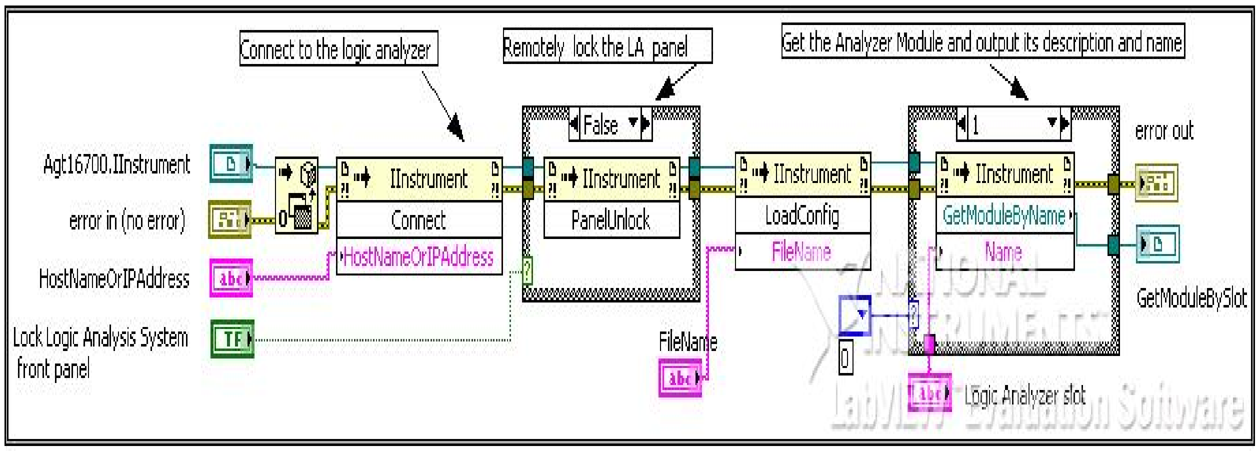


Figure: 5.12 Block Diagram of Sub VI Initial.VI

- **Description:** This sub- sequence is to input the basic information required to connect to the Logic Analyzer. Remote panel lock feature is also incorporated.

- **Inputs:** Hostname /IP Address, Logic Analyzer slot name, Configuration File to be loaded from Logic Analyzer.

B) <u>Description.VI</u>

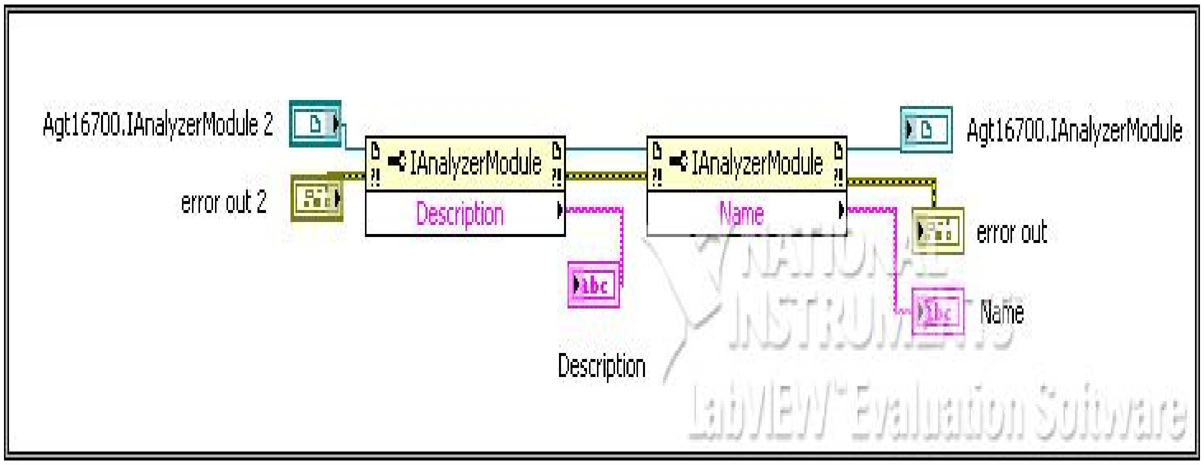
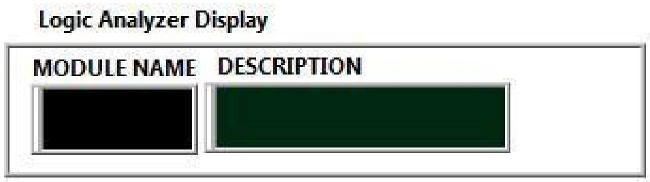


Figure: 5.13 Block Diagram of Description.VI

- **Description**

This sub- sequence is to display the logic analyzer module name and its mode of operation once the connection with the Logic Analyzer is established. If connection fails error out will display the possible reason of failure.

Output



- **Equivalent flow Chart Of Initial.VI & Description.VI**

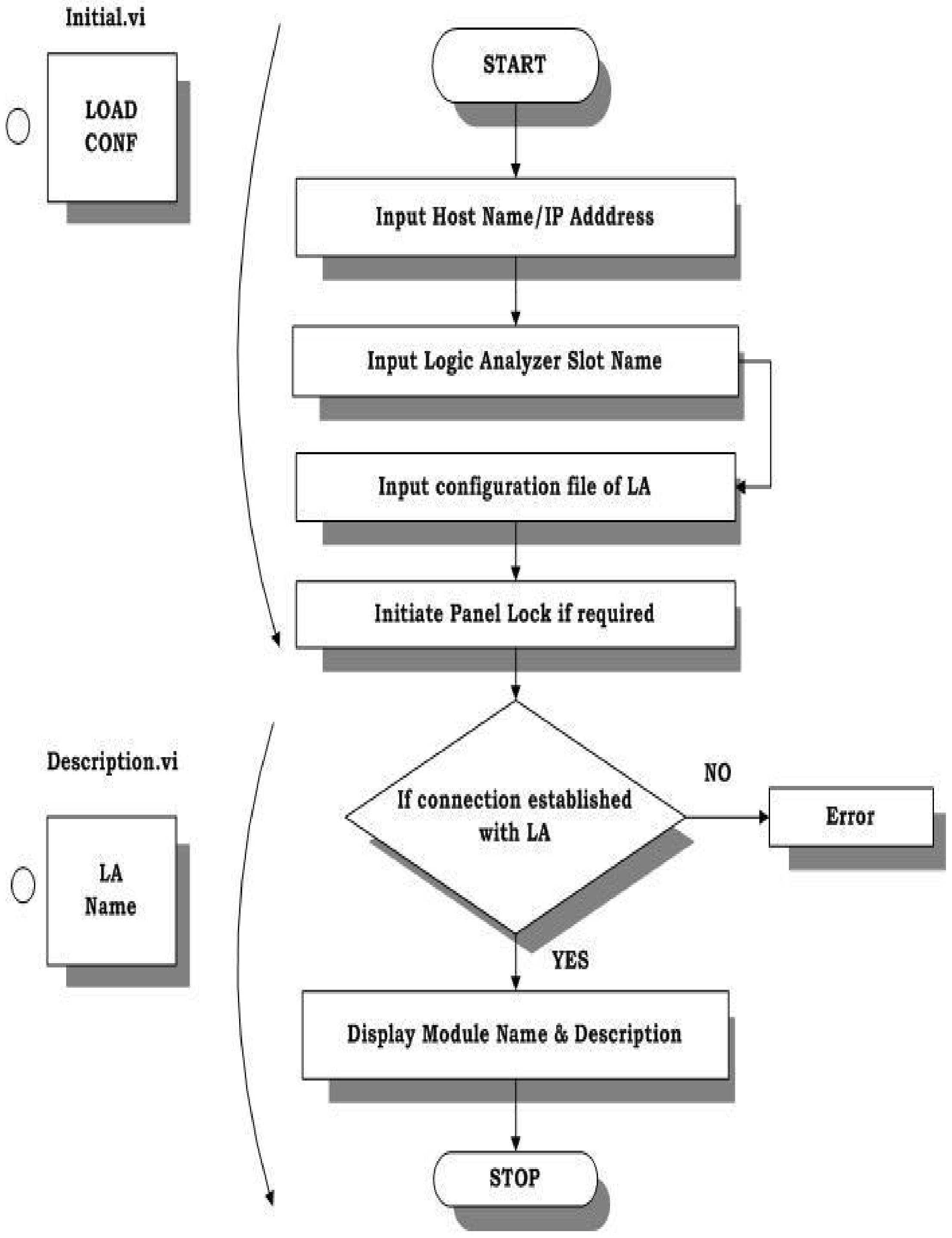

Initial.vi
LOAD CONF
START
Input Host Name/IP Adddress
Input Logic Analyzer Slot Name
Input configuration file of LA
Initiate Panel Lock if required
Description.vi
LA Name
If connection established with LA
NO
Error
YES
Display Module Name & Description
STOP

Figure: 5.14 Equivalent flow Chart Of Initial.VI & Description.VI

C) Trigger.VI

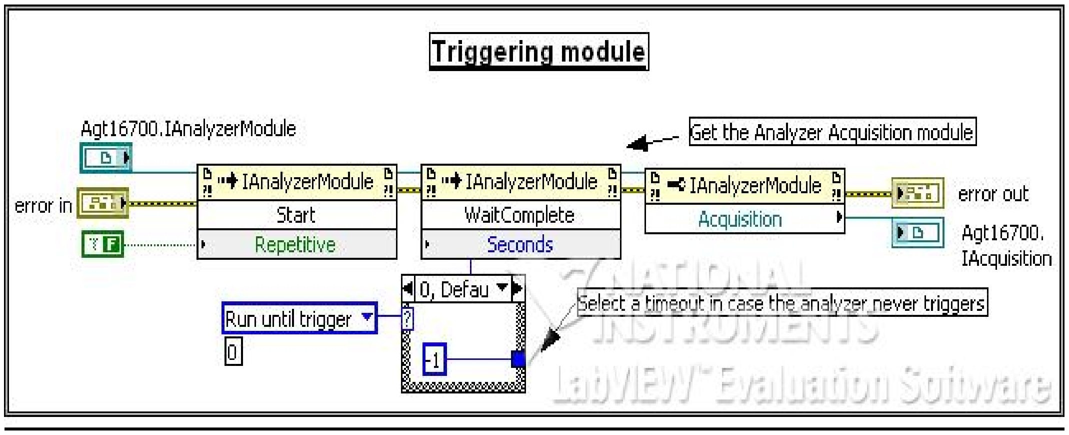
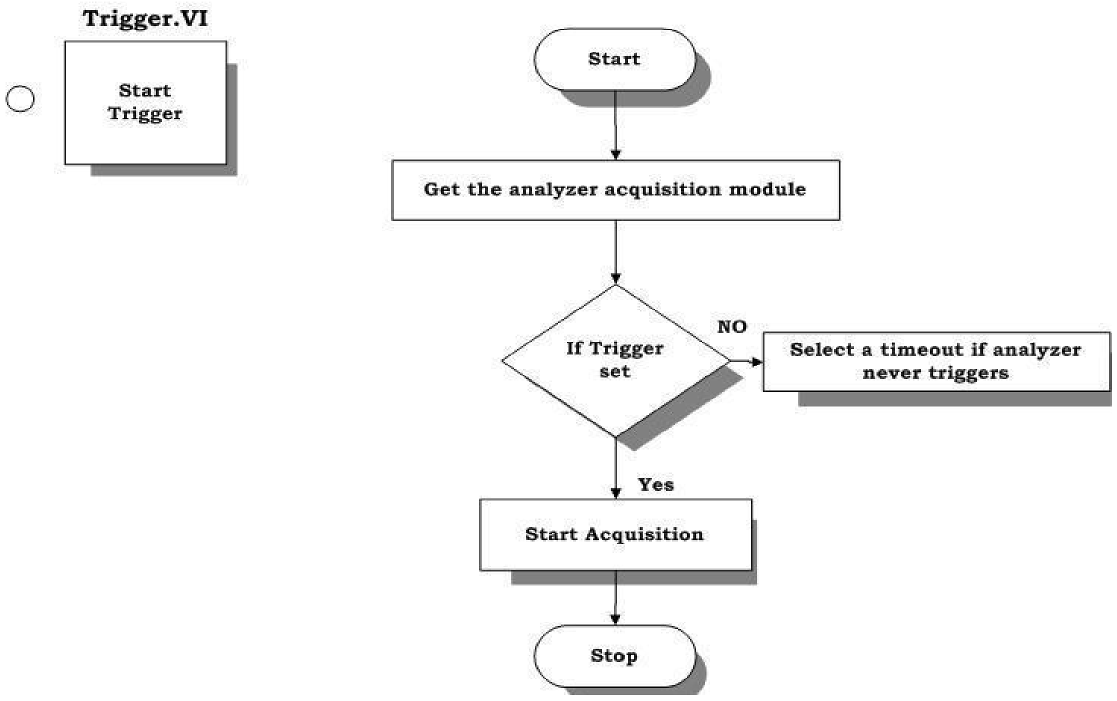


Figure: 5.15 Block Diagram of Trigger.VI

- **Description:** This sub- sequence initiates acquisition only if Logic Analyzer triggers which is set in the configuration file of LA otherwise it stops after the specified Time out
- **Input:** The duration for time out

➢ **Equivalent flow Chart**



Trigger.VI
Start Trigger
Start
Get the analyzer acquisition module
If Trigger set
NO
Select a timeout if analyzer never triggers
Yes
Start Acquisition
Stop

Figure: 5.16 Equivalent flow Chart of Trigger.VI

D) Acquire.VI

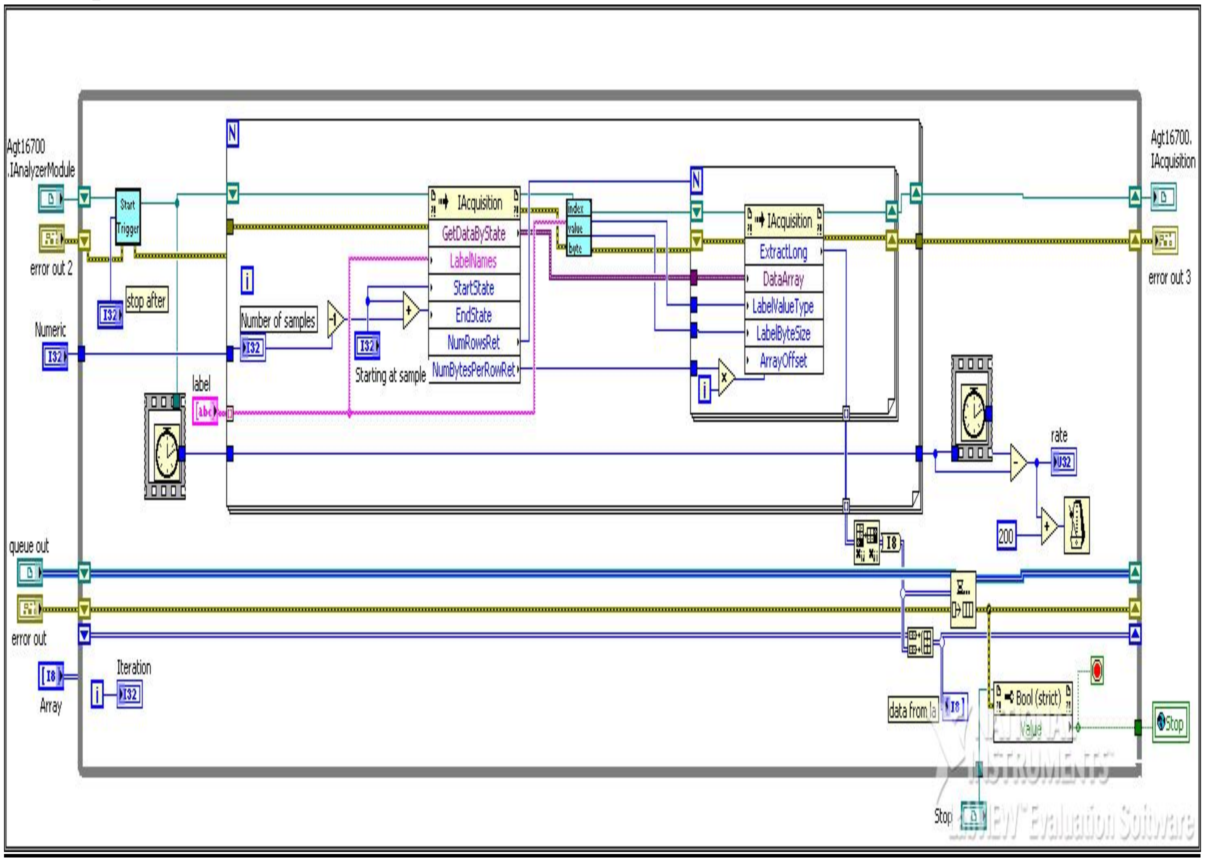


Figure: 5.17 Block Diagram of Acquire.VI

- **Description:**

This sub- sequence calls two sub VIôs namely Trigger.VI & Index value byte.VI. Once the logic analyzer triggers and acquires data the I Analyzer module of Trigger .VI pass its status & I Acquisition module initiates the acquisition. Data to be acquired from LA is identified by its label which must be specified, Index value byte.VI consists of the property nodes of I acquisition module which features the property of the acquired data based on which acquisition is carried out like label name, index, byte etc.Acquired data is put in a queue. Timer is used to indicate the rate of acquisition .Delay is included to give ample time to the acquisition module to get ready before next acquisition starts.

- **Inputs:** Starting at sample, No. of samples to be acquired, Labels to be acquired from Logic Analyzer

- **Outputs:** Data acquired by LA, No of iterations, Error statement if any

- **Equivalent flow Chart**

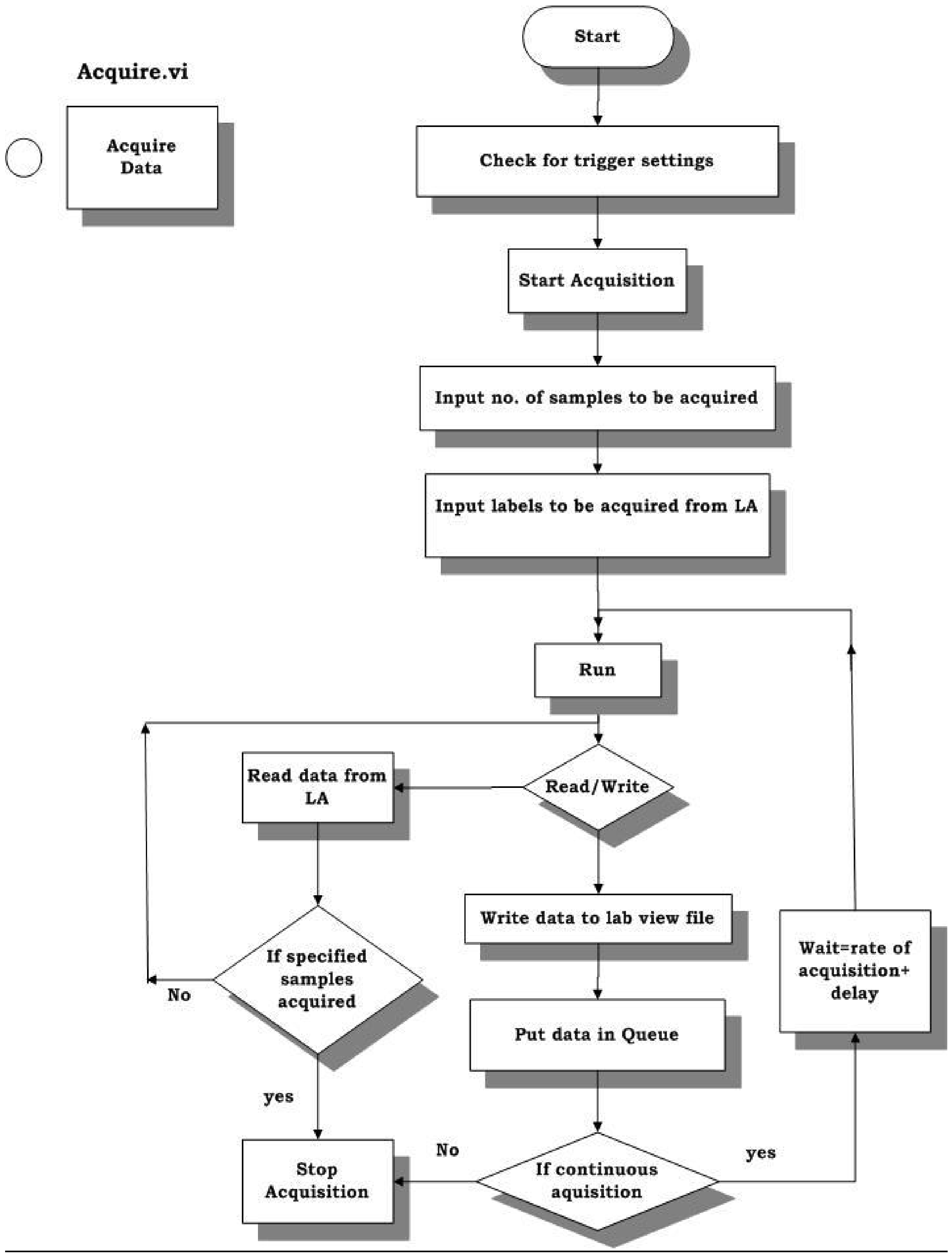

Acquire.vi
Acquire Data
Start
Check for trigger settings
Start Acquisition
Input no. of samples to be acquired
Input labels to be acquired from LA
Run
Read/Write
Read data from LA
If specified samples acquired
No
yes
Stop Acquisition
Write data to lab view file
Put data in Queue
If continuous aquisition
No
yes
Wait=rate of acquisition+ delay

Figure: 5.18 Equivalent flow Chart of Acquire.VI

E) Path for ideal Packet .VI

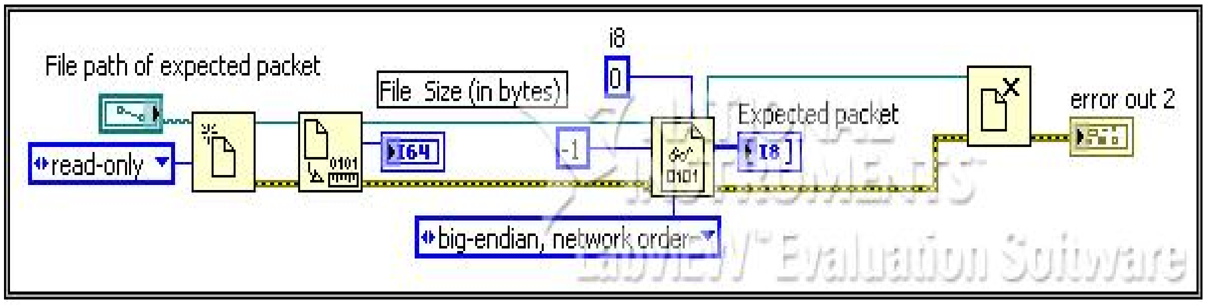

File path of expected packet
read-only
File Size (in bytes)
I64
i8
0
-1
Expected packet
I8
big-endian, network order
error out 2
LabVIEW Evaluation Software

Figure: 5.19 Block Diagram of Path for ideal packet.VI

- **Description:**

  This sub- sequence is used to read the Expected packet from its location and display its size in bytes.

- **Input:** Path of Expected packet

- **Output:** File size of Expected packet in bytes and Expected Packet in array format

➢ **Equivalent Flow Chart**

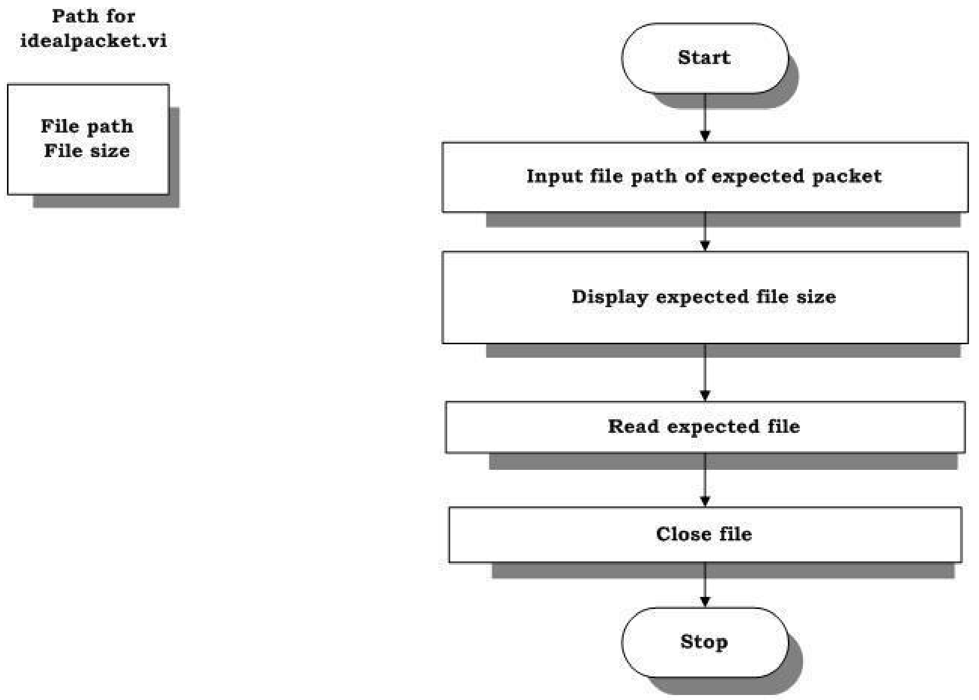


Figure: 5.20 Equivalent flow Chart of Path for Ideal Packet.VI

F) Byte to Nibbles .VI

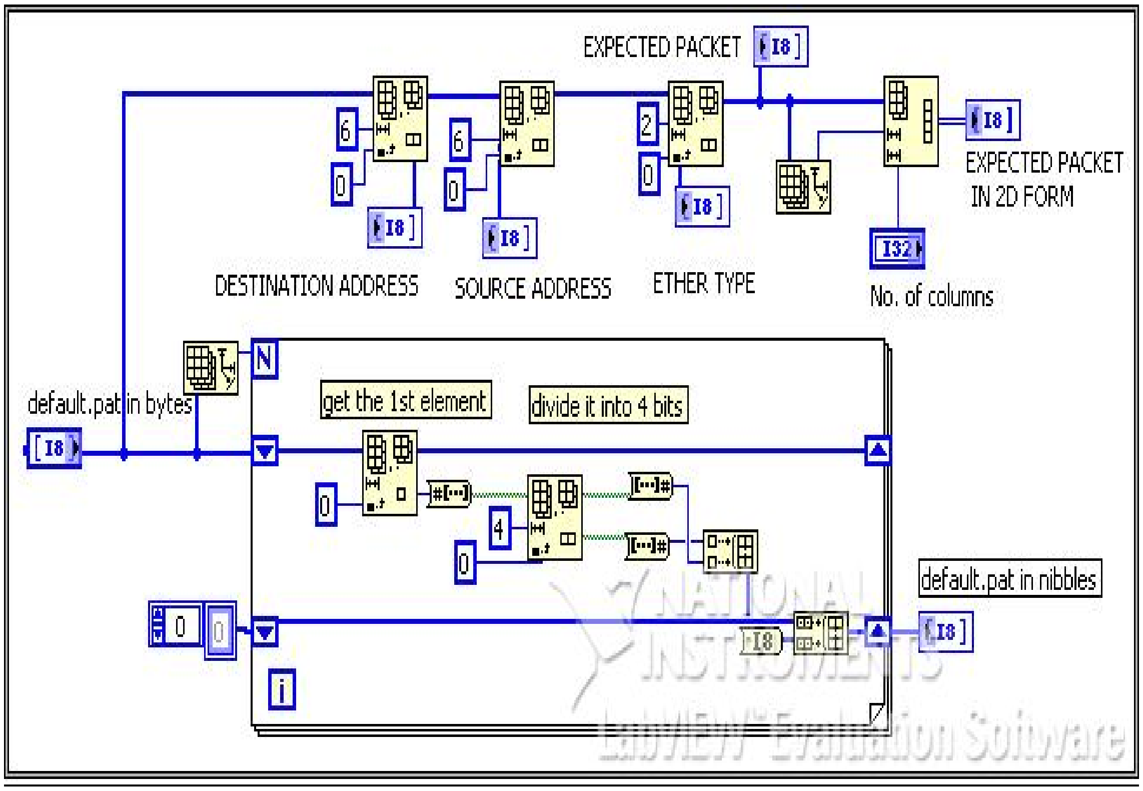


Figure: 5.21 Block Diagram of Byte to Nibbles.VI

- **Description**

This sub- sequence is used to convert the Expected packet from byte format to Nibbles as data acquired from Logic Analyzer is in nibbles format .Each byte is split it into its nibble form and lower order nibble is placed over the higher order nibble so as to synchronize with the data obtained from the logic analyzer. Also packet features like source address, destination address, ether type is separated.

- **Input:** Expected packet in bytes

- **Output:** Expected packet in Nibbles, source address, destination address, ether type.

➢ **<u>Equivalent Flow Chart</u>**

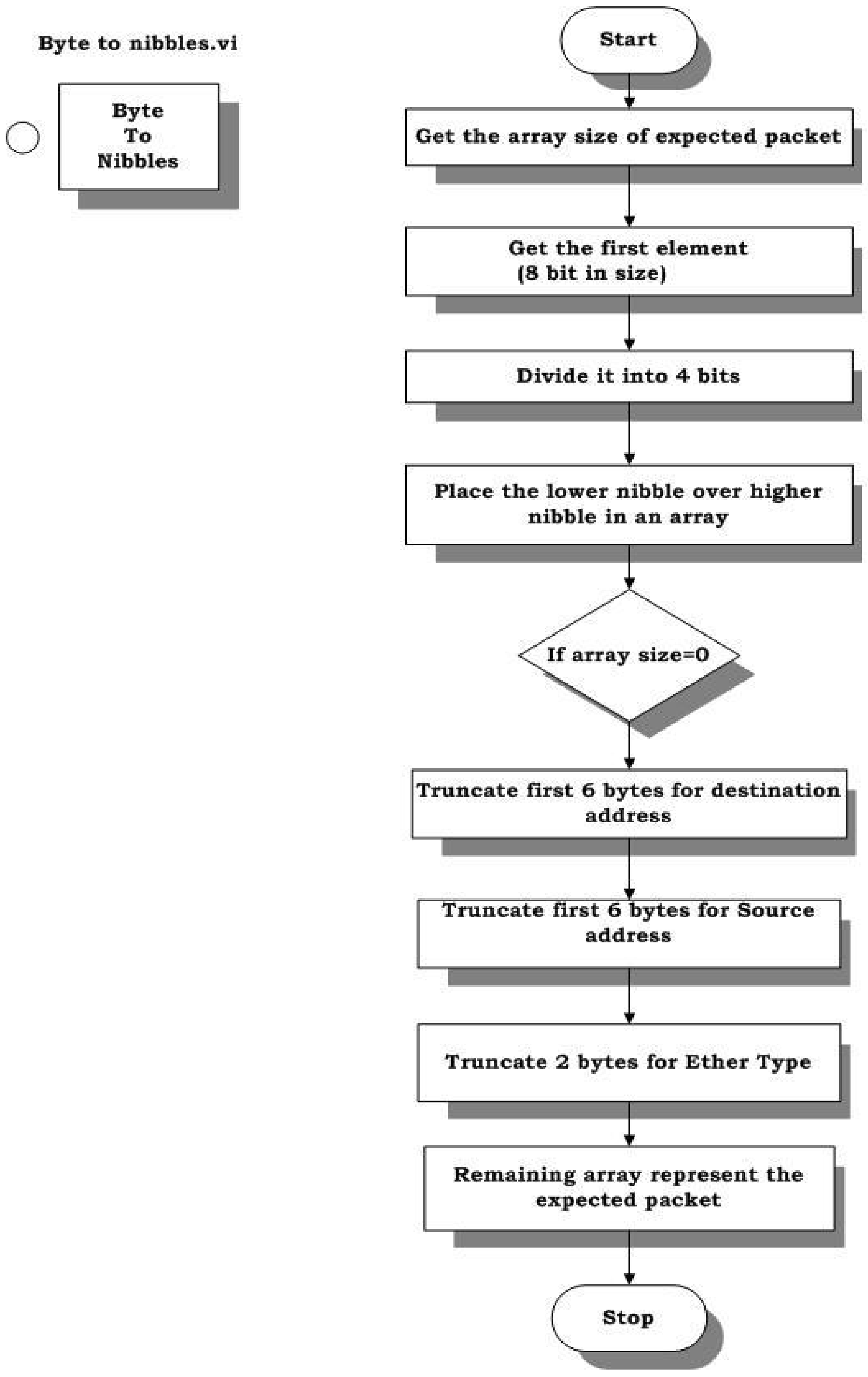

Byte to nibbles.vi
Byte To Nibbles
Start
Get the array size of expected packet
Get the first element (8 bit in size)
Divide it into 4 bits
Place the lower nibble over higher nibble in an array
If array size=0
Truncate first 6 bytes for destination address
Truncate first 6 bytes for Source address
Truncate 2 bytes for Ether Type
Remaining array represent the expected packet
Stop

Figure: 5.22 Equivalent flow Chart of Byte to Nibbles.VI

G) <u>Extract Packet Re-entrant .VI</u>

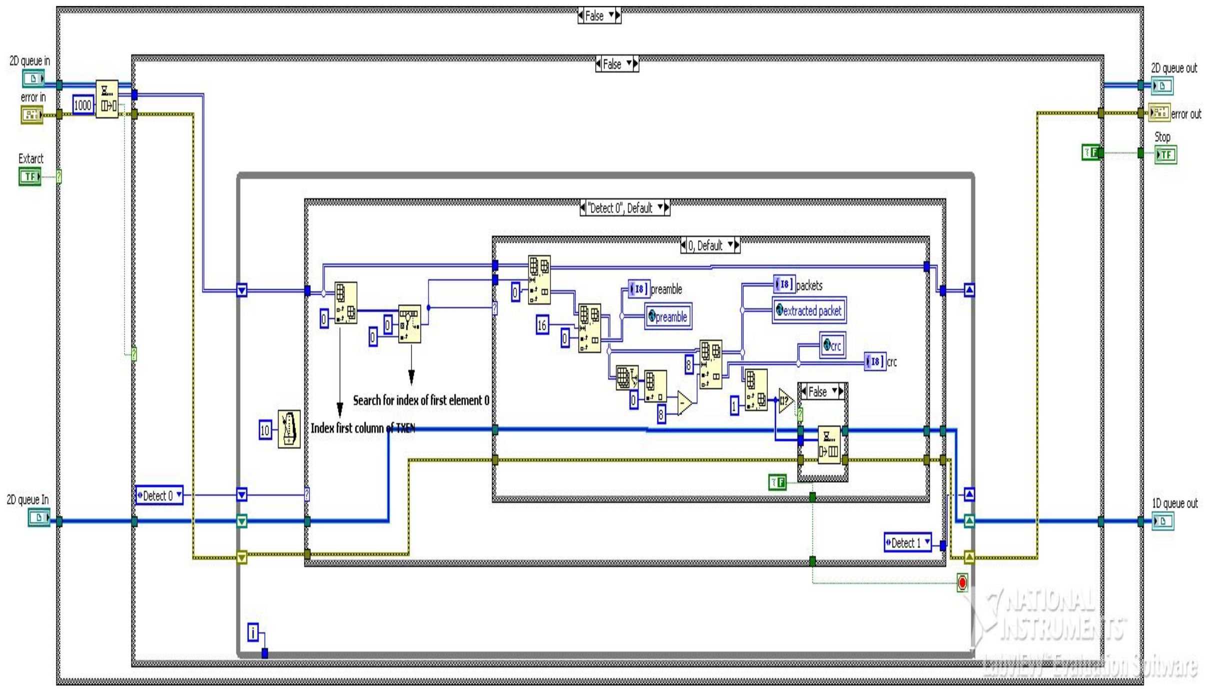


Figure: 5.23 Block Diagram of Extract Packet Re-entrant.VI

- **Description:**

This sub- sequence is used to extract the relevant packet from the data acquired from the Logic Analyzer. The data is valid when transmit enable (TXEN) line is high, TXEN is searched till it is high and the data is extracted till it turns low, truncate first 8bytes (preamble) & last 4bytes (CRC) of extracted packet and enqueue it & continue till all the packets are extracted.

- **Input:** Queue of acquired data from LA

- **Output:** Queue of extracted data

➢ **Equivalent Flow Chart**

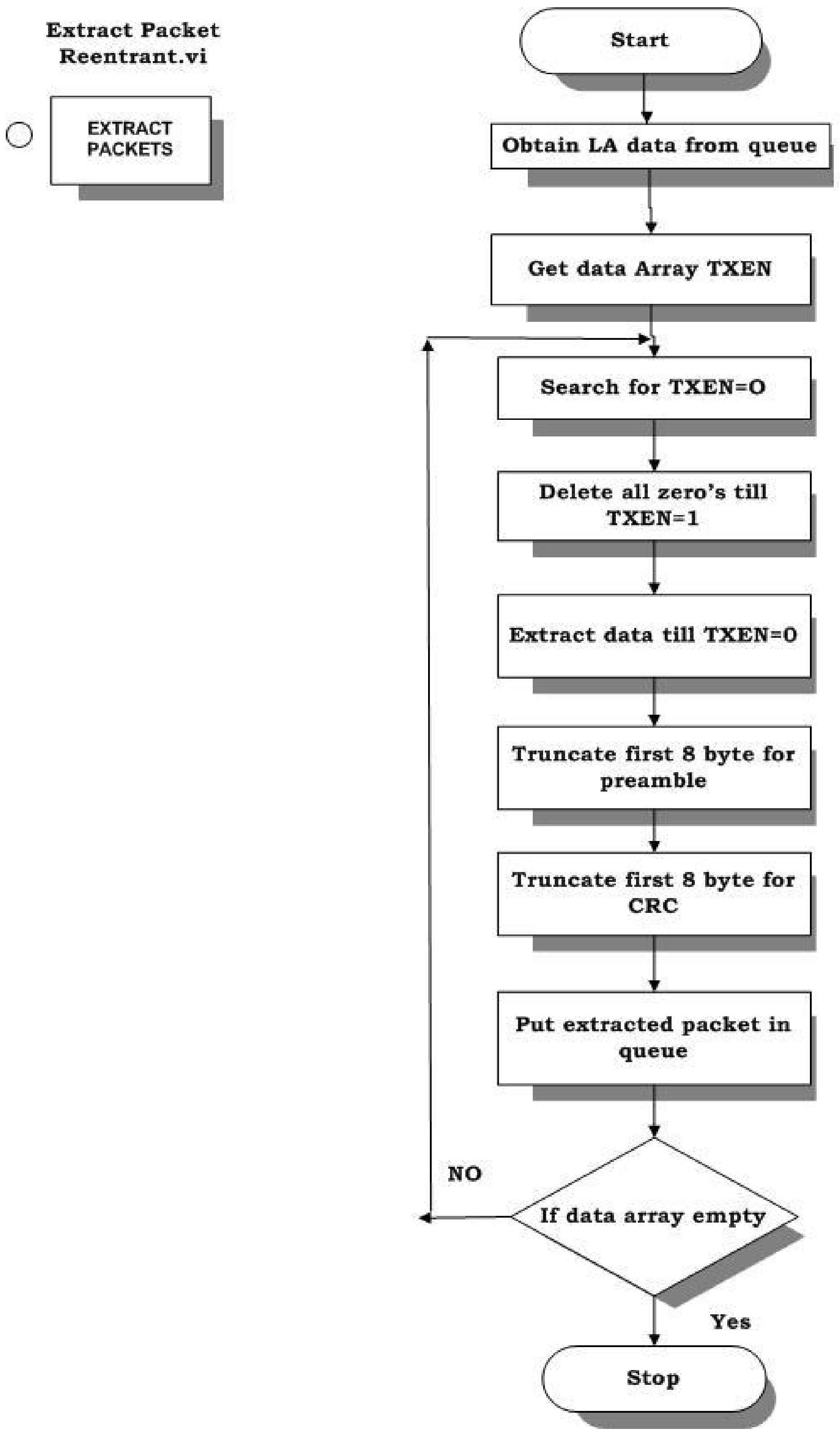

Extract Packet Reentrant.vi
EXTRACT PACKETS
Start
Obtain LA data from queue
Get data Array TXEN
Search for TXEN=O
Delete all zero's till TXEN=1
Extract data till TXEN=0
Truncate first 8 byte for preamble
Truncate first 8 byte for CRC
Put extracted packet in queue
If data array empty
NO
Yes
Stop

Figure: 5.24 Equivalent flow Chart of Extract Packet Re-entrant.VI

H) Extract Packet Trial .VI

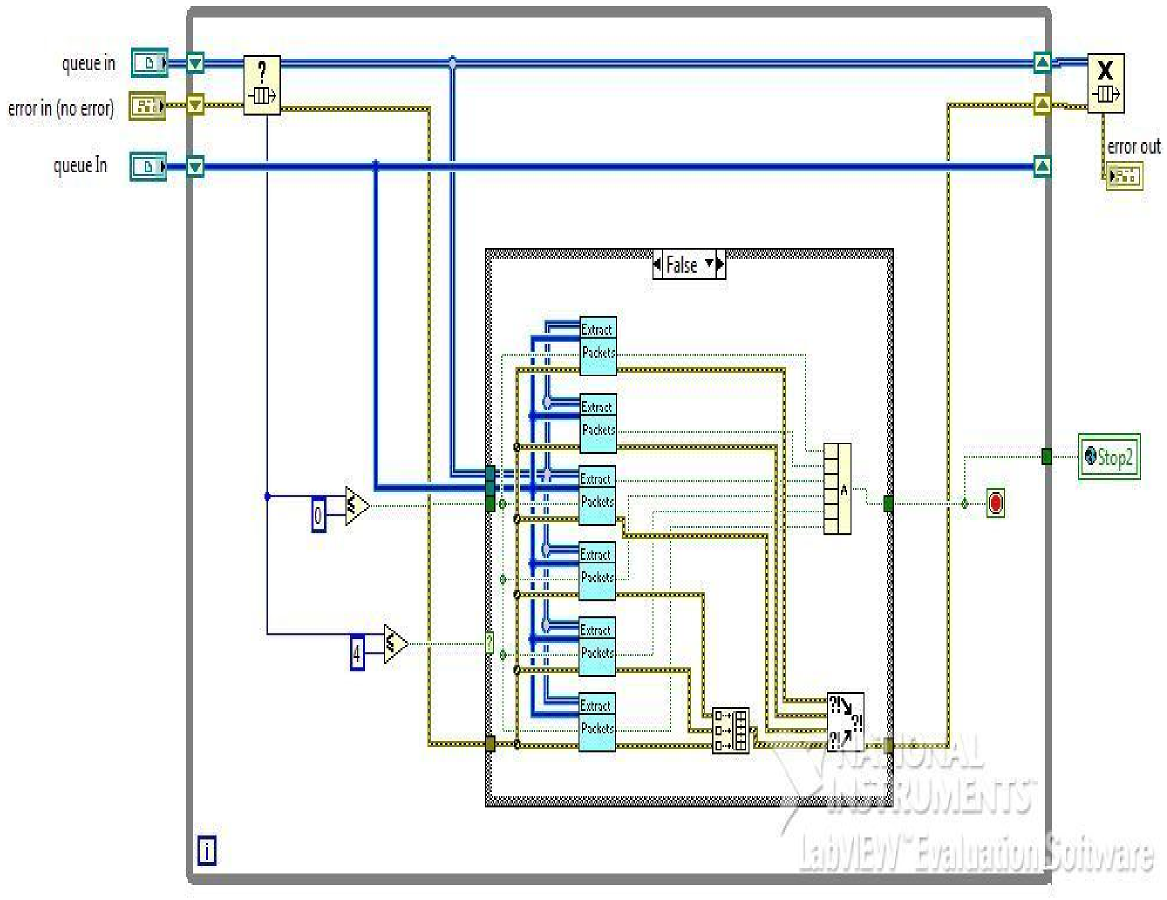


Figure: 5.25 Block Diagram of Extract Packet Trial.VI

- **Description**

This sub- sequence is basically designed to make the extraction process faster when the no. of packets to be Extracted in queue is more ,usually such situation occur when large amount of data is acquired or when the module is working in continuous acquisition mode. It uses clones of Extract packet Re-entrant.VI .All clones work in parallel they are independent of each other and all the execution done by clones are segregated together to get the final result. Based on packets remaining in queue number of clones vary when packets remaining in queue <4 single Extract packet Re-entrant VI is used and and for packets remaining in queue >=4 six clones of Extract packet Re-entrant.VI is used. It not only makes the execution faster but also it reduces the load on extraction module.

- **Input:** Queue of acquired data from LA

- **Output:** Queue of extracted data

➢ **Equivalent Flow Chart**

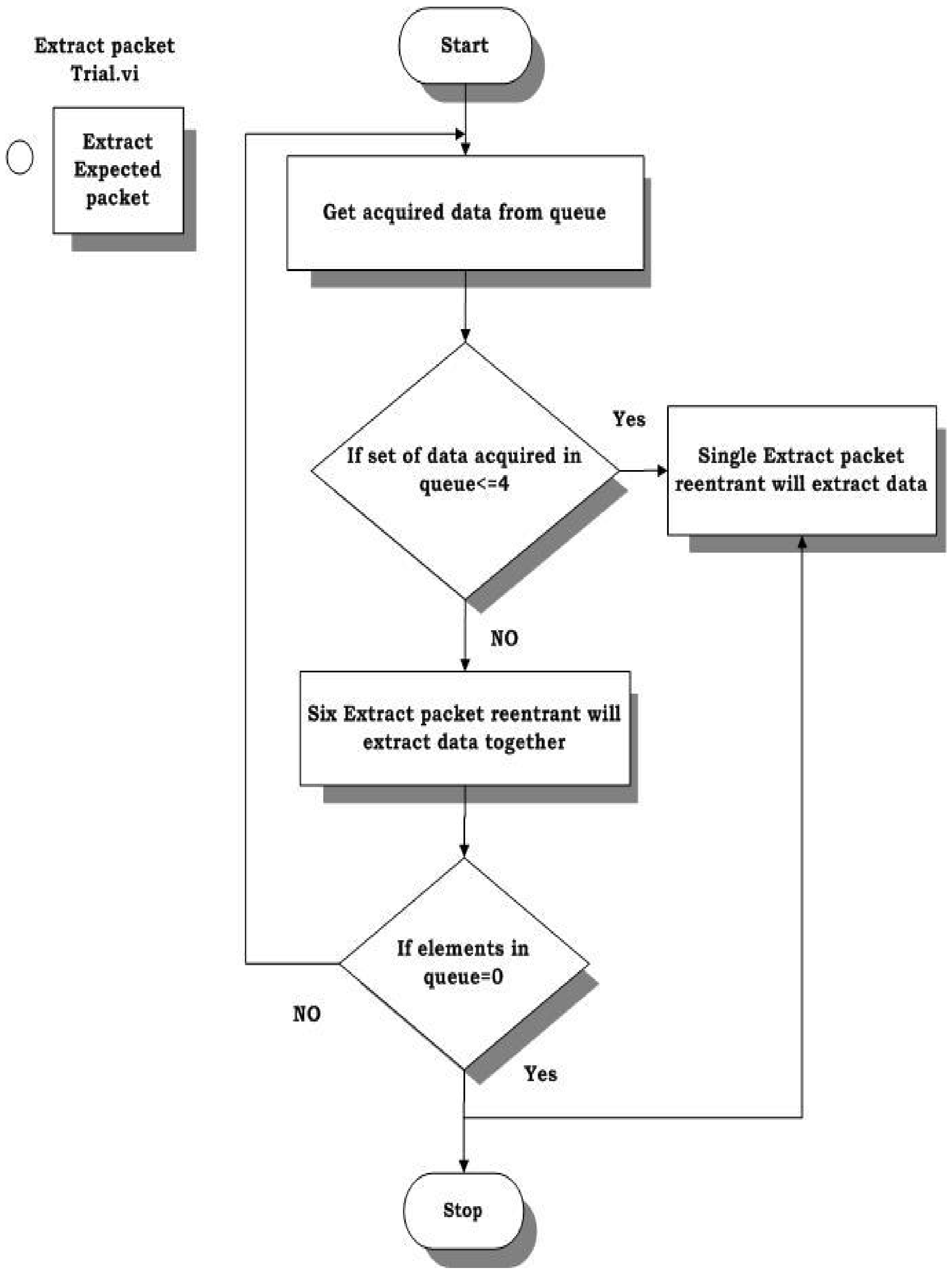

Extract packet Trial.vi
Extract Expected packet
Start
Get acquired data from queue
If set of data acquired in queue<=4
Yes
Single Extract packet reentrant will extract data
NO
Six Extract packet reentrant will extract data together
If elements in queue=0
NO
Yes
Stop

Figure: 5.26 Equivalent flow Chart of Extract Packet Trial.VI

I) Extract Packets .VI

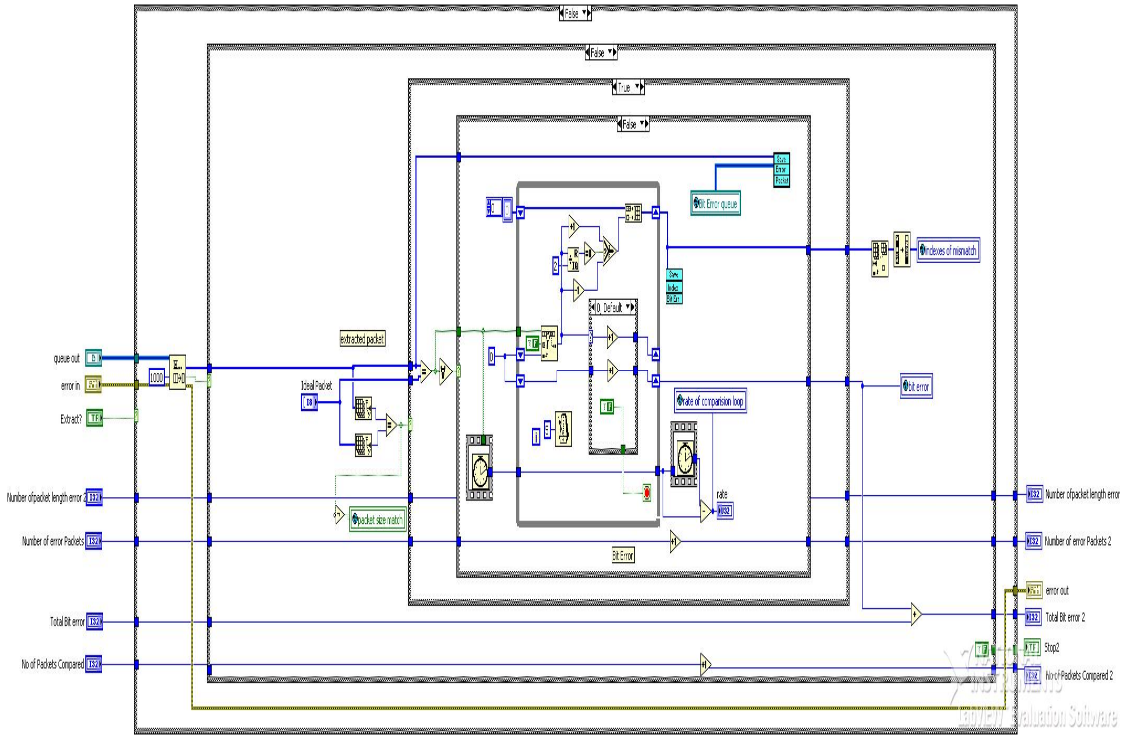


Figure: 5.27 Block Diagram of Extract Packets.VI

- **Description**

This sub- sequence is used to compare the Extracted packet with the Expected packet. Basically it is divided into 3 cases according to the comparison result. Extracted data from queue is compared with the Expected packet for same size if it is true then further nibble position is checked and if it is also true then next packet comparison begins. If failure occurs in first stage i.e. packet size mismatch then no further processing is done and the erroneous data is put in queue to save after processing. If failure occurs in second stage bit error data and its index is put in queue to save after processing, the whole process is done till queue is empty

- **Input:** Queue of Extracted packet

- **Output:** Comparison results like No. of packet compared, No of bit error, no of packet length error, total bit error.

➢ Equivalent Flow Chart

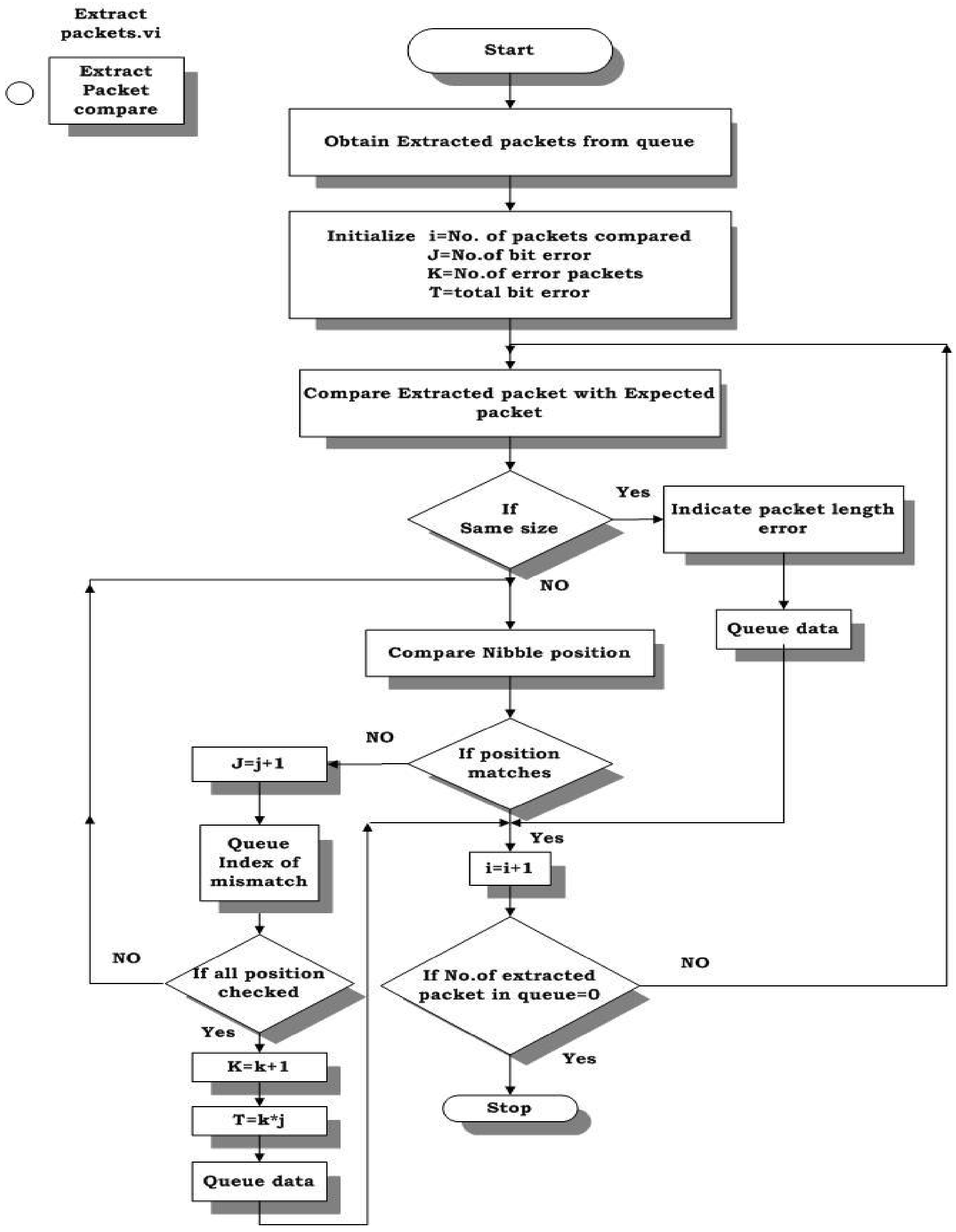


Figure: 5.28 Equivalent flow Chart of Extract Packets.VI

J) Comparison Module .VI

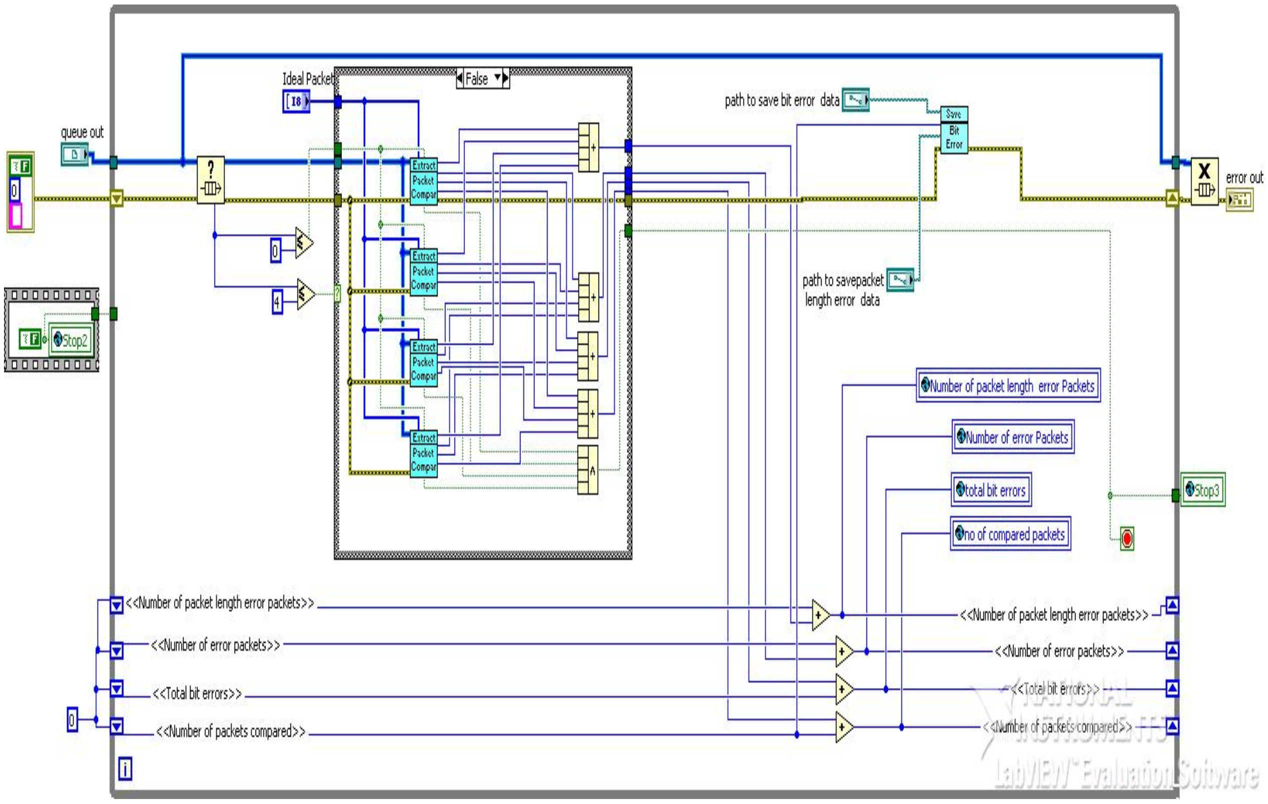

Ideal Packet
False
queue out
path to save bit error data
Save Bit Error
error out
path to savepacket length error data
Stop2
Extract Packet Compar
Number of packet length error Packets
Number of error Packets
total bit errors
no of compared packets
Stop3
<<Number of packet length error packets>>
<<Number of error packets>>
<<Total bit errors>>
<<Number of packets compared>>
LabVIEW Evaluation Software

Figure: 5.29 Block Diagram of Comparison Module.VI

- **Description**

This sub- sequence is basically designed to make the comparison process faster when the no. of packets to be compared in queue is more ,usually such situation occur when large amount of data is acquired or when the module is working in continuous acquisition mode. It uses clones of Extract packets.VI .All clones work in parallel they are independent of each other and all the execution done by clones are segregated together to get the final result. Based on packets remaining in queue number of clones vary, single Extract packets.VI is used when packets remaining in queue <4 and for packets remaining in queue >=4 four clones of Extract packets.VI is used. It not only makes the execution faster but also it reduces the load on comparison module.

- **Input:** Queue of extracted data

- **Output:** Comparison results like No. of packet compared, No of bit error, no of packet length error, total bit error.

➢ **Equivalent Flow Chart**

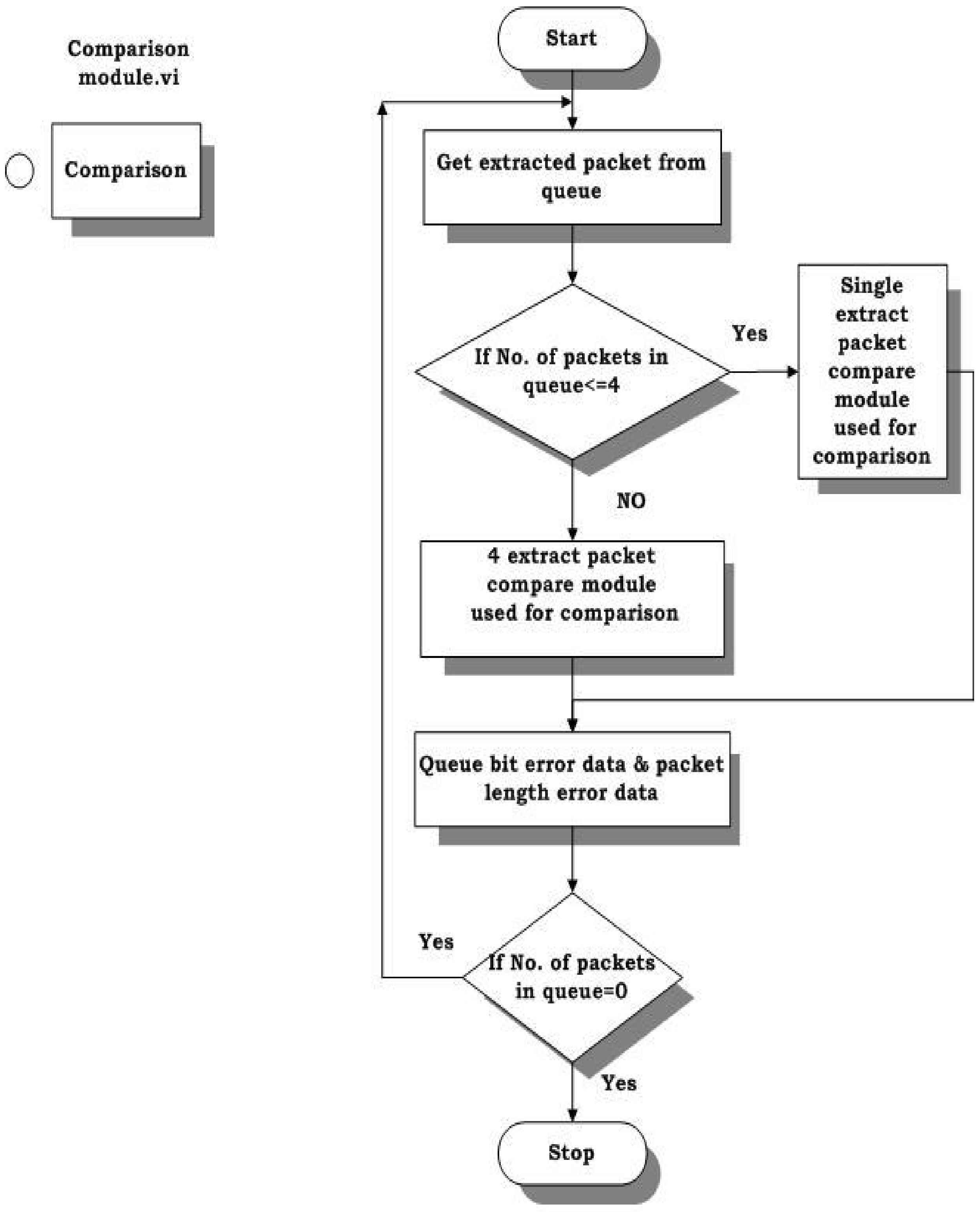

Comparison module.vi
Comparison
Start
Get extracted packet from queue
If No. of packets in queue<=4
Yes
Single extract packet compare module used for comparison
NO
4 extract packet compare module used for comparison
Queue bit error data & packet length error data
If No. of packets in queue=0
Yes
Yes
Stop

Figure: 5.30 Equivalent flow Chart of Comparison Module.VI

K) Save Bit Error .VI

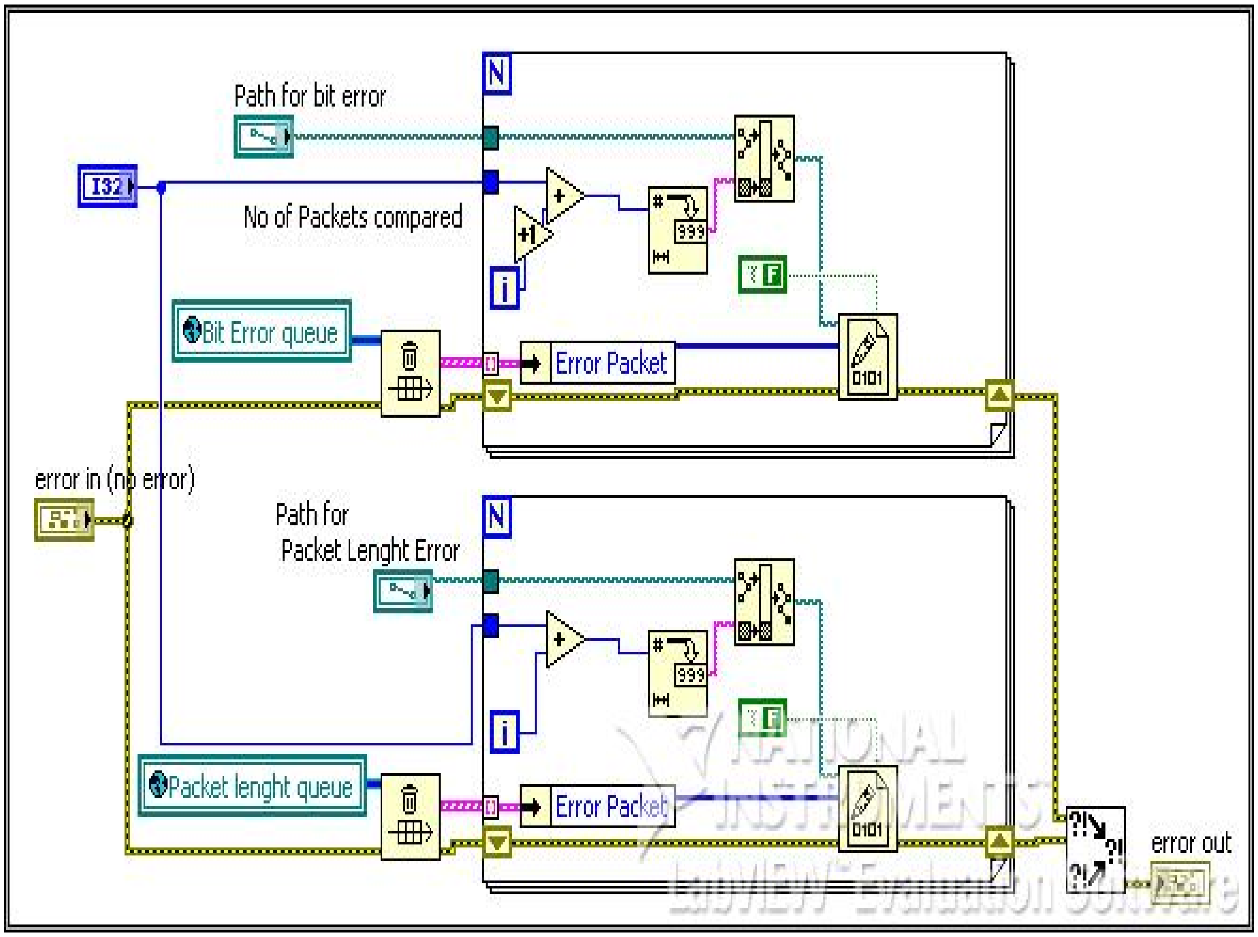


Figure: 5.31 Block Diagram of Save Bit Error.VI

- **Description**

  This sub- sequence is basically designed to save the erroneous data namely Bit Error Data & Packet Length Error Data. Both the data while execution of comparison module is put in two separate queues , to save the same in binary file they are retrieved from queue and the no of packets to be compared is specified so that data can be saved with the packet no.

- **Input:** Path to save Bit Error Data & Packet Length Error Data, No of compared Packets.

➢ **Equivalent Flow Chart**

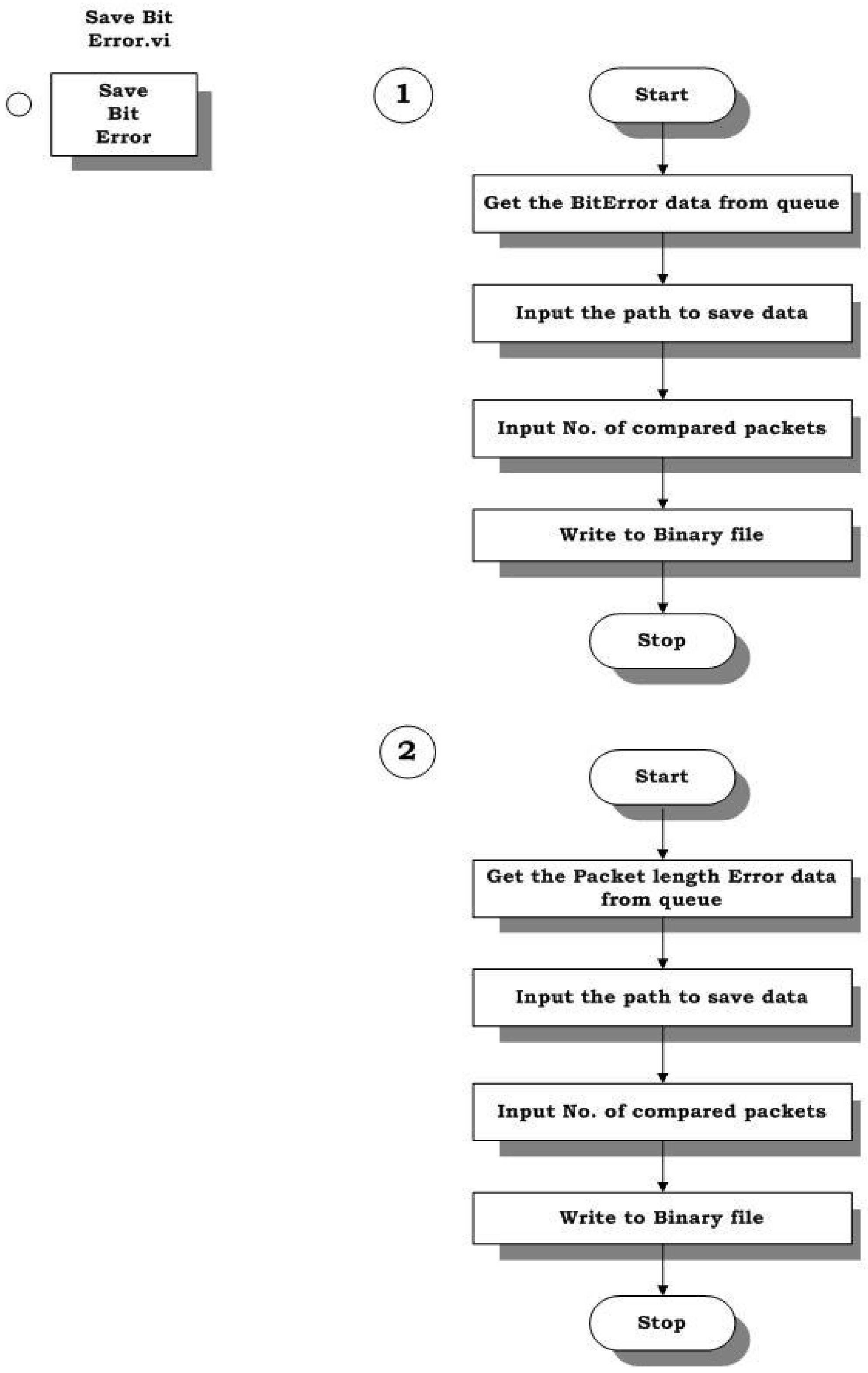

Save Bit Error.vi
Save Bit Error
1
Start
Get the BitError data from queue
Input the path to save data
Input No. of compared packets
Write to Binary file
Stop
2
Start
Get the Packet length Error data from queue
Input the path to save data
Input No. of compared packets
Write to Binary file
Stop

Figure: 5.32 Equivalent flow Chart of Save Bit Error.VI

L) Packet Feature View .VI

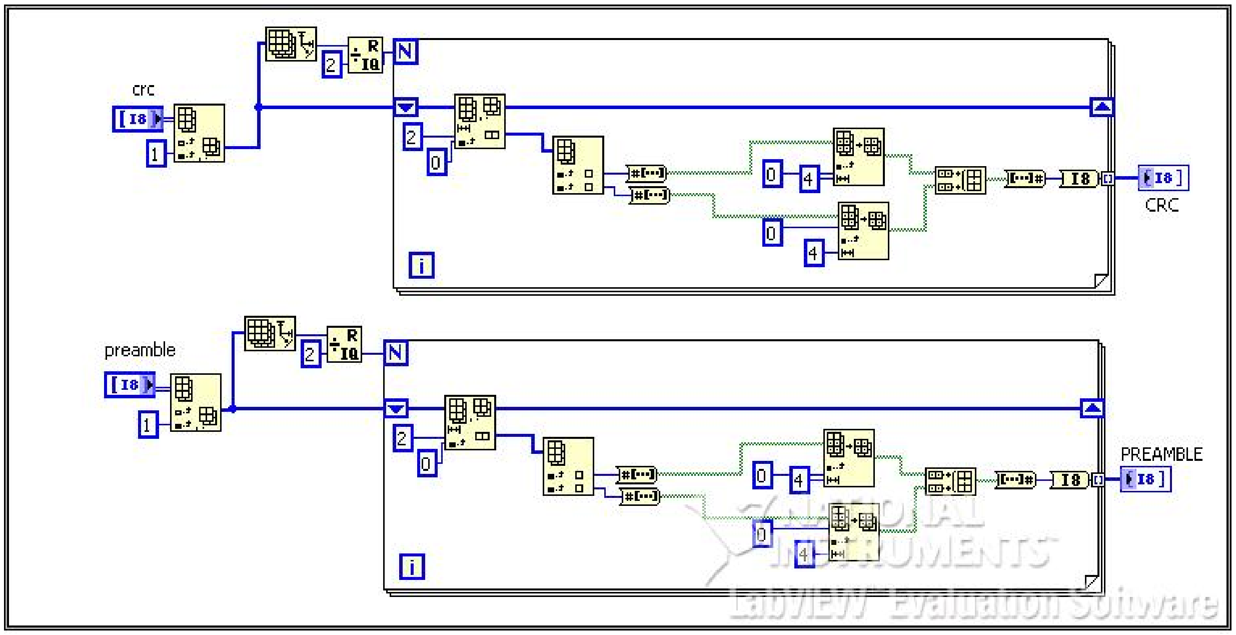


Figure: 5.33 Block Diagram of Nibble to byte conversion of CRC & Preamble

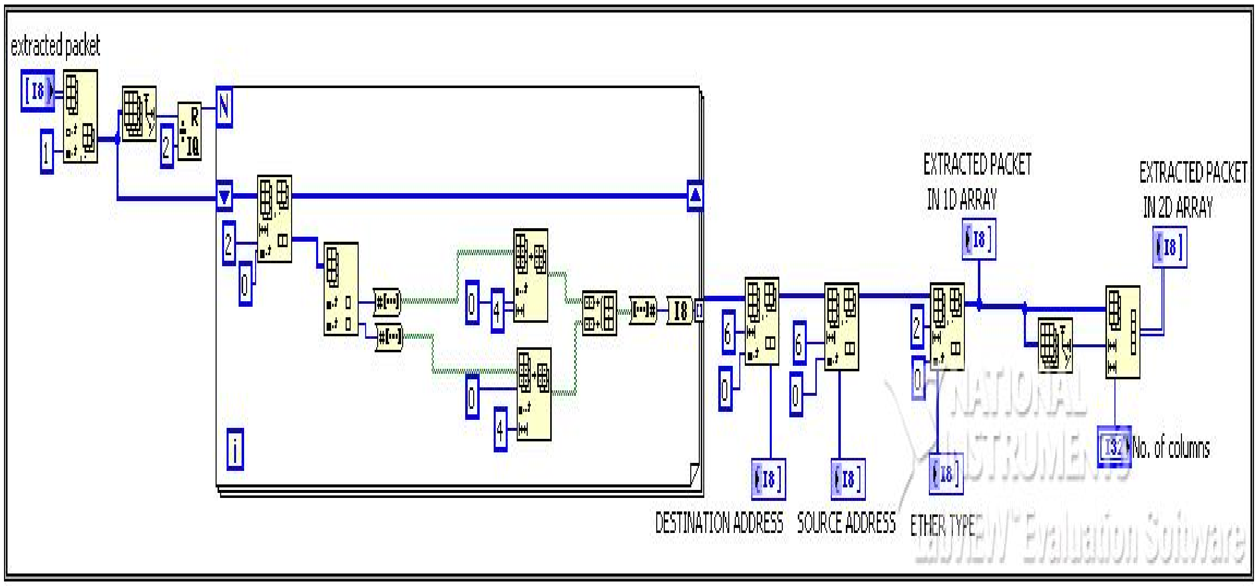

extracted packet
EXTRACTED PACKET IN 1D ARRAY
EXTRACTED PACKET IN 2D ARRAY
No. of columns
DESTINATION ADDRESS
SOURCE ADDRESS
ETHER TYPE
NATIONAL INSTRUMENTS LabVIEW Evaluation Software

Figure: 5.34 Block Diagram of Nibble to byte conversion of Extracted Packet

♦ **Description**

This sub- sequence is basically designed to convert the obtained CRC, Preamble & Extracted Packet in byte format for display purpose & further destination address, source address & ether type is truncated from the extracted packet.

- **Input:** CRC, Preamble &Extracted packet in nibble format
- **Output:** CRC, Preamble & Extracted packet in Byte format. Packet features like destination address, source address, ether type & payload

➢ **<u>Equivalent Flow Chart</u>**

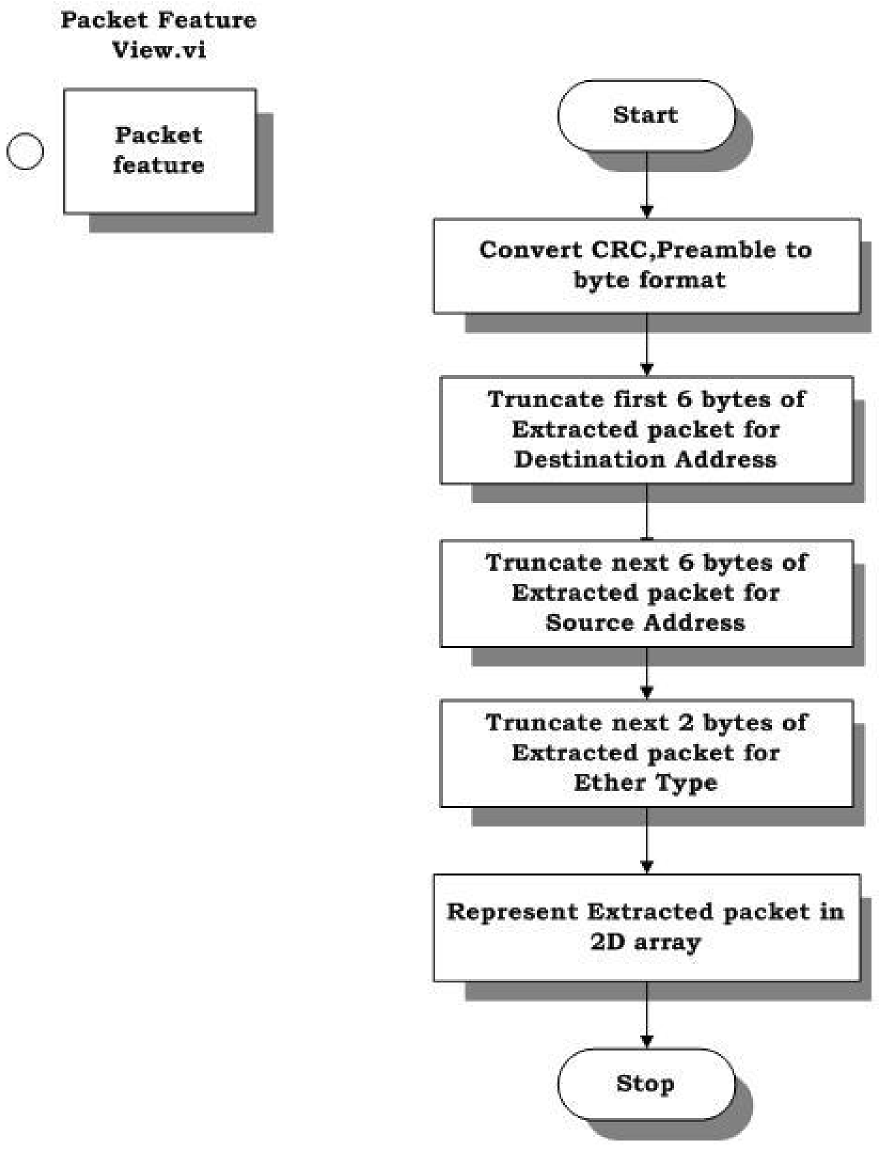

Packet Feature View.vi
Packet feature
Start
Convert CRC,Preamble to byte format
Truncate first 6 bytes of Extracted packet for Destination Address
Truncate next 6 bytes of Extracted packet for Source Address
Truncate next 2 bytes of Extracted packet for Ether Type
Represent Extracted packet in 2D array
Stop

Figure: 5.35 Equivalent flow Chart of Packet Feature.VI

M) Save Indexes .VI

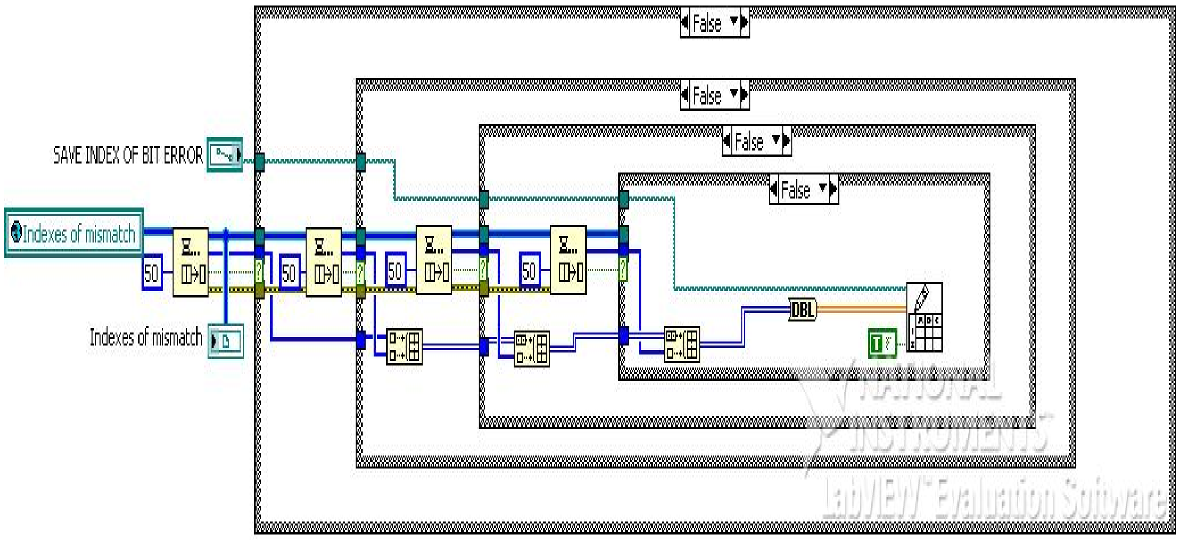


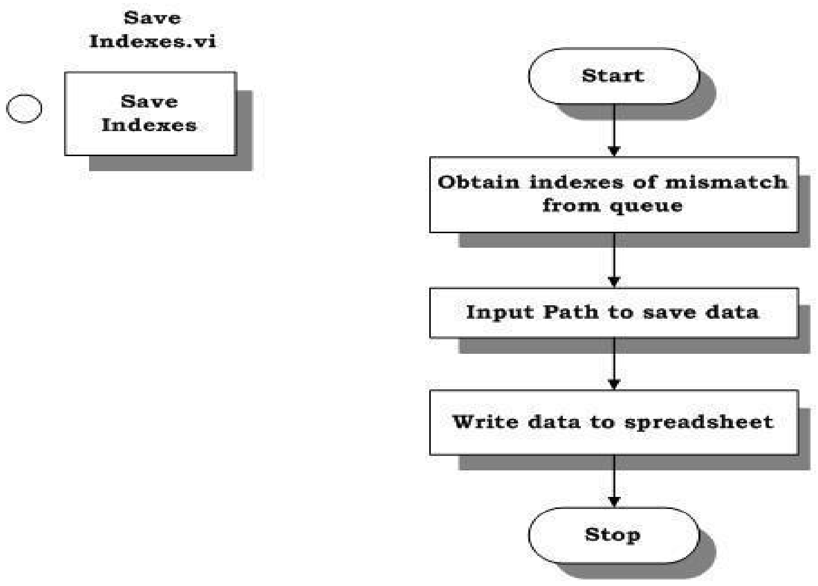
Figure: 5.36 Block Diagram of Save Indexes.VI

- **Description**

  This sub- sequence is basically designed to save the Indexes of Bit Error Data, index of mismatch is put in queue, four queues are used to speed up the process, data from queue is saved to a spreadsheet

- **Input:** Index of Mismatch from the comparison module to queue it, Path to save Index of Mismatch

  - **Equivalent Flow Chart**

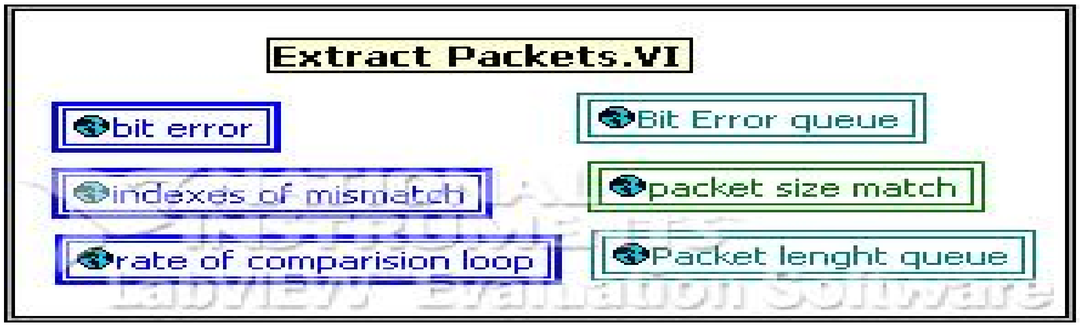


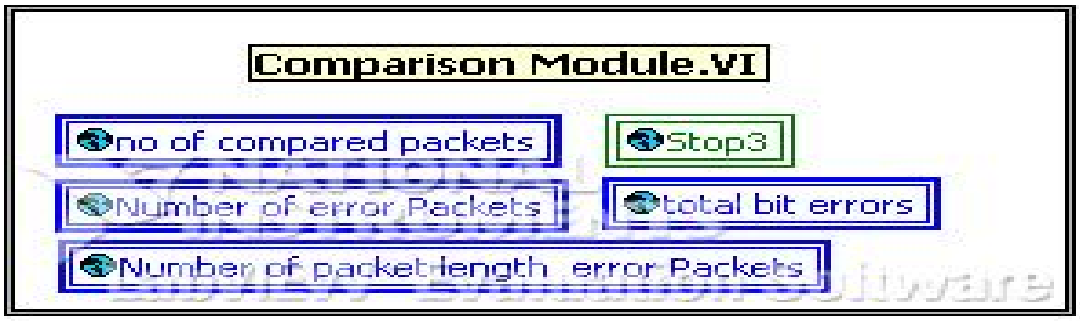
Save Indexes.vi
Save Indexes
Start
Obtain indexes of mismatch from queue
Input Path to save data
Write data to spreadsheet
Stop

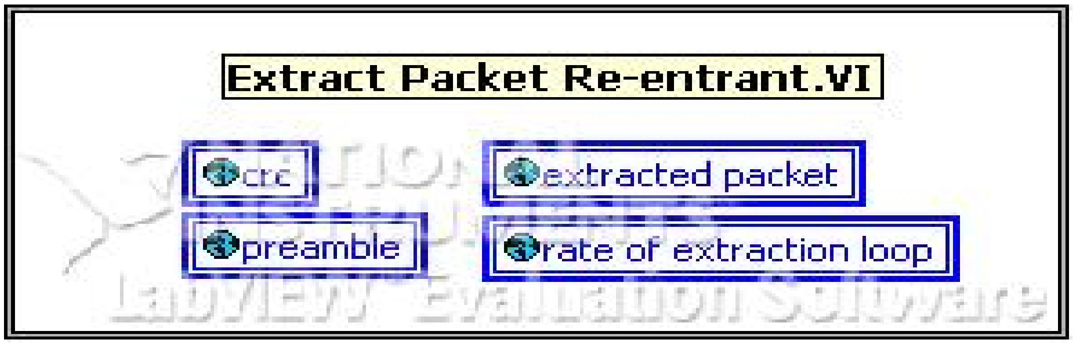
Figure: 5.37 Equivalent flow Chart of Save Indexes.VI

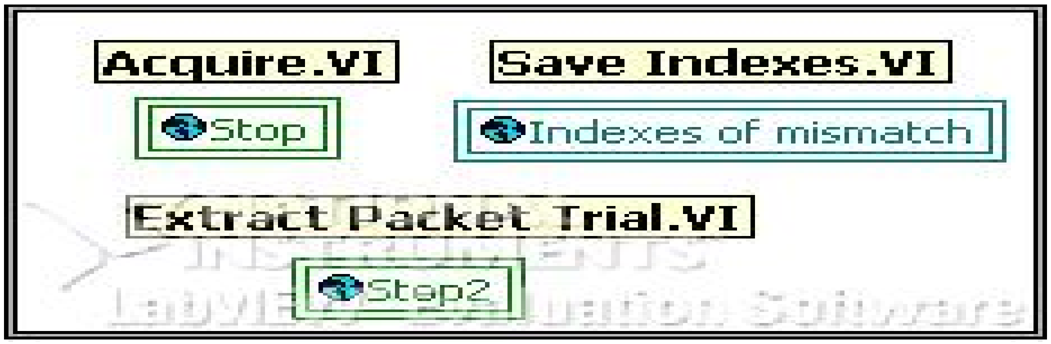
5.3.4) Global Variable Description

To get the relevant data out of the Sub VIôs global variables are used, different global variables used in different sub VIôs is listed below :









5.4) Conclusion

This chapter described the programming concepts of the developed software model & highlighted the different features of graphical programming language Lab View which dramatically decreased the development time & enabled the research code to be prototyped extremely fast. It explained the different modules developed to implement the control and monitoring application for the Validation set up which allowed the automated control of the Logic Analyzer.

CHAPTER -VI
Testing & Limitations

6.1) Introduction

This chapter presents the tests and measurements conducted to assess the developed software. It features the design process used in implementing software model, followed by case studies. Case study is classified based on mode of operation of Logic Analyzer. For each case study details of logic analyzer configuration file ,size of packet transmitted along with screen shots of front panel for different stages is presented .It also outlines the key issues faced during implementation and suggests its solution. It also features the limitations of the developed module.

6.2) Case formulation

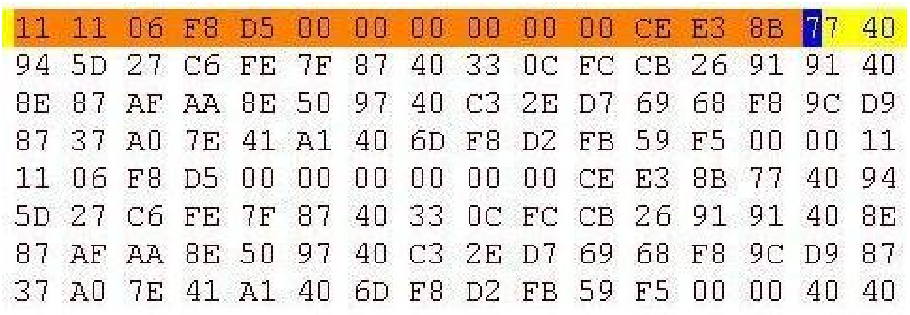
Case is formulated based on Mode of operation of logic analyzer .Logic analyzer can be operated in two modes namely state mode and timing mode. In ***state mode*** synchronous sampling is done & clock is provided by device under test. In ***timing mode*** asynchronous sampling is done & clock is provided by analyzer itself. Acquisition of data captured by logic analyzer through developed software model can be obtained in two modes namely continuous acquisition mode and single acquisition mode. In ***single acquisition mode*** the module acquire the data from logic analyzer once and in continuous acquisition mode module run the logic analyzer in infinite loop & keep on acquiring data after each triggering condition till it is stopped using stop button. Testing of module was done in both continuous and single acquisition mode for different packet size. Maximum and minimum packet sizes used for testing were as follows:

Minimum packet size: 64 bytes

Maximum packet size: 8000 bytes

6.2.1) Format of Packet

A packet is the basic unit of data transfer in a networked environment. Packets are individual chunks of data; flowing in a single direction Ethernet uses variable-length frames of data to transmit information from a source to one or more destinations. There are different kinds of packet floating around various networks. A packet that travels on top of nothing but a Layer 1 packet is a ***Layer 2 packet - the Ethernet packet***. It is the lowest level software packet seen on a LAN. Every Ethernet frame has two fields defined as the source and destination addresses (6 bytes each), which indicate the MAC address of the network devices where a frame originated and the MAC address of the network device where the frame is destined, respectively. Figure 6.1 shows the packet format in binary data stream. The *first* ***6 byte represents the destination address,*** *next* ***6 bytes represents the source address,*** next ***2 bytes represents the Ether Type and*** *rest is* ***Payload Field i.e. the data.***



```
11 11 06 F8 D5 00 00 00 00 00 00 CE E3 8B 77 40
94 5D 27 C6 FE 7F 87 40 33 0C FC CB 26 91 91 40
8E 87 AF AA 8E 50 97 40 C3 2E D7 69 68 F8 9C D9
87 37 A0 7E 41 A1 40 6D F8 D2 FB 59 F5 00 00 11
11 06 F8 D5 00 00 00 00 00 00 CE E3 8B 77 40 94
5D 27 C6 FE 7F 87 40 33 0C FC CB 26 91 91 40 8E
87 AF AA 8E 50 97 40 C3 2E D7 69 68 F8 9C D9 87
37 A0 7E 41 A1 40 6D F8 D2 FB 59 F5 00 00 40 40
```

Figure 6.1 Ethernet packet format

6.2.2) Performance Evaluation in State Mode of Logic Analyzer

➢ ***<u>Format of configuration file of logic analyzer in state mode are given below</u>:***

- ***Sampling****: State mode-synchronous sampling clocked by device under test.*
- ***Trigger position****: start*
- ***Trigger condition****: Find pattern 1 until pattern 2 occurs Trigger when TXEN=0 until TXEN=1*
- ***Acquisition depth****: 1M*
- ***Sample period****: 40ns ,clock of device under test MII mode=25MHz*

Case I: ***Transmission test results for Packet size: 7968 bytes.*** Table 6.1 depicts test results obtained in state mode of Logic Analyzer.

♦ ***Preamble=8 bytes, CRC=4 bytes,*** Size of Transmitted Packet =7968 bytes.

♦ ***Size of Packet captured by Logic Analyzer*** =. Preamble+CRC+ Packet size

♦ ***Size of Packet captured by Logic Analyzer*** =8+4+7968=7980bytes.

♦ ***Size of single packet to be acquired from Logic Analyzer in Nibbles***=15960 Nibbles

♦ **Rate of acquisition** represents the time by the acquisition module to acquire the data captured by the Logic Analyzer

Table 6.1 Test results for Packet size: 7968 bytes in State mode of Logic Analyzer

Sl No.	No. of samples acquired	No of packets received & compared	Rate of acquisition(ms)
1	16K	1	2239
2	32K	2	7627
3	64K	4	31669
4	128K	8	248982
5	256K	16	2076854

➢***Analysis Result:***

♦ All acquired packets matched the Expected Packet.

♦ More the no. of samples to be acquired acquisition time taken increases.

Case II: ***Transmission test results for Packet size: 64 bytes.*** Table 6.2 depicts test results obtained in state mode of Logic Analyzer.

♦ ***Size of Packet captured by Logic Analyzer = 8*** **+4+64=76 bytes**

♦ ***Size of single packet to be acquired from Logic Analyzer in Nibbles*****=152 Nibbles**

Table 6.2 Test results for Packet size: 64 bytes in State mode of Logic Analyzer

Sl No.	No. of samples acquired	No of packets compared	Rate of acquisition(ms)
1	1024	6	163
2	8K	53	779
3.	16K	107	2151

➢***Analysis Result:***

♦ All acquired packets matched the Expected Packet.

6.2.3) Performance Evaluation in Timing Mode of Logic Analyzer

➢ ***Format of configuration file of logic analyzer in timing mode are given below***

♦ ***Sampling****: Timing mode-Asynchronous sampling clocked internally by analyzer.*

♦ ***Trigger position****: start*

♦ ***Trigger condition****: Trigger when TXCLK=1*

♦ ***Acquisition depth****: 1M*

♦ ***Sample period****: 40ns*

Case I: ***Transmission test results for Packet size: 7968 bytes.*** Table 6.3 depicts test results obtained in Timing mode of Logic Analyzer.

♦ ***Size of Packet captured by Logic Analyzer*** =8+4+7968=7980bytes.

♦ ***Size of single packet to be acquired from Logic Analyzer in Nibbles***=15960 Nibbles

Table 6.3 Test results for Packet size: 7968 bytes in Timing mode of Logic Analyzer

Sl No.	No. of samples acquired	No of packets compared	Rate of acquisition(ms)
1	16K	1	1955
2	32K	2	10861
3	64K	4	44403
4	128K	8	265542

➢ ***Analysis Result:***

All acquired packets showed packet size mismatch when compared with the Expected Packet

Case II: ***Transmission test results for Packet size: 64 bytes.*** Table 6.4 depicts test results obtained in Timing mode of Logic Analyzer.

♦ ***Size of Packet captured by Logic Analyzer = 8*** **+4+64=76 bytes**

♦ ***Size of single packet to be acquired from Logic Analyzer in Nibbles*****=152 Nibbles**

Table 6.4 Test results for Packet size: 64 bytes in State mode of Logic Analyzer

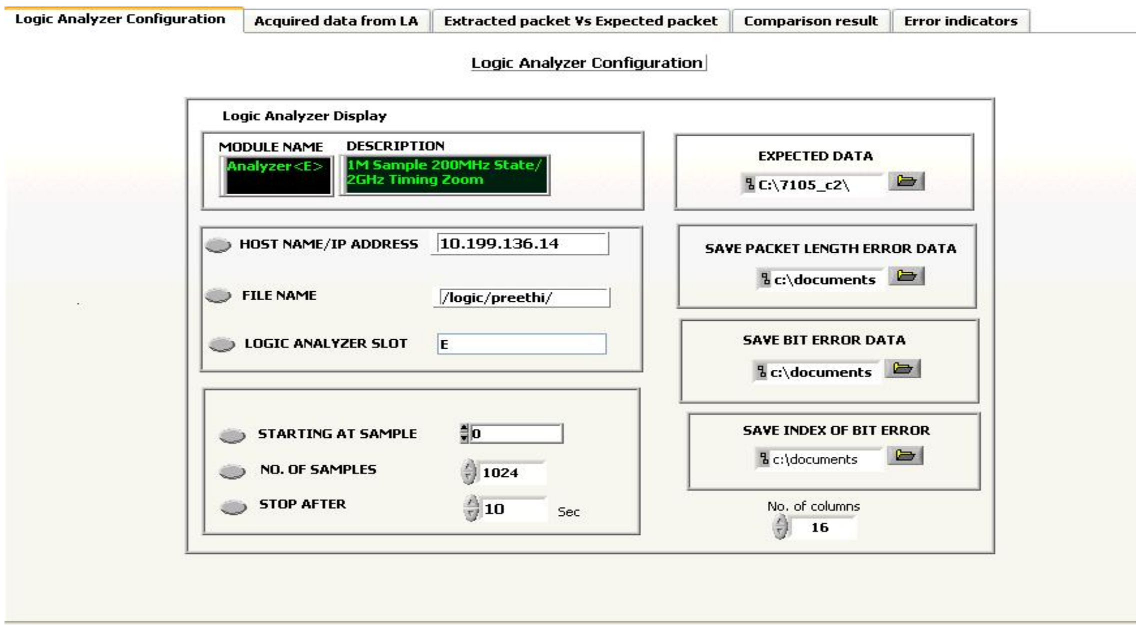
Sl No.	No. of samples acquired	No of packets compared	Rate of acquisition(ms)
1	1024	6	163
2	8K	53	779
3.	16K	107	2151

➢ ***Analysis Result:***

♦ Few packets were same as expected packet and few had bit error and packet length error.

6.3) Screen Shots of Graphic user interface

Screen shots of Graphic user interface after acquisition is presented in this section. Analysis was done in Timing mode of Logic Analyzer (LA) with packet size 64 bytes. A total of 6 packets were obtained from samples of 1024 bytes. Rate of acquisition was 163ms.Out of 6 packets 2 packets were same as expected packet and 3 packets had bit error and 1 packet had packet length error.



Logic Analyzer Configuration | Acquired data from LA | Extracted packet Vs Expected packet | Comparison result | Error indicators

Logic Analyzer Configuration

Logic Analyzer Display

MODULE NAME: Analyzer<E>
DESCRIPTION: 1M Sample 200MHz State/ 2GHz Timing Zoom

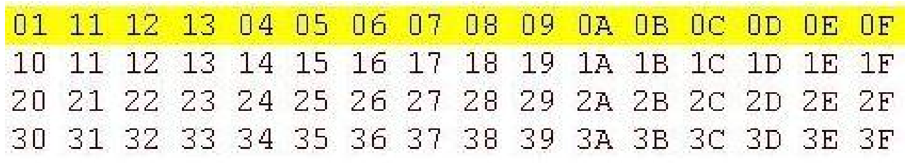
HOST NAME/IP ADDRESS: 10.199.136.14
FILE NAME: /logic/preethi/
LOGIC ANALYZER SLOT: E

STARTING AT SAMPLE: 0
NO. OF SAMPLES: 1024
STOP AFTER: 10 Sec

EXPECTED DATA: C:\7105_c2\
SAVE PACKET LENGTH ERROR DATA: c:\documents
SAVE BIT ERROR DATA: c:\documents
SAVE INDEX OF BIT ERROR: c:\documents
No. of columns: 16

Figure 6.2 Front Panel View after acquisition

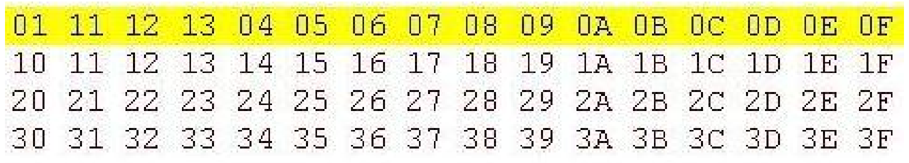
➢ **Transmitted packet**



```
01 11 12 13 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
```

Figure 6.3 Transmitted packet /Expected Packet (size 64 bytes)

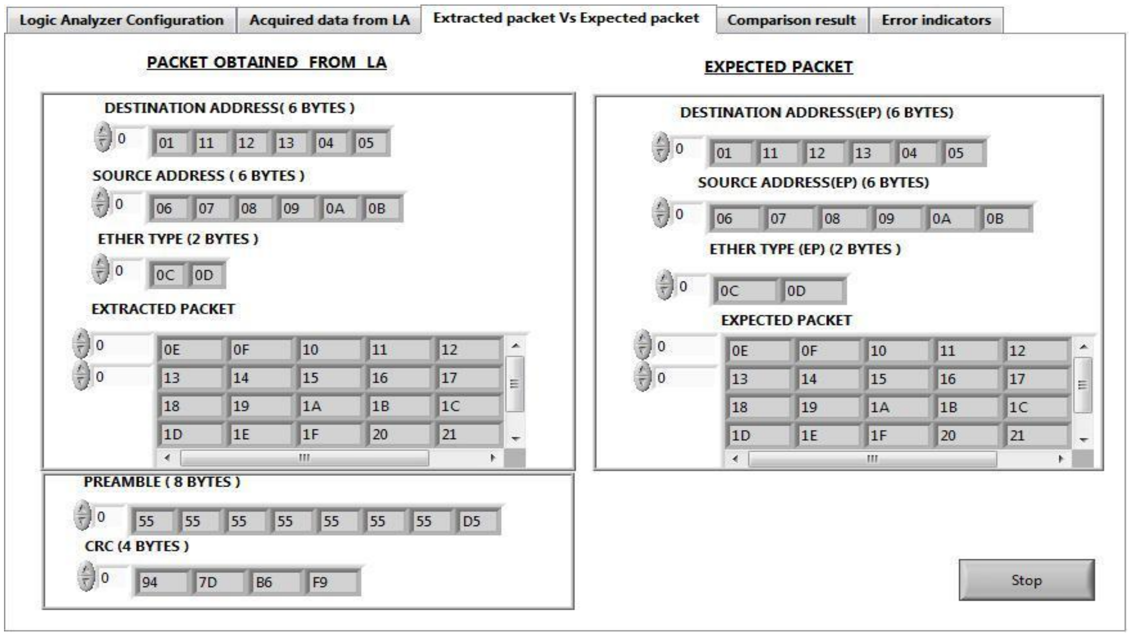
➢ **Received Packet analysed using Lab View**



```
01 11 12 13 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
```

Figure 6.4 Received Packet same as Expected Packet (size 64 bytes)

➢ **Extraction result indicating Packet Features**


Logic Analyzer Configuration
Acquired data from LA
Extracted packet Vs Expected packet
Comparison result
Error indicators
Stop

PACKET OBTAINED FROM LA

DESTINATION ADDRESS(6 BYTES): 0 | 01 11 12 13 04 05

SOURCE ADDRESS (6 BYTES): 0 | 06 07 08 09 0A 0B

ETHER TYPE (2 BYTES): 0 | 0C 0D

EXTRACTED PACKET: 0, 0

0E	0F	10	11	12
13	14	15	16	17
18	19	1A	1B	1C
1D	1E	1F	20	21

PREAMBLE (8 BYTES): 0 | 55 55 55 55 55 55 55 D5

CRC (4 BYTES): 0 | 94 7D B6 F9

EXPECTED PACKET

DESTINATION ADDRESS(EP) (6 BYTES): 0 | 01 11 12 13 04 05

SOURCE ADDRESS(EP) (6 BYTES): 0 | 06 07 08 09 0A 0B

ETHER TYPE (EP) (2 BYTES): 0 | 0C 0D

EXPECTED PACKET: 0, 0

0E	0F	10	11	12
13	14	15	16	17
18	19	1A	1B	1C
1D	1E	1F	20	21

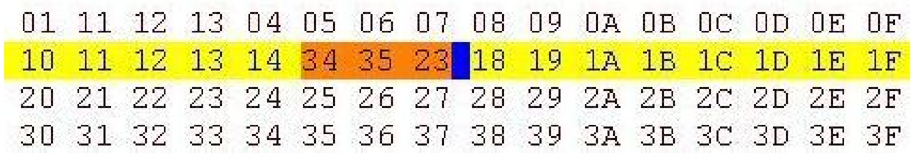
Figure 6.5 Front Panel showing Extracted Packet & Expected Packet

➢ **Analysed erroneous Packet**

♦ ***Bit Error Packet***

Index of mismatch: 26, 27, 28, 29, 30, 31(nibble position) & total bit error=6

Mismatch data 34, 35 & 23 instead of 15, 16 & 17

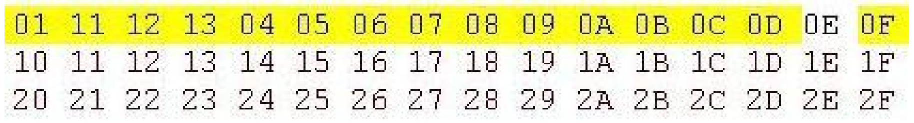


```
01 11 12 13 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 34 35 23 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
```

Figure 6.6 Bit Error packet of size 64 bytes

♦ ***Packet size mismatch***

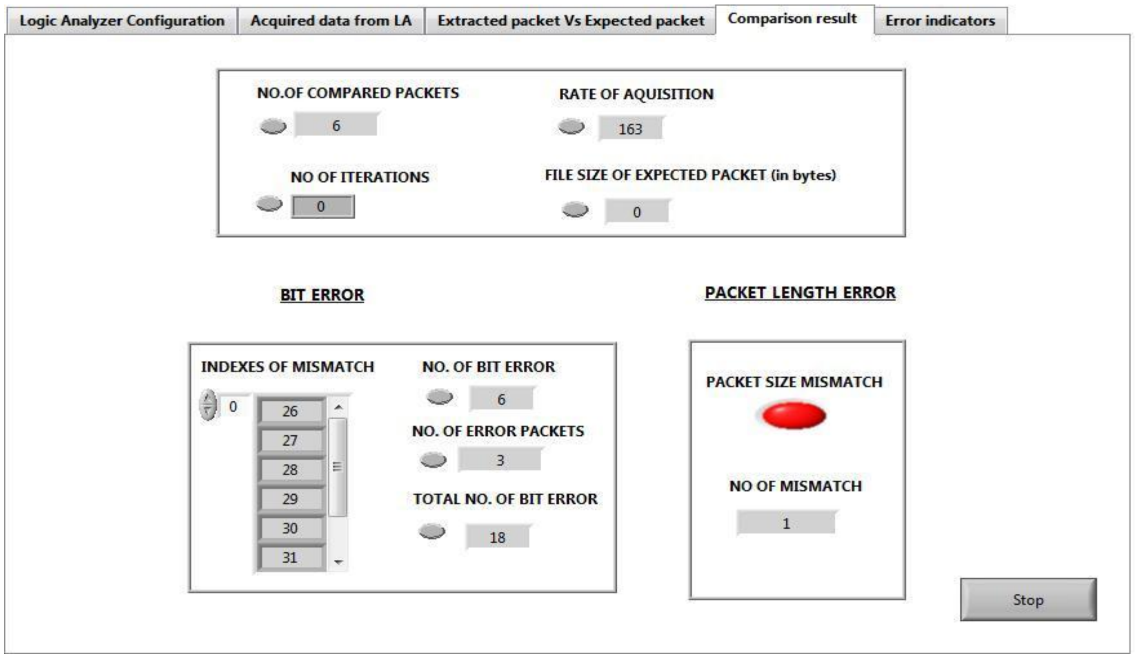
Received packet size (48 bytes) < Expected Packet size (64 bytes)



```
01 11 12 13 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
```

Figure 6.7 Packet with size mismatch (size 48 bytes)

➢ **Comparison results indicating error in received packet**


Logic Analyzer Configuration
Acquired data from LA
Extracted packet Vs Expected packet
Comparison result
Error indicators
Stop

NO.OF COMPARED PACKETS: 6
RATE OF AQUISITION: 163
NO OF ITERATIONS: 0
FILE SIZE OF EXPECTED PACKET (in bytes): 0

BIT ERROR
INDEXES OF MISMATCH: 0 — 26, 27, 28, 29, 30, 31
NO. OF BIT ERROR: 6
NO. OF ERROR PACKETS: 3
TOTAL NO. OF BIT ERROR: 18

PACKET LENGTH ERROR
PACKET SIZE MISMATCH: 🟥
NO OF MISMATCH: 1

Figure 6.8 Front Panel showing comparison results

6.4) Design Hurdles & Solutions

This section presents the issues encountered in implementing the software model
In each case, the solution for the issue is outlined.

➢ ***Initialization & Synchronization Issue***

Time taken to initialize the hardware and software module created the synchronization issues between the testing bench and the developed software model.

♦ **Solution**

To resolve the synchronization issue Transmission time & Activation time of Logic Analyzer was adjusted by introducing delay in validation software as well as in acquisition module.

➢ ***Data Interpretation issue in Timing Mode of Logic Analyzer***

In ***timing mode*** due to asynchronous sampling data interpretation problem aroused. Captured data didnôt match the transmitted data & hence extraction and comparison module of developed model didnôt work.

♦ **Solution**

To resolve data interpretation issue sampling frequency of analyzer & transmission frequency of data was made equal.

➢ ***<u>Performance issue in Timing Mode of Logic Analyzer</u>***

Analysis of Data captured in timing mode displayed erroneous packets & for large data it always displayed packet mismatch error.

♦ **Solution**

State mode of Logic Analyzer resolved the performance issue due to synchronous sampling of data.

➢ ***<u>Acquisition Depth issue</u>***

If the no. of samples to be acquired and the acquisition depth of Logic Analyzer are kept same acquisition module ran in infinite loop indicating error.

♦ **Solution**

To overcome the acquisition depth issue No. of samples to be acquired from the Logic Analyzer was kept less than the set acquisition depth of Logic Analyzer.

➢ ***<u>Processing time issue</u>***

As data to be acquired increased, processing time for analysing the extracted packet also increased.

♦ **Solution**

Producer consumer topology was used for designing the module.Clones/re-entrants of extraction and comparison modules were used & Expected Packet was converted into nibbles which eased the comparison and reduced the processing time.

➢ ***<u>Acquisition time issue</u>***

As no. of signals to be acquired from the test bench increased, acquisition time of software model increased

Solution: Data lines were clubbed to form single channel which increased the acquisition speed & hence reduced the acquisition time

6.5) Limitations

- ***Slow Response:***

ActiveX interface and LAN connection between Logic Analyzer & Lab view slow down the response of the software model. Maximum of 64K data can be acquired without compromising time. Performance detoriates when no. of samples to be acquired is increased.

- ***Non Generic Design:***

The Design of the developed software model is system specific; it is specifically designed for Ethernet Validation requirements so it cannot be used for other Validation Applications.

- ***No Triggering flexibility:***

Triggering cannot be set from the software model it is completely controlled by the set configuration file of the Logic Analyzer.

- ***Incompatibility for other modes of interfacing:***

Designed Software module is tested for layer2 protocol and MII mode of interfacing only. As per design it is compatible for 4 data lines ,if no of data lines are increased/decreased the designed extraction & comparison module doesnȇt work hence compatibility issue arises for other interfacing modes like RMII,GMII,RevMII .

- ***Data Loss in Continuous Acquisition Mode:***

As Logic Analyzer takes some time to load the configuration file after each acquisition, data is lost.

6.6) Conclusion

This chapter presented the tests and measurements conducted to assess the developed software model. It featured the test cases conducted in both state mode and timing mode of logic analyzer. The different design issues their solutions & limitations were discussed in detail. The next chapter discusses the conclusions that were developed from the research and any future work that could be done to add to the already powerful process.

CHAPTER VII
Conclusions & Further Scope of Work

7.1) Introduction

This chapter highlights the accomplishments of this thesis; it contains two brief sections that cover the final conclusions of the research and any future work that could be done to follow this work.

7.2) Conclusions

Ethernet is a Layer 2 switched technology; a hardware loopback might not be the perfect test approach. The integrity of an Ethernet frame is verified at each switching element, and a single bit error will result in the entire frame being discarded. The error bit will never get to the analyzer and the analyzer will declare a frame loss. For this reason, conventional testing methodologies are no longer sufficient for performance testing of Ethernet networks. There are multiple issues like data loss, data integrity, error tracking etc, and network element manufacturers and service providers require automation of the test equipments that can provide solutions to engineers to fully validate and benchmark their Ethernet network through comprehensive testing.

The implementation of developed software model has proven to be highly versatile in achieving fast error tracking. Lab View≎s ideal prototyping environment enabled the validation problem to be resolved quickly. The main factors of the developed model which makes it versatile is that the acquisition of data & its analysis is done simultaneously, it can track the erroneous packet and also the error position for each iteration. Once the test is completed it provides the performance metrics of the Ethernet network indicating the total no of erroneous packets, the frame size, test duration and number of test iterations etc.

The Application development environment Lab VIEW made the software model more modular, reusable & analysis independent from the electronics interface issues. The use of advanced programming concepts such as multi-threading, queuing, data-sharing, and user-interface control has made this model more user friendly. As Lab View handles most memory management automatically and the details of complex structures such as arrays, the user need not to focus on the algorithm being prototyped. In addition, the front panel / block diagram structure has eliminated the need for developing a separate user-interface software module required in most text-based languages.

The developed software model has been thoroughly tested in ST Micreoelectronics and proven to work satisfactorily, it has not only **reduced the error tracking problem** during validation but also **the time required** to do the validation.

.

7.3) Further Scope of Work

This work is only a prototype, since it is Ethernet system specific and can only be used in MII mode of interfacing and is limited to only four data lines. Future work will need to focus on improving the software so that it can be used for other mode of interfacing as well as other validation process.

.

1. Improving Software

There are a few improvements that can be done to the software. It would be desirable to expand the capabilities of the software so that it can be used for other modes of interfacing like RMII, GMII & RevMII. Another addition would be to include Triggering flexibility from the software itself. This software can be made more generic so that it can be used for other validation process such as audio & video applications, also it can be enhanced to work satisfactorily in timing mode of Logic Analyzer.

2. Improving Synchronization

If the validation set up is also controlled using Lab view many of the synchronizing issues will resolve. It will synchronize the transmission timing and the activation timing of Logic Analyzer which will further minimize the data loss during continuous acquisition mode.

References

[1] National Instruments Course manual ñLab VIEW Basics Iò, Course Software Version 6.0, September 2000 Edition, Part Number 320628G-01

[2] National Instruments Course manual ñLab VIEW Basics II" Course Software Version 6.0, September 2000 Edition, Part Number 320629G-01

[3] National Instruments Lab VIEW ósystem identification toolkit manualô http://www.ni.com/pdf/manuals/371001b.pdf

[4] National Instruments Corp., Getting started with Lab VIEW, P/N 3323427A-01.

[5] National Instruments Corp., Lab VIEW user manual, P/N 320999E-01

[6] National Instruments, Data Acquisition Fundamentals, Application Note 007

[7] National Instruments Lab VIEW ócontrol design toolkit manualô, P/N 371057d . http://www.ni.com/pdf/manuals/371057d.pdf

[8] National Instruments Lab VIEW ósimulation module manualô. P/N 3371013a http://www.ni.com/pdf/manuals/371013a.pdf

[9] B. R. Kumar, K. Sridhar an, and K. Srinivasan, ñThe Design and Development of a Web-based Data Acquisition System,òIEEE Trans. Instr. and Measure., Vol. 51, No. 3, June 2002.

[10] A. B. Buckman, Computer-Based Electronic Measurement, 2000; Prentice Hall, Inc., Upper Saddle River, New Jersey.

[11] R. H. Bishop, Learning with Lab VIEW, 1999; Addison-Wesley.

[12] L. K. Wells, and J. Travis, Lab VIEW for Everyone Graphical Programming Made Even Easier, 1997; Prentice Hall, Inc., Upper Saddle River, New Jersey.

[13] J. Essik, Advanced Lab VIEW Labs, 1999; Prentice Hall, Inc.,Upper Saddle River, New Jersey.

[14] M. L. Chugani, Lab VIEW Signal Processing, 1998; Prentice Hall, Inc., Upper Saddle River, New Jersey.

[15] J. Y. Beyon, Hands-On Exercise Manual for Lab VIEW Programming, Data Acquisition and Analysis, 2001; Prentice Hall, Inc., Upper Saddle River, New Jersey.

[16] Barry E. Paton 1999,ò Lab VIEW Graphical Programming for Instrumentationò. Prentice Hall PTP, New Jersey, U.S.A

[17] Gary W. Johnson, 1994. ñLab VIEW Graphical Programmingò. McGraw-Hill, Inc., New York, U.S.A

[18] STi7xxx programming Manual, Volume 1: System, March 2008, 8065505 Rev A ï DRAFT 5.

[19] STi7xxx Validation Module schematics, March 2008, 435/19352 Rev 06.

[20] STi7xxx Top Level Architecture Manual, March 2008, ADCS 8087611 Rev A.

[21] ST Micro Connect 2 Manual, May 2008, 7912386 Rev F.http://www.st.com

[22] Synopsisôs Design Ware Cores Ethernet MAC Universal Manual., March 07, release 3.30a. http://www.synopsys.com

[23] DP83865 Gig PHYTER® V10/100/1000 Ethernet Physical Layer Manual, October 2004. http://www.national.com

[24] Spirentôs Smart Applications 3.00, P/N 341-1195-001 Rev. A, 1/04. http://support.spirentcom.com.

[25] 16700 Series Logic Analysis System Manual.ò Product Overview, P/N 5968-4632E http://www.agilent.com

[26] Andrew S.Tanenbaum, Computer Networks, Prentice Hall PTP, New Jersey, U.S.A, iv edition, 2003

[27] C. E. Spurgeon, Ethernet: The Definitive Guide, Ird. Ed., O'RELLY, 2000.

[28] IEEE 802.3/ISO 8802-3 - Information processing systems - Local area networks - Part 1, Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications, 2nd edition, 21 September 1990.

[29] ANSI/IEEE Std 802.3, ISO/IEC 802.3, Carrier sense multiple access with collision detection (CSMA/CD) and physical layer special, IEEE Std 802.3ad-2000, Section 1, Section 2, Section 3

[30] IEEE Std 802.3-2002. [10] IEEE Std 802.3-2002. [B1] Ethernet Version 2.0, A Local Area Networkð Data Link Layer and Physical Layer Specifications. Digital Equipment Corp., Intel Corp., and Xerox Corp., November 1982

[31] E.P. Elkins, ñData acquisition and control using ETHERNETò, Nuclear Instrumentation Methods Phys. Res. A., vol. A247, no.1, pp. 197-201, June 2005

[32] Wittenmar, B., Nillson, J., Torngren, M., ñTiming problems in real-time control Systemsò, Proceedings of the American Control Conference, pp. 2000-2005 2002.

[33] Ciscoôs Internetworking Technologies Handbook, 1992,Cisco Networking matters

[34] Harold D. Ethernet Everywhere. Control Engineering, pages 64-73. 1999., Prentice Hall PTP, New Jersey, U.S.A.

[35] J. D. Decotignie, "A perspective on Ethernet-TCP/IP as a field bus," in Proceedings of LORIA.4th International Conference on Field bus Systems and their Applications, 15-16 Nov. 2001, Nancy, France, 2002.

[36] J. P. Thomesse, "Field bus Technology in Industrial Automation," Proceedings of the IEEE, Vol. 93, No.6, June, 2005.

[37] Andy Baldmanôs, BIT ERROR RATIO TESTING: HOW MANY BITS ARE ENOUGH? , UNH Interoperability Lab March 18, 2003.

[38] An Overview of the Electrical Validation of 10BASE-T, 100BASE-TX, &1000BASE-T Devices Application Note 1600 Agilent Technologies, Inc. 2008

[39] John Pickerd, Patent Application, Tektronix Inc. 7424-US1.

[40] Physical Layer Compliance Testing for 100BASETX Application Note http://www.tektronix.com/ethernet

[41] White Paper: Validate LAN installations for optimal service delivery, Feb 2009, Fluke Networks. http://www.flukenetworks.com

Preethi G S

E-mail: gspreethiii@yahoo.co.in **Phone:** +91-9968093982 (M.) ~ 0120-4111424 (R.)
Passport No. –E4076695

AN OVERVIEW

A competent professional with over **4 years** of qualitative experience in teaching and auditing in an Engineering Institution. Have handled **PLC based projects** and also worked on application software's like **MATLAB, Pspice , MiPower, and LABVIEW**. During the tenure have successfully guided and completed numerous engineering projects. **Awarded Certification on Internal Auditing** of Quality Management System. (ISO 9001:2000) by Nathan & Nathan Consultants. Secured GATE Score 2006: 325 (AIR 2785/19527). **Topped in 1st semester M.E with 78% & 2nd semester M.E with 76.7%** in **Delhi College of Engineering, Delhi**.

EDUCATION

- Currently pursuing M.E in Control & Instrumentation from Delhi College of Engineering, Delhi with **78% in first semester and 76.7% in 2nd semester, Aggregrate-77.3%.**
- **GATE Score – 325 (Air 2785/19527)**
- **B.Tech** (Electrical & Electronics) from Govt. Engineering College, Kasaragod, Kerala with **74.7%** in 2002.
- Class 12 from Kendriya Vidyalaya, New Delhi (CBSE) in 1998 with **79%** marks.
- Class 10 from Kendriya vidhyalaya, New Delhi (CBSE) in 1996 with **84.6%** marks.

ACADEMIC ENHANCEMENTS/ TRAINING

- **Major project on "DATA ACQUISITION & ANALYSIS FOR ETHERNET SYSTEM USING Lab VIEW" done in STMicroelectronics, Greater Noida.**
- Introduction & Application of PLC organized by AEI & EEE Dept. MESECE, from 16th to 20th Aug.
- International Seminar on **Non Conventional Energy, Renewables, Energy Efficiency and Conservation** of Prithvi 2005 Global Eco Meet on 20th & 21st Feb 2005
- Short course in **Electric Power Management** conducted by The Centre For Excellence in Engineering & Business Administration on 10th Jan 2004
- Attended a national workshop on **GNU LINUX Technologies** for generation next at dept of CSE, MESCE from 29 Sep to 1st Oct 03.

____

Accomplishments

- **Topped in 2nd semester,M.E in DCE with 76.7%**
- **Topped in 1st semester, M.E in DCE with 78%.**
- **Member of IEEE**
- **Member of ISTE**
- **Life Member of ECS**
- **Certification on Internal Auditing** of Quality Management System. (ISO 9001:2000) by Nathan & Nathan Consultants

PAPERS PRESENTED.

- On **WiTricity-Wireless Electricity** in **Delhi College of Engineering, Delhi**
- On **Self-Adjusting Solar Panel** in International Interdisciplinary Conference on Sustainable Technologies for Environmental Protection (ICSTEP2006)
- An **Overview to MiPower** for the three day workshop on application software's in electrical &electronics engineering organized by EEE Dept MESCE.

PROFESSIONAL EXPERIENCE

NIEC, Delhi **(Feb 1st, 07- July 30th,07)**

Worked as Lecturer in EEE dept.

MES College of Engg., kerala **(July, 03 to Dec, 06)**

As Lecturer in EEE Dept.
Key Highlights:

***Courses taught**:* Industrial Drives, Control System, Power System, Machines, Basic Electrical & Measurements

***Laboratory Handled**:* Machines, measurements & advanced electrical engg. Labs

***Application Software'**s worked on*: MATLAB, PSpice, PLC, MiPower,Lab VIEW

***Teaching evaluations**:* 1 year experience as External Examiner in Calicut University

Projects Guided:

- Self Adjusting Solar Panel
- Analysis of 440 KV substation using MiPower
- Robotic Path Finder

PROFESSIONAL AFFILIATIONS:

- Life Member of ECS(Energy Conservation Society)
- Member of ISTE

COMMUNITY INVOLVEMENT/ VOLUNTEER WORK

- *Representative from EEE Dept. for IEEE and was an active member for AKSC'07(All Kerala Students Conference) organized in MESCE Placement*
- *ISO Cell Representative for EEE Dept.*
- *Placement Cell Representative for EEE Dept.*
- *Internal Quality Auditor for 3 years*

PERSONAL DETAILS

Date of Birth	:	18th July, 1981
Address	:	A-403, Sector -6, Vasundhara Valley Apartments, Vasundhara, Ghaziabad, UP- 201012

References:

Mr. Parmod Kumar
HOD, EEE
Delhi College of Engineering
#01127871047

Mr. Sudeep.K.Srivastava
Manager, HED Dept.
ST Microelectronics
Greater Noida
#9891923884